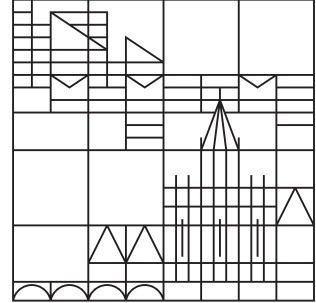


Universität
Konstanz



Extending the OLAP Technology to Handle Non-Conventional and Complex Data

DISSERTATION

ZUR ERLANGUNG DES AKADEMISCHEN GRADES
DES DOKTORS DER NATURWISSENSCHAFTEN
AN DER UNIVERSITÄT KONSTANZ
IM FACHBEREICH INFORMATIK UND INFORMATIONSWISSENSCHAFT

vorgelegt von
Svetlana Mansmann

September 2008

Tag der mündlichen Prüfung: 29. September 2008

Referenten: Prof. Dr. Marc H. Scholl, Universität Konstanz

Prof. Dr. Daniel A. Keim, Universität Konstanz

Acknowledgments

*In any moment of decision the best thing you can do is the right thing,
the next best thing is the wrong thing, and the worst thing you can do is nothing.*
Theodore Roosevelt

THE years I spent on this PhD research mark a challenging and demanding, yet rewarding period in my life. I was confronted with a variety of intriguing research problems in the field data warehousing and had an opportunity to cooperate with excellent researchers from different parts of the world. I want to thank all persons and institutions that supported me during this research undertaking:

- ◆ First of all, I would like to express my deepest gratitude to my supervisor Marc H. Scholl for his continued guidance and support, and for having encouraged me to pursue a PhD degree in the first place. In fact, it is through his lectures during my Master's studies that I discovered passion for databases and got tightly integrated into the workgroup's research and teaching activities as a student assistant.
- ◆ I also want to thank professor Daniel A. Keim, who has been my second supervisor within the PhD Graduate Program, for valuable insights and continuous collaboration in the field of visual analysis.
- ◆ I am very indebted to my dearly beloved husband Florian Mansmann, who has also been my major collaboration partner and supporter in the past years. We conducted a number of joint projects and coauthored some publications.
- ◆ I would like to thank my external collaboration partners Dr. Thomas Neumuth and Dr. Oliver Burgert from the Innovation Center Computer Assisted Surgery in Leipzig, Dr. Alfredo Cuzzocrea from the University of Calabria in Italy, Dr. Juan Carlos Trujillo Mondéjar from the University of Alicante in Spain, and Dr. Il-Yeol Song from Drexel University, USA.
- ◆ Many thanks go to my colleagues, especially the members of the PhD Graduate Program for fruitful joint work, assistance, expertise, and all the fun we had.
- ◆ I thank student assistants Roman Rädle, Andreas Weiler, Marion Herb, Matthias Röger, Elena Povalyayeva, and Michael Seiferle for contributing to the implementation of the presented solutions.
- ◆ I am more than grateful to Barbara Lühke for proof-reading the entire thesis as well as to my husband Florian Mansmann, my sister Olya Dadressan, my mother Svetlana Vinnik, and my friend Carola Welinsky for proof-reading parts of the work.
- ◆ Last not least, I am grateful to my yet unborn baby, who accompanied me through the last months of accomplishing the thesis hardly giving me any extra troubles and allowing me to complete my PhD research in due time.

This work has been partially supported by the German Research Society (DFG) under the grant GK-1042, *Explorative Analysis and Visualization of Large Information Spaces*, University of Konstanz. The first 18 months of my PhD research were funded by the Ministry of Science and Arts of the Federal State Baden-Württemberg within the pilot project “Prognosemodell”.

Publications

- [1] Svetlana Vinnik and Marc H. Scholl. UNICAP: Efficient Decision Support for Academic Resource and Capacity Management. In *TCGOV2005: Proceedings of the TED Conference on e-Government*, LNAI, Bozen-Bolzano, Italy, 2005. Springer.
- [2] Svetlana Vinnik and Marc H. Scholl. Decision Support System for Managing Educational Capacity Utilization in Universities. In *Proceedings of the International Conference on Engineering and Computer Education (ICECE'05)*, Madrid, Spain, 2005. ICECE'05 Best Paper.
- [3] Svetlana Vinnik and Florian Mansmann. From Analysis to Exploration: Building Enhanced Visual Hierarchies from OLAP Cubes. In *EDBT 2006: Proceedings of the 10th International Conference on Extending Database Technology*, LNCS, pages 496–514. Springer, 2006.
- [4] Svetlana Mansmann and Marc H. Scholl. Extending Visual OLAP for Handling Irregular Dimensional Hierarchies. In *DaWaK'06: Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery*, LNCS, pages 95–105. Springer, 2006. DaWaK'06 Best Paper.
- [5] Svetlana Mansmann, Marc H. Scholl, Daniel A. Keim, and Florian Mansmann. Exploring OLAP Aggregates with Hierarchical Visualization Techniques. In *Poster Compendium of 12th IEEE Symposium on Information Visualization (InfoVis 2006)*, pages 136–137, 2006. (Poster paper).
- [6] Florian Mansmann and Svetlana Vinnik. Interactive Exploration of Data Traffic with Hierarchical Network Maps. *IEEE Transactions on Visualization and Computer Graphics (Special Issue on Visual Analytics)*, 12(6):1440–1449, 2006.
- [7] Svetlana Mansmann, Florian Mansmann, Marc H. Scholl, and Daniel A. Keim. Hierarchy-driven Exploration of Multidimensional Data Cubes. In *BTW 2007: 12. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web*, volume 103 of *LNI*, pages 96–111, Aachen, Germany, 2007. GI.
- [8] Svetlana Mansmann and Marc H. Scholl. Exploring OLAP Aggregates with Hierarchical Visualization Techniques. In *SAC 2007: Proceedings of 22nd Annual ACM Symposium on Applied Computing, Multimedia & Visualization Track*, pages 1067–1073, New York, NY, USA, 2007. ACM Press.
- [9] Svetlana Mansmann and Marc H. Scholl. Decision Support System for Managing Educational Capacity Utilization. *IEEE Transactions on Education*, 50(2):143–150, 2007.
- [10] Svetlana Mansmann and Marc H. Scholl. Empowering the OLAP Technology to Support Complex Dimension Hierarchies. *International Journal of Data Warehousing and Mining*, 3(4):31–50, 2007. (Invited Paper).

- [11] Svetlana Mansmann, Thomas Neumuth, and Marc H. Scholl. OLAP Technology for Business Process Intelligence: Challenges and Solutions. In *DaWaK'07: Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery*, LNCS, pages 111–122. Springer, 2007. DaWaK'07 Best Paper.
- [12] Svetlana Mansmann, Thomas Neumuth, and Marc H. Scholl. Multidimensional Data Modeling for Business Process Analysis. In *ER 2007: Proceedings of the 26th International Conference on Conceptual Modeling*, LNCS, pages 23–38, Auckland, New Zealand, 2007. Springer.
- [13] Svetlana Mansmann and Marc H. Scholl. Extending the Multidimensional Data Model to Handle Complex Data. *Journal of Computing Science and Engineering*, 1(2):125–160, 2008. (Invited tutorial paper).
- [14] Marc H. Scholl and Svetlana Mansmann. Visual OLAP (Online Analytical Processing). In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*. Springer, 2008. (To appear).
- [15] Alfredo Cuzzocrea and Svetlana Mansmann. Models, Issues, and Techniques in OLAP Visualization. In John Wang, editor, *Encyclopedia of Data Warehousing and Mining, 2nd Edition*. Information Science Reference, 2008.
- [16] Svetlana Mansmann and Marc H. Scholl. Chapter 4.31: Empowering the OLAP Technology to Support Complex Dimension Hierarchies. In John Wang, editor, *Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications*, pages 2164–2184. Information Science Reference, 2008.
- [17] Thomas Neumuth, Svetlana Mansmann, Marc H. Scholl, and Oliver Burgert. Data Warehousing Technology for Surgical Workflow Analysis. In *CBMS 2008: Proceedings of the 21st IEEE International Symposium on Computer-Based Medical Systems*, pages 230–235, Jyväskylä, Finland, 2008. IEEE Computer Society.
- [18] Svetlana Mansmann and Marc H. Scholl. Visual OLAP: a New Paradigm for Exploring Multidimensional Aggregates. In *CGV2008: Proceedings of the 2nd IADIS International Conference on Computer Graphics and Visualization*, pages 59–66, Amsterdam, Netherlands, 2008. IADIS.
- [19] Svetlana Mansmann, Thomas Neumuth, Oliver Burgert, Matthias Röger, and Marc H. Scholl. Conceptual Data Warehouse Design Methodology for Business Process Intelligence. In *Complex Data Warehousing and Knowledge Discovery for Advanced Retrieval Development: Innovative Methods and Applications*, Advances in Data Warehousing and Mining (ADWM). IGI Publishing, 2008. (Invited book chapter, to appear).

Abstract

COMPREHENSIVE DATA ANALYSIS has become indispensable in a multitude of applications. Data warehousing and OLAP evolved in the 90s as a response to this need in business environments and proved their nearly universal applicability for decision support. In the last decade, data warehouses went beyond classical business performance oriented tasks and reached out for novel application domains, such as government, science and research, medicine, web usage, network security, etc. Although many research areas in the field of data warehousing and OLAP have reached a state of maturity, new challenges arise when applying the traditional technology in such non-conventional contexts. The exact causes of unsatisfactory performance are manifold, ranging from the rigidity of the conceptual data model, adopted logical schema, and implementation strategy to available data transformation techniques, prevailing metadata modeling standards, limited set of supported operators, and frontend issues.

The aim of this thesis is to fundamentally extend the functionality of the widely adopted OLAP framework in order to adequately handle usage scenarios not supported by the standard setting. More specifically, we investigate the limitations of the state-of-the-art relational OLAP systems, analyze the requirements of comprehensive data analysis, introduce respective extensions at the conceptual, logical, and metadata level and demonstrate how the extended model can be implemented at both the backend and the frontend layer. Real-world case studies from the domains of academic administration and medical engineering are used for exemplifying modeling challenges. Two complimentary ideas determine the contribution of this work: *i)* to extend the multidimensional data model that represents the very foundation of OLAP and *ii)* to enrich the frontend layer with innovative features for visual analysis of large and complex data volumes.

The primary focus is on extending the backend functionality of data warehousing, obtained by inspecting the requirements of novel application domains, identifying bottle-necks of the conventional OLAP technology with respect to those requirements, and searching for appropriate solutions at the conceptual level. Since the ultimate worth of the resulting extended conceptual data model is determined by its implementability into a logical schema, we propose a set of corresponding conceptual-to-logical mappings and transformation techniques for ensuring correct aggregation. The central contribution of this part of the work is the extended conceptual model expressed in terms of formal definitions and a graphical notation *X*-DFM (Extended Dimensional Fact Model). *X*-DFM consists of a set of visual modeling constructs as well as a set of guidelines for designing multidimensional schemes in accordance with the defined formal semantics. Both the formal and the graphical model provide three levels of abstraction – the lower, the intermediate, and the upper level – to support successive layering of the conceptual scheme at different design stages.

The main features of the extended model are the unification of the multidimensional space as a foundation for handling semantically related elements, measure aggregability and additivity constraints, many-to-many relationships between facts and dimensions and between hierarchy levels in a dimension, fact schemes with no measures, heterogeneity in facts and dimension hierarchies, degeneration of facts, dimensions, and roll-up relationships, optional and partial related roll-up relationships, and various kinds of irregularity in dimension hierarchies at scheme and instance levels. Besides, the model supports object-oriented properties, such as

generalization and specialization in fact and dimension schemes, different roles of an element, associations and self-associations of fact schemes, modeling of abstract, derived and dependent elements.

The feasibility of the proposed conceptual model is demonstrated by describing how the former can be mapped to a relational representation by formulating general guidelines for logical design. A well-established requirement of summarizability is used to determine the necessary scheme and/or instance transformations. The issue of metadata modeling is also considered. Metadata plays a prominent role in data warehouse systems by acting as an intermediary between different layers and components of the system architecture. In particular, we consider the metadata of the application layer as the former is responsible for capturing the multidimensional semantics behind “plain” data tables.

The other part of our research concentrates on the frontend functionality. End-users typically interact with OLAP data using exclusively visual tools, which allow them to conveniently navigate to the subset of interest and explore it using various visual layouts and interaction techniques. Therefore, the overall capabilities of the analysis ultimately depend on the functionality of the provided frontend tools. Aware of the deficiencies of standard business visualization techniques and data navigation options, we propose a comprehensive visual exploration framework, which incorporates the awareness of the conceptual extensions into the frontend in form of enriched metadata and a powerful data interface for interactive specification of ad hoc queries. Finally, we propose a class of multidimensional visualization techniques for advanced visual data analysis, called *Enhanced Decomposition Trees*, which map a series of decomposition (i.e., drill-down) steps to a hierarchy of obtained subaggregates.

To the best of our knowledge, the proposed modeling framework is superior to other approaches to multidimensional modeling proposed prior or in the course of carrying out this research. Its main advantage is the coherence of the formalization, the accompanying graphical notation, the relational and the metadata mapping, and the frontend implementation. However, we do not claim to have found an ultimate solution to providing OLAP support to all kinds of complex data and applications and realize that such a goal is simply unattainable within a single PhD work.

Zusammenfassung

UMFASSENDE DATENANALYSE ist in einer Vielzahl von Anwendungen unentbehrlich geworden. Data Warehousing und OLAP entstanden in den 90er Jahren als eine Antwort auf diesen Bedarf im betriebswirtschaftlichen Bereich und bewiesen ihre nahezu universale Einsetzbarkeit zur Entscheidungsunterstützung. Das letzte Jahrzehnt bezeugte die Verbreitung von Data Warehouses über die klassischen betriebswirtschaftlichen Anwendungen hinaus und die Erschließung neuer Einsatzgebiete, zum Beispiel in der Verwaltung, Wissenschaft und Forschung, Medizin, Webanalyse, Netzwerksicherheit, usw. Obwohl viele Forschungsbereiche des Data Warehousing und OLAP den Status der Reife erreicht haben, entstehen neue Herausforderungen durch die Anwendung der traditionellen Technologie in derartigen unkonventionellen Kontexten. Die genauen Ursachen für unbefriedigende Leistungen sind vielfältig und reichen von der Starrheit des konzeptuellen Datenmodells, des angepassten logischen Schemas und der Implementierungsstrategie bis hin zu verfügbaren Datentransformationstechniken, vorherrschenden Standards der Metadatenmodellierung, der eingeschränkten Menge an unterstützten Operatoren und Problemen der Endbenutzeroberflächen.

Ziel dieser Doktorarbeit ist es, die Funktionalität des weit verbreiteten OLAP-Rahmenwerks fundamental zu erweitern, sowie die vom Standard bisher nicht unterstützte Anwendungsszenarien adäquat zu bewerkstelligen. Im Detail erforschen wir die Beschränkungen der State-of-the-Art relationalen OLAP-Systeme, analysieren Anforderungen für umfassende Datenanalysen, führen entsprechende Erweiterungen auf der konzeptionellen, logischen und Metadaten-Ebene ein und demonstrieren wie das erweiterte Modell sowohl in der Backend- als auch in der Frontend-Schicht umgesetzt werden kann. Um die Herausforderungen bei der Modellierung exemplarisch aufzuzeigen, werden dazu praxisnahe Fallstudien aus den Bereichen der akademischen Verwaltung und Medizintechnik verwandt. Zwei komplementäre Ideen bestimmen den Beitrag dieser Arbeit: *i)* die Erweiterung des multidimensionalen Datenmodells, welches die Grundlage von OLAP repräsentiert, und *ii)* die Bereicherung der Frontend-Schicht durch innovative Funktionalitäten für visuelle Analyse großer und komplexer Datenbestände.

Der Schwerpunkt liegt in der Erweiterung der Backend-Funktionalität von Data Warehousing. Diese wird durch die Überprüfung der Anforderungen neuer Anwendungsdomänen, die Identifizierung von Problem-bereichen in der konventionellen OLAP Technologie hinsichtlich dieser Anforderungen und die Suche nach angemessenen Lösungen auf der konzeptionellen Ebene erreicht. Da der letztendliche Wert des resultierenden erweiterten konzeptuellen Datenmodells durch seine Implementierbarkeit in einem logischen Schema bestimmt ist, schlagen wir eine Menge von entsprechenden Umsetzungs- und Transformationstechniken zur Überführung des konzeptuellen Schema ins logische vor, um die korrekte Summierbarkeit sicherzustellen. Der Kernbeitrag dieses Teils der Arbeit ist das erweiterte konzeptuelle Modell, welches in Form von formalen Definitionen und einer graphischen Notation *X*-DFM (Extended Dimensional Fact Model) ausgedrückt wird. *X*-DFM besteht sowohl aus einer Menge von visuellen Modellierungskonstrukten als auch aus einer Menge von Richtlinien zum Entwurf multidimensionaler Schemata, welche sich im Einklang mit der definierten formalen Semantik befinden. Sowohl das formale als auch das graphische Modell stellen drei Abstraktionsebenen bereit – die untere, die mittlere, und die obere Ebene – um einen stufenweisen Entwurf

des konzeptuellen Schemas zu unterstützen.

Die wesentlichen Merkmale des erweiterten Modells sind die Vereinheitlichung des multidimensionalen Raumes als Grundlage zur Handhabung semantisch verwandter Elemente, Aggregierbarkeits- und der Additivitätsbedingungen von Kennzahlen, viele-zu-viele Beziehungen zwischen Fakten und Dimensionen und zwischen den Hierarchiestufen einer Dimension, Faktschemata ohne Kennzahlen, Heterogenität von Fakten und Dimensionshierarchien, degenerierte Fakten und Dimensionen, optional und partiell verwandte “Roll-up”-Beziehungen und verschiedene Arten von Irregularitäten in Dimensionshierarchien sowohl auf der Schema- als auch auf der Instanzebene. Außerdem unterstützt das Modell objektorientierte Eigenschaften wie Generalisierung und Spezialisierung in Fakten und Dimensionen, multiple Rollen eines Elements, Assoziationen und Selbst-Assoziationen von Faktschemata, sowie die Modellierung von abstrakten, abgeleiteten und abhängigen Elementen.

Die Umsetzbarkeit des vorgeschlagenen konzeptuellen Modells wird durch die Darstellung dessen demonstriert, wie es auf eine relationale Repräsentation mithilfe von generellen Richtlinien des logischen Design abgebildet werden kann. Dabei wird die gängige Summierbarkeitsbedingung zur Bestimmung der notwendigen Schema- und/oder Instanztransformationen verwendet. Des Weiteren betrachten wir das Problem der Metadatenmodellierung. Metadaten spielen in Datawarehouse-Systemen eine prominente Rolle, da sie als Vermittler zwischen verschiedenen Schichten und Komponenten der Systemarchitektur fungieren. Insbesondere beschäftigen wir uns mit den Metadaten der Anwendungsschicht, da diese für die Erfassung der multidimensionalen Semantik hinter den “flachen” Datenbanktabellen verantwortlich ist.

Der andere Teil unserer Forschung konzentriert sich auf Funktionalitäten der Endbenutzerschicht. Auf OLAP-Daten wird typischerweise ausschliesslich durch visuelle Analysewerkzeuge zugegriffen, wobei die Benutzer bequem zu den relevanten Datenmengen navigieren und diese mittels verschiedenartiger visueller Layouts und Interaktionstechniken erforschen können. Daher hängt die gesamte Leistungsfähigkeit der Analyse letztendlich von den Funktionalitäten der angebotenen Frontend-Werkzeugen ab. Da wir uns den Defiziten von gängigen Business-Visualisierungstechniken und Ansätzen zur Datennavigation bewusst sind, schlagen wir ein umfassendes visuelles Explorationsrahmenwerk vor, welches die konzeptuellen Erweiterungen in Form von angereicherten Metadaten und einer mächtigen Datenschnittstelle zu interaktiven Spezifikation von Ad-hoc-Anfragen in die Frontend-Schicht propagiert. Abschließend stellen wir eine Klasse von multidimensionalen Visualisierungstechniken namens *Enhanced Decomposition Trees* zur fortgeschrittenen visuellen Datenanalyse vor, welche eine Serie von Zerlegungsschritten (d.h., Drill-down) auf eine Hierarchie von erhaltenen Teilaggregaten abbildet.

Nach bestem Wissen und Gewissen ist das vorgestellte Modellierungsframework allen zuvor oder während der Durchführung dieser Forschung vorgeschlagenen Ansätzen zur multidimensionalen Modellierung überlegen. Seine Hauptvorteile bestehen in der Kohärenz der Formalisierung, der begleitenden graphischen Notation, der relationalen und Metadaten- Abbildungstechniken und der Frontend-Implementierung. Dennoch erheben wir nicht den Anspruch, die endgültige Lösung für die Unterstützung aller Arten von komplexen Daten und Anwendungen durch die OLAP-Technologie gefunden zu haben und stellen fest, dass ein derartiges Ziel innerhalb einer einzigen Promotion nicht erreichbar ist.

Contents

Acknowledgments	iii
Abstract	vii
Zusammenfassung	ix
1 Introduction	1
1.1 Data Warehousing and OLAP	1
1.2 Motivation	2
1.3 Aims and Contributions of the Thesis	4
1.4 Thesis Outline	5
2 Background and Related Work	9
2.1 Business Intelligence and its Components	9
2.1.1 Business Intelligence	9
2.1.2 Data Warehousing	11
2.2 OLAP and the Multidimensional Data Model	14
2.2.1 Elements of the Multidimensional Data Model	14
2.2.2 OLAP Operations	17
2.2.3 OLAP Implementation Alternatives	20
2.2.4 Data Warehouse Design Methodology	24
2.3 Visual Analysis and Exploration	29
2.3.1 Visual OLAP as an Emerging Trend	30
2.3.2 Visual Exploration Framework	31
3 Extending the Multidimensional Data Model	35
3.1 State of the Art of Multidimensional Modeling	35
3.1.1 Fundamental Constraints of the Multidimensional Data Model	36
3.1.2 Related Work	37
3.2 Requirements of Comprehensive Multidimensional Analysis	38
3.3 Conceptual Model: Graphical Notation and Formalization	41
3.3.1 X -DFM as the Graphical Modeling Notation	41
3.3.2 Formalization	47
3.4 Terminology and Definitions	48
3.4.1 Unified Multidimensional Space	48
3.4.2 Facts and Dimensions	49

4	Dimensions and Hierarchies in the Multidimensional Data Model	59
4.1	State of the Art of Dimensional Modeling	59
4.1.1	Rigidity of OLAP Dimensions	60
4.1.2	Related Work on Modeling Dimension Hierarchies	60
4.2	Academic Management as the Motivating Case Study	62
4.3	Categorization of Dimension and Hierarchy Types	66
4.3.1	Refining the Formal Framework	66
4.3.2	Dimension Types	68
4.4	Classification of Hierarchy Types	70
4.4.1	Strict vs. Non-Strict Hierarchies	71
4.4.2	Types of Homogeneous Hierarchies	73
4.4.3	Types of Heterogeneous Hierarchies	75
4.5	Classification of Multiple Hierarchies	84
5	Measures, Facts, and Galaxies in the Multidimensional Data Model	89
5.1	Surgical Workflow Analysis as a Motivating Case Study	90
5.1.1	Requirements of Surgical Workflow Analysis	90
5.1.2	Structuring Surgical Workflows	92
5.2	Categorization of Facts and Measures	94
5.2.1	Measurable Facts and Measure Types	95
5.2.2	Non-Measurable Facts	98
5.3	Types of Multi-Fact Schemes	99
5.3.1	Fact Degeneration	100
5.3.2	Fact Roll-up	101
5.3.3	Fact Generalization	101
5.3.4	“Fading” Duality of Fact and Dimension Roles	103
5.4	Classification of Dimension Sharing Patterns	105
5.4.1	Dimension Sharing Modes in <i>X</i> -DFM	105
5.4.2	Levels and Types of Dimension Sharing	106
6	Data Warehouse Design for Non-Conventional Applications	111
6.1	Challenges of Conceptual Data Warehouse Design	111
6.1.1	Standard Stages of Conceptual Data Warehouse Design	112
6.1.2	Limitations of Conventional Design Methodologies	112
6.2	Acquisition of Multidimensional Schemes from the E/R Schemes	113
6.2.1	Verification and Refinement of the E/R Scheme	114
6.2.2	Identifying Facts and Dimensions	118
6.3	Evaluation of the Proposed Framework	123
6.3.1	Usage Scenario 1: Discectomy Surgery	124
6.3.2	Usage Scenario 2: Functional Endoscopic Sinus Surgery	127
7	Relational Implementation of the Multidimensional Data Model	131
7.1	Mapping the Multidimensional Model to the Relational Model	132
7.1.1	Mapping Fact Schemes	133
7.1.2	Handling Derived Elements	134
7.1.3	Mapping Dimension Hierarchies	135
7.2	Mapping Heterogeneous Hierarchy Schemes	136

7.2.1	Mapping Non-Covering Hierarchies	136
7.2.2	Mapping Generalized Hierarchies	140
7.3	Enforcing Summarizability in Homogeneous Hierarchies	148
7.3.1	Mapping to Covering	149
7.3.2	Mapping to Strict	150
7.3.3	Mapping to Onto	153
7.4	Metadata for the Analysis Layer	154
7.4.1	Overview of the CWM	155
7.4.2	Representing multidimensional properties in the CWM	162
8	Interactive Exploration of OLAP Aggregates	167
8.1	Visual Analysis Framework	168
8.1.1	Related Work on Visualization for OLAP	168
8.1.2	Components of a Visual OLAP Tool	172
8.2	Navigating in Multidimensional Data	176
8.2.1	Prevailing Data Navigation Schemes	176
8.2.2	Enhancing Data Navigation through Scheme Awareness	178
8.2.3	Dynamic Properties of the Navigation Scheme	182
8.2.4	OLAP Operators and their Implementation Options	186
8.3	Hierarchical Visualization Techniques for OLAP	188
8.3.1	Salient Characteristics of Visual Interaction Patterns	188
8.3.2	Decomposition Tree	188
8.3.3	Enhanced Decomposition Trees	191
8.3.4	Spatio-temporal Visualization Techniques	199
9	Thesis conclusions and Future Work	201
9.1	Summary of Supported Multidimensional Properties	201
9.2	Conclusions	204
9.3	Future Work	208
	Bibliography	209
	Index	221

List of Tables

2.1	Data warehouse characteristics according to W. H. Inmon	11
2.2	ROLAP versus MOLAP	21
2.3	Diagram elements of the UML Profile for Database Design	29
3.1	Overview of the desirable multidimensional properties	40
3.2	Graphical constructs of the original DFM notation	42
3.3	Graphical node type constructs of <i>X</i> -DFM	44
3.4	Graphical edge type constructs of <i>X</i> -DFM	46
4.1	Example of a non-structured hierarchy storage	65
6.1	Statistical overview of the acquired data	128
7.1	Representing multidimensional properties in the CWM OLAP metamodel	163
7.2	Representing multidimensional properties in the CWM metamodels other than OLAP	164
8.1	OLAP operations and their implementation in a visual frontend	187
9.1	Multidimensional properties for mapping semantically related elements	202
9.2	Multidimensional properties related to measures and facts	202
9.3	Multidimensional properties related to dimensions and hierarchies	203
9.4	Multidimensional properties related to categories and roll-up relationships	203
9.5	Dynamic multidimensional properties	204

List of Figures

1.1	An overall data warehouse system architecture	3
1.2	Thesis outline with respect to the multi-layer data warehouse reference model	7
2.1	Definitional levels of Business Intelligence	10
2.2	A multi-layer data warehousing system architecture	12
2.3	A sample 3-dimensional cube (fragment) storing student enrollment numbers	15
2.4	Dimension Degree with multiple hierarchies: scheme and instances	16
2.5	DRILL-DOWN and ROLL-UP across dimensions (top) and within a dimension hierarchy (bottom)	18
2.6	Example of a SLICE&DICE operation	19
2.7	Example of a PIVOT operation with two rotations (shaded cells are visible in both views)	20
2.8	Example of a DRILL-ACROSS operation with two input cubes	20
2.9	A sample star schema	22
2.10	A sample snowflake schema	23
2.11	A sample galaxy schema with normalized dimension tables	23
2.12	Fact constellation schema	24
2.13	Star cluster schema	24
2.14	A pair of compatible fact schemes and their overlap modeled in DFM	27
2.15	Exploring a multidimensional data cube with a pivot table	30
2.16	Trends in data visualization for business analysis	31
2.17	Data exploration cycle	32
2.18	Tableau Software as a powerful frontend for visual analysis	33
2.19	Examples of sophisticated visualizations generated by VizQL statements	34
3.1	A 5-dimensional fact scheme PURCHASE in the original DFM notation	43
3.2	The revised fact scheme PURCHASE in the X -DFM notation	47
3.3	Modeling purchasing facts at higher levels of abstraction	48
3.4	Examples of dimension sharing in star and multi-star schemes	50
3.5	Arranging facts into a cluster in X -DFM	51
3.6	A 3-dimensional space produced by dimensions X , Y , and Z	52
3.7	The UL construct set of X -DFM	52
3.8	Dimension schemes as directed graphs of various complexity	54
3.9	Clustered IL view of purchasing facts with compatible and conform categories	57
3.10	Alternatives of dealing with property attributes in the LL model	58
4.1	SuperX fact table COB_BUSA_CUBE (university expenditures) and its dimension tables	63
4.2	SuperX fact table STUD_ALLG_CUBE (student enrollments) and its dimension tables	64
4.3	Organization hierarchy (fragment) of a university	64
4.4	Fact scheme PURCHASE with added complexity	66
4.5	Categorization of dimension and hierarchy types	67

4.6	Dimension time as a merge graph of its hierarchies	68
4.7	Categorization of dimension types	69
4.8	Examples of “shadow” dimensions	69
4.9	Categorization of hierarchy types with respect to strictness	71
4.10	A sample instance of a non-strict hierarchy	72
4.11	Normalizing non-strictness via a weighted hierarchy	73
4.12	Categorization of homogeneous hierarchy types	73
4.13	Examples of a symmetric and an asymmetric hierarchy	74
4.14	Categorization of heterogeneous hierarchy types	75
4.15	Project locations as a non-covering hierarchy	76
4.16	A non-covering hierarchy transformed into a generalized hierarchy	77
4.17	Examples of inheritance hierarchies in purchaser dimension	78
4.18	An inheritance hierarchy with local root categories added	78
4.19	An inheritance hierarchy with aggregation hierarchies added	79
4.20	A generalized hierarchy scheme with a common bottom level	79
4.21	A completed generalized hierarchy scheme	80
4.22	Staff hierarchy with incomplete specialization	81
4.23	Handling incomplete specialization hierarchies in <i>X</i> -DFM	81
4.24	Person hierarchy with overlapping specialization	82
4.25	Revealing mixed grain by eliminating redundant scheme fragments	83
4.26	Categorization of multiple hierarchies	84
4.27	Sample instances of parallel hierarchies	85
4.28	Examples of combining parallel hierarchies into new aggregation hierarchies	85
4.29	Parallel hierarchies with fictitious convergence in city and country	86
4.30	Examples of multiple hierarchies	86
4.31	Adding semi-annual and quarter as aggregation levels in time hierarchy	87
5.1	A sample 3-dimensional cube (fragment) storing surgical instrument usage statistics	91
5.2	Vertical (de-)composition of a surgical process	92
5.3	Recording scheme of a surgical process model as a UML class diagram	93
5.4	Recording scheme of a surgical process model as an E/R diagram	93
5.5	Example of multidimensional modeling of surgery data using <i>X</i> -DFM	94
5.6	Categorization of fact and multi-fact schemes	95
5.7	Categorization of measure types	96
5.8	Hospitalization records as transactions, periodic and accumulating snapshots	98
5.9	Examples of non-measurable fact schemes	99
5.10	Examples of degenerate fact schemes	101
5.11	Examples of hierarchical relationships between fact schemes	102
5.12	Fact generalization of classes EVENT and ACTIVITY as a superclass COMPONENT	102
5.13	Fact SURGERY as a dimension in its satellite fact SURGERY-PARTICIPANT	104
5.14	Example of a PUSH operation	104
5.15	Fact scheme SURGERY modeled using different dimension sharing modes in <i>X</i> -DFM	105
5.16	Categorization of dimension sharing patterns	107
5.17	Examples of dimension conformance within and across fact schemes	107
5.18	Examples of dimension inclusion within and across fact schemes	108
5.19	A recursive “fact-as-dimension” inclusion in a series of fact roll-ups	109
6.1	Examples of presenting complex attributes and re-modeling multivalued attributes	116
6.2	Transforming composite attributes into related entity types	116

6.3	Transforming attributes into entity types to reveal implied roll-up relationships between them	117
6.4	Adding specialization to the heterogeneous entity type SYSTEM	117
6.5	Transforming entity type STEP into a fact scheme	119
6.6	Transforming entity type ACTIVITY (left) into a fact scheme	119
6.7	Transforming $m:n$ and recursive relationships of COMPONENT into degenerate facts	121
6.8	Fragment of the E/R scheme relevant for building phase dimension of COMPONENT	122
6.9	Multiple alternative and parallel hierarchies in DATE dimension	123
6.10	The resulting dimension scheme of the PHASE dimension in COMPONENT	123
6.11	Multidimensional scheme of a surgical workflow structure	124
6.12	Annotated diagram of vertebra as an anatomic structure affected by a discectomy intervention	125
6.13	Spine cross-section view	125
6.14	Instrument usage statistics as a pivot table	126
6.15	Occurrence and duration of bone ablation steps in discectomy interventions	127
6.16	Synchronizing multiple workflow versions of the same surgical intervention	128
6.17	Pivot table view of the instrument usage statistics	129
7.1	Example of conceptual fact schemes and their logical representations	133
7.2	Derived fact modeled as a view linked to the base fact tables	135
7.3	A non-covering hierarchy of project locations: instance and scheme	137
7.4	State of the non-covering hierarchy with city normalized to covering	137
7.5	Mapping to covering by normalizing non-covering levels building and office	138
7.6	Transformation of a non-covering hierarchy into a set of multiple alternative hierarchies	139
7.7	Logical schema of project location hierarchies	139
7.8	Transforming overlapping specialization into a disjoint one	141
7.9	Adding alternative generalization layers for grouping disjoint subclasses	142
7.10	Generalized hierarchy person after eliminating overlapping specialization	142
7.11	Resolving mixed grain into self-specializations	143
7.12	Adding roll-up relationships to a self-specialization scheme	144
7.13	A completed generalized hierarchy scheme purchaser	144
7.14	Logical schema of a generalized hierarchy purchaser	145
7.15	Example of obtaining the instance of the generalized relation PURCHASER	146
7.16	Resolving mixed grain into multiple dimension tables	147
7.17	Optimized logical schema of dimension tables with mixed grain	148
7.18	A sample hierarchy containing non-onto, non-covering and non-strict elements	149
7.19	The state of the hierarchy after mapping to covering	150
7.20	The state of the hierarchy after eliminating multi-parent roll-ups	150
7.21	The state of the hierarchy after adding categories with “fused” elements	151
7.22	The state of the sample hierarchy after unlinking non-strict roll-up relationships	152
7.23	Resolving multi-parent relationships by assigning weights to roll-up edges	153
7.24	The state of the hierarchy after ad hoc edge elimination	153
7.25	An inherently asymmetric department hierarchy	154
7.26	The state of the department hierarchy after mapping to covering, strict, and onto	154
7.27	The multi-layered metamodel of the CWM and its constituent packages	155
7.28	OLAP Metamodel of the CWM: Major Classes and Associations	156
7.29	OLAP Metamodel of the CWM: Dimension and Hierarchy	157
7.30	OLAP Metamodel of the CWM: Inheritance from the Object Model	158
7.31	Transformation Metamodel (fragment) of the CWM	159
7.32	Deployment mapping structures in the CWM OLAP Metamodel	160

7.33	A sample XML file for metadata export according to the CWM	161
7.34	Description of a computed measure at the Analysis layer (CWM OLAP)	165
7.35	Description of a computed measure at the Resource layer (CWM Relational)	166
8.1	Segmentation of visualization techniques for OLAP by layout and granularity	170
8.2	Mapping data fields to a visual layout in Tableau	173
8.3	Elementary perceptual tasks in visual data analysis	175
8.4	Ranking of perceptual tasks with respect to the data type	175
8.5	Instance-based hierarchy navigation in Cognos PowerPlay	177
8.6	Explicit enumeration of hierarchy levels in OracleBI Discoverer	177
8.7	Treating hierarchy levels as dimensions in MicroStrategy OLAP Services	178
8.8	Instance-based vs. scheme-based navigation for a hierarchical dimension	179
8.9	Obtaining a scheme-based navigation hierarchy of a dimension	179
8.10	Examples of navigating in multiple hierarchies	180
8.11	Different approaches to representing measures	181
8.12	Performing a PUSH operation	183
8.13	Example of a combined usage of PUSH and PULL	184
8.14	“Galaxy” view of the navigation for a pair of related cubes	185
8.15	Multicube navigation scheme with enhanced semantics	186
8.16	Mapping a sequence of decomposition steps to a set of bar-charts	189
8.17	Arranging a sequence of decomposition steps into a decomposition tree	190
8.18	Commercial decomposition tree techniques	190
8.19	Using different decomposition modes with a bar-chart tree	192
8.20	A decomposition tree produced via a nested, an outer, and an inner split	193
8.21	A decomposition tree with an orthogonal outer split (from faculty to cost class)	193
8.22	Exploring a drill-across measure costs-per-student ratio with a plain-text decomposition tree	194
8.23	A rectangular space-filling decomposition tree (TreeMap)	195
8.24	A radial space-filling decomposition tree (SolarPlot)	196
8.25	Hierarchical HeatMaps for non-cumulative disaggregation	197
8.26	A decomposition tree with space-filling bars	198
8.27	A decomposition tree with area-preserving bars	198
8.28	Multiscale <i>Recursive Pattern</i> visualization for drilling down along the time dimension	199
8.29	<i>Hierarchical Network Map</i> with geographical decomposition of the network traffic load	200

Chapter 1

Introduction

THIS CHAPTER INTRODUCES the general context, the aims, and the rationale of the thesis with a brief description of each chapter's contents. The motivation for employing the data warehousing technology in non-conventional application domains is given, followed by identifying the limitations of the standard techniques to satisfy the requirements of novel usage scenarios and complex data.

Contents

1.1	Data Warehousing and OLAP	1
1.2	Motivation	2
1.3	Aims and Contributions of the Thesis	4
1.4	Thesis Outline	5

1.1 Data Warehousing and OLAP

In the 90s and beyond, a key to survival in the business world has been the ability to analyze, plan and react to changing business conditions in a much more rapid fashion. In the same decade, a set of significant new concepts and tools have evolved into a new technology that makes it possible to attack the problem of providing all the key people in the enterprise with access to whatever level of information they need for decision making [138]. The terms that have come to characterize this technology are *data warehousing* and *OLAP* (*On-line Analytical Processing*) [30]. The applicability of the data warehousing approach is by no means restricted to business scenarios. As comprehensive data analysis is becoming indispensable in a variety of real-world applications, the deployment of data warehouses has reached out for non-business domains, such as government, science, education, research, medicine, to name the prominent ones.

The core feature of data warehousing is to provide a separate database that integrates the data extracted from various operative systems and external sources and rearranges it into multidimensional views to enable simple but powerful aggregation. The analysis is preceded by a highly complex ETL (extract, transform, load) process aimed at integrating the data and bringing it into a consistent state. The OLAP technology draws its analytical power from the underlying *multidimensional data model*. This model categorizes data as *measurable facts*, or simply *measures*), which are typically of numeric type. Measures are determined by a set of descriptive *dimensions*, which serve as exploration axes for aggregation. Member values of a

dimension may be further organized in a containment type hierarchy to support additional aggregation levels. The resulting multidimensional data structures are often referred to as *OLAP cubes*.

The task of data warehouse design is to model and to implement a database for a particular analytical application. The modeling task consists in mapping the relevant data of a given application domain into the data model of the data warehouse system. The design typically undergoes three phases: *i*) conceptual modeling of multidimensional cubes, *ii*) logical modeling as the basis for schema implementation, and *iii*) physical design addressing the actual implementation issues, such as indexing, partitioning, view materialization, etc.

1.2 Motivation

Even though data warehousing is an established and widely adopted practice in the modern information technology platform, there exist numerous open research issues in this area. Many of those issues emanate from the attempts to apply the business performance oriented OLAP techniques to non-conventional application domains. The causes of deficiencies and failures are manifold, from the underlying conceptual model to the frontend analysis tools. However, it is our conviction that despite any limitations the data warehousing technology has the potential to provide adequate analysis support to a wide spectrum of usage scenarios. The claimed universality of data warehousing bears on the concept of generic “analyzability”: the data should be homogenized, integrated and preprocessed to enable efficient and goal-oriented analysis [8]. The need for this kind of analysis is encountered virtually in any application domain where large data volumes get accumulated over time.

Having the data available in the form of multidimensional cubes in a data warehouse is not only a prerequisite for OLAP, but is also beneficial for applying *data mining* techniques. These techniques are aimed at discovering new information from vast amounts of data in terms of rules or patterns, which cannot be found by merely querying the data [41]. Used in conjunction with one another, data warehousing and data mining achieve a synergetic effect: while computationally expensive data mining algorithms display better performance and provide more accurate results when run on consolidated and aggregated data sets, the multidimensional data itself can be enriched by capturing the output generated by the data mining techniques as additional measures and classification hierarchies.

Figure 1.1 shows a simplified view of the data warehouse reference architecture comprising three basic layers – transformation, application, and presentation [89]. Some authors propose to refine this standard technical architecture into four or even five layers. Schütte et al. [161] suggest to separate the back-end architecture into two layers: the operational data sources and the import, or ETL, layer. Further, Propach and Reuse [151] obtain the fifth layer by subdividing the presentation layer into the analysis, encompassing data mining and OLAP, and the actual presentation, encompassing only frontend interfaces. In such multilayer environments, the overall power of the entire system depends on the maturity of each component and their interplay: the constraints of the data model at the back-end propagate themselves to the upper layers, while the limitations of the end-user tools may hinder full utilization of the application layer’s functionality.

At the conceptual level, the rigidity of the conventional data warehouse design is caused primarily by the enforcement of summarizability for all dimension hierarchies. The concept of summarizability, introduced by Rafanelli and Shoshani [152] in the context of statistical databases, ensures correct aggregation by requiring dimension hierarchies to be strict and balanced [98]. However, hierarchies in many real-world applications are not summarizable and their transformation may be undesirable in order to preserve the original hierarchical relationships. Another critical constraint is that of homogeneity. Each element of a specific fact type or a dimension is required to fully conform to the respective schema and roll-up along the same paths as all other entries of the same class. Prohibition of NULL values in facts and dimensions ensures reliable aggregate computation but results in the inability to cope with contingent uncertainty and imprecision in the input data.

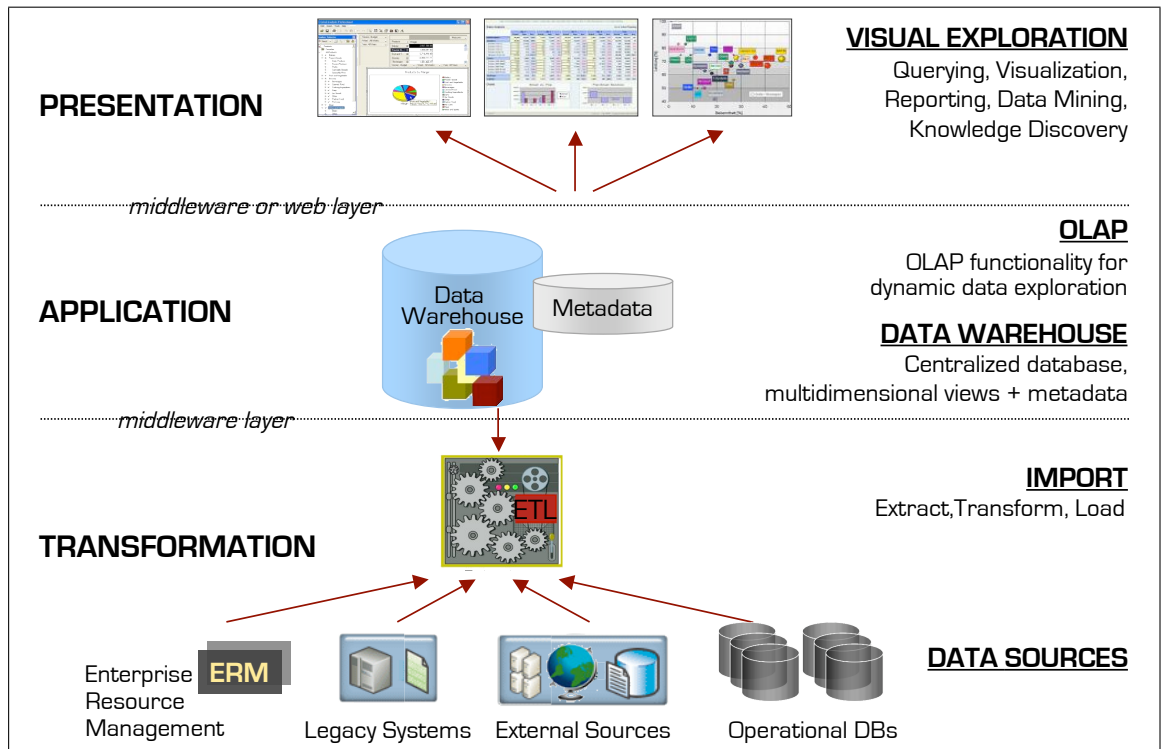


Figure 1.1: An overall data warehouse system architecture

In a multitude of applications, analysts are confronted with irregular and complex data that violates the constraints imposed by the standard multidimensional data model. In a survey on open issues in multidimensional modeling [63], Hümmer et al. identified unbalanced and irregular dimension hierarchies as one of the major data warehouse design challenges for both researchers and practitioners.

Even more problems arise when attempting to warehouse non-conventional data, such as events, processes, workflows, and streams. The original input data, supplied by the logs of process executions, protocols or other sources, even if reshaped into a multidimensional view, might not contain explicit quantitative metrics to serve as measurable facts. In business process analysis, measures of interest are often specified ad hoc at runtime and not a priori during the design phase.

Many of the above problems arise at the very root of the data warehouse system architecture, i.e., they are due to the rigidity of the conceptual model itself and, therefore, should be addressed by extending the multidimensional model at the conceptual level. However, the value of any semantic extension is determined by the ability to map it to the logical layer of the data warehouse, and, subsequently, to the end-user OLAP tools. At present, there appears to be a discrepancy between advanced conceptual data models and their implementations in state-of-the-art data warehouse systems. Obviously, this gap needs to be bridged in order to propagate the benefits of the extended models up to the presentation layer, i.e., to the end-user interfaces.

To overcome the restrictions mentioned above and thus to increase the capacity of the OLAP technology to handle a broader spectrum of practical situations, analysis tools have to be extended at virtually all levels of the system architecture:

- ◆ recognition and classification of complex data structures,
- ◆ conceptual and logical model extensions,
- ◆ data and schema normalization techniques,
- ◆ enhanced metadata model to ensure correct querying and aggregation,
- ◆ lossless mapping of cube schemes to a visual navigation,
- ◆ adequate visualization techniques for presenting complex query results.

The above enumeration of challenges is by far not exhaustive and should only convey a general idea about the complexity of the defined task.

1.3 Aims and Contributions of the Thesis

The title of this thesis reflects the overall goal of this work, which is to extend the capacity of the OLAP technology to adequately handle complex data and non-conventional applications. Although many research areas in the field of data warehousing and OLAP have reached the state of maturity, new challenges arise when applying the established technology to novel usage scenarios. In this thesis, we focus on overcoming the deficiencies of the conventional OLAP technology in providing adequate decision support to non-standard applications. Real-world case studies from the domains of academic administration and medical engineering are used to exemplify the challenges and motivate the proposed solutions. Two complimentary ideas determine the contribution of this work:

- a) to extend the multidimensional data model that represents the very core of OLAP and
- b) to enrich the frontend layer with innovative features for visual analysis of large data volumes.

The primary focus of the work is on extending the backend functionality of data warehousing, obtained by inspecting the requirements of novel application domains, identifying the bottle-necks of the conventional OLAP technology with respect to those requirements, and searching for the ways to overcome the identified limitations at the conceptual level. Since the ultimate worth of the resulting extended conceptual data model is determined by its implementability in a state-of-the-art data warehouse system, we propose a set of corresponding conceptual-to-logical mappings and transformation techniques for restoring summarizability and correct aggregate navigation and obtaining a logical data model.

The other part of this work concentrates on the frontend issues. Since end-users typically interact with OLAP data using exclusively visual tools, which allow them to conveniently navigate to the subset of interest and explore it using various visual layouts and interaction options, the capability of the analysis is eventually limited to the functionality of the frontend tools. Aware of the deficiencies of standard business visualization techniques and data navigation options, we incorporate the awareness of the backend extensions into the end-user interface in the form of enriched metadata, adjust the data navigation framework for interactive query specification to account for the extended semantics, and, finally, propose a set of advanced hierarchical visualization techniques for exploring multidimensional data.

We describe a comprehensive framework for extending the data warehouse technology at the database, application, and presentation layer. Specifically, at the backend level, the following is proposed:

- ◆ a categorization of fact, dimension and hierarchy types,
- ◆ a categorization inter-fact and fact-dimensional relationships in the unified multidimensional space,
- ◆ a formal definition of the extended conceptual multidimensional data model,
- ◆ a graphical notation *X*-DFM (Extended Dimensional Fact Model) compliant with the formal model,
- ◆ a conceptual-to-relational mapping of the extended multidimensional model with scheme-level and instance-level data transformation techniques for ensuring summarizability.

At the application level, we adjust the metadata model to reflect the conceptual changes. Frontend enhancements comprise a powerful navigation approach to interactive specification of ad hoc queries and a class of multidimensional visualization techniques, denoted *Enhanced Decomposition Tree*, for advanced visual data analysis. The proposed techniques definitely do not exhaust the challenges of comprehensive analysis and supporting non-conventional applications, but we expect them to be useful for a wide range of usage scenarios.

1.4 Thesis Outline

To address the specified aims this thesis is outlined as follows:

- ◆ Chapter 2 provides the necessary background of the concepts relevant in the context of our research. Section 2.1 sets the stage by giving an overview of the area of Business Intelligence as a whole and data warehousing as its major component, sketching recent achievements and trends in the field. Section 2.2 is dedicated to OLAP fundamentals, such as the multidimensional data model, OLAP operations, implementation alternatives, and the data warehouse design methodology. In the concluding Section 2.3 we elaborate on the issues of visual exploration of OLAP cubes, state-of-the-art analysis tools, prevailing practices and frameworks, and recent research findings in the area of visual data analysis and decision support.
- ◆ Chapter 3 presents the overall framework of the proposed extended multidimensional data model. Section 3.1 reviews the state of the art in the field of conceptual data warehouse design, followed by formulating the requirements of comprehensive multidimensional data analysis in Section 3.2. Section 3.3 introduces the conceptual design methodology composed of the formal model and the accompanying graphical notation *X-DFM*, both defined at three levels of semantic abstraction: a lower, an intermediate, and an upper one. Section 3.4 proceeds with the actual formalization of the fundamental elements of the conceptual model, and defines a valid subset of *X-DFM* for modeling the presented elements at each of the three abstraction layers.

The fundamental elements of the multidimensional model presented in Chapter 3 are further investigated, refined and categorized in Chapters 4 and 5:

- ◆ Chapter 4 extends the multidimensional data model with respect to supporting complex dimensions and hierarchy types. State-of-the art research on dimensional modeling and the prevailing modeling constraints are described in Section 4.1. Section 4.2 presents a motivating case study from the domain of academic management. A systematic categorization of OLAP dimension and hierarchy types in the form of a metamodel is undertaken in Section 4.3, followed by an in-depth categorization of hierarchy types in Section 4.4 and that of multiple hierarchies within the same dimension in Section 4.5.
- ◆ Chapter 5 undertakes a similar extension and classification effort for the multidimensional elements of type facts, measures, and multi-fact schemes (galaxies). In Section 5.1 we present a case study from the area of Surgical Workflow Analysis as an example of an operational data warehousing application to exemplify the challenges of modeling complex fact schemes. Section 5.2 is dedicated to the categorization of fact and measure types according to their aggregation semantics. Inter-factual relationships and the resulting types of galaxy schemes are presented in Section 5.3. Finally, Section 5.4 investigates dimension sharing patterns occurring in the unified multidimensional space and formulates guidelines for handling dimension sharing in *X-DFM*.

Chapters 6 and 7 focus on the challenges of designing, engineering, and implementing a data warehouse for non-conventional applications:

- ◆ In Chapter 6 we propose a methodology for semi-automatic engineering of multidimensional models from the existing conceptual schemes of the underlying operational data sources. Section 6.1 overviews standard approaches to the conceptual data warehouse and shows their limitations when handling non-conventional domains. In Section 6.2 we propose an alternative approach to engineering multidimensional models via a cardinality-based transformation of the existing operational models. Section 6.3 demonstrates the feasibility of the proposed modeling framework through two usage scenarios and a series of sample analysis tasks from the field of surgical workflows.
- ◆ Chapter 7 describes a relational implementation of the extended multidimensional model, presented in Chapters 3 through 5. Section 7.1 describes the fundamentals of obtaining a logical scheme from a conceptual one, subdivided into the guidelines for implementing fact schemes, derived elements, and dimension hierarchies. In Sections 7.2 and 7.3, we present a two-phase transformation approach to obtaining a relational mapping of non-summarizable dimension hierarchies. In the first phase (Section 7.2), scheme transformation techniques are employed to normalize heterogeneous hierarchy schemes, e.g., to eliminate overlapping specialization and mixed granularity. In the second phase (Section 7.3), instance normalization techniques are applied to eliminate irregularity in homogeneous schemes. Section 7.4 concludes the description of the relational implementation by presenting the metadata model of the analysis
- ◆ layer, responsible for capturing the multidimensional semantics behind the logical model.

The remainder of the thesis is dedicated to enhancing the presentation layer of data warehousing systems and summarizing the contributions of the entire modeling framework:

- ◆ Chapter 8 addresses the limitations of the data warehouse systems at the application and presentation layers. Specifically, we focus on the approaches to interactive visual specification of ad hoc queries and consider established and novel visualization techniques adequate for advanced exploration and analysis of multidimensional data. Section 8.1 sets the stage by introducing the overall visual exploration framework and reviewing the state of the art, both in research and in commercial tools, in the field of visualization techniques for OLAP. In Section 8.2 we focus on the data navigation paradigm for interactive query specification, consider the prevailing exploration patterns and elaborate on the implementation of logical OLAP operators in a visual framework. The aim pursued in Section 8.3 is to enhance the exploratory framework by employing hierarchical visualization techniques for displaying the results of a series of aggregations in a single view: we introduce Enhanced Decomposition Trees as a specialized hierarchical visualization technique for OLAP, followed by inspecting various layout, scaling, visual mapping, and interaction options.
- ◆ Finally, Chapter 9 summarizes the contributions of the thesis, draws conclusions, and identifies future research directions we regard promising in the context of this work.

The outline of the thesis reflects the multi-layer structure of data warehouse systems depicted in Figure 1.1. This correspondence between the thesis structure and the data warehouse reference architecture is shown in Figure 1.2. Chapters 3 to 5 address the backend layer by tackling the challenges of the conceptual modeling. Chapter 6 proposes a methodology for automating the acquisition of conceptual multidimensional schemes from the models of the underlying data sources and operational systems. Relational implementation guidelines and the metadata model presented in Chapter 7 belong to the application layer. The visual exploration framework in Chapter 8 concentrates on the enhancements of the analysis and presentation layer.

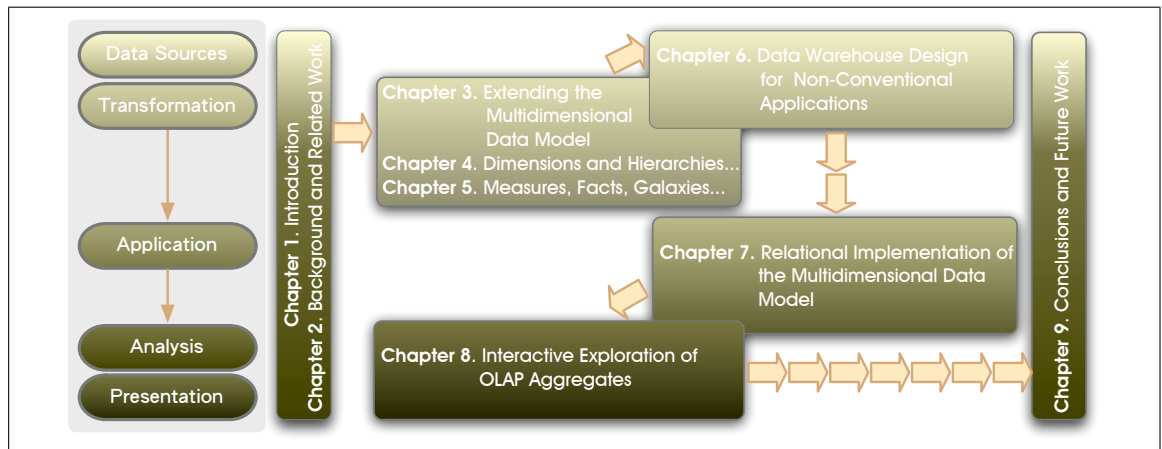


Figure 1.2: Thesis outline with respect to the multi-layer data warehouse reference model

Due to a high degree of independence and heterogeneity of the concepts and techniques relevant within the scope of this thesis, we have decided against putting the entire related work into one chapter. Instead, Chapter 2 concentrates on the contributions relevant for presenting the fundamental terminology and describing the background of our research, whereas more specific related work appears in the respective chapters, interleaved with the flow of the thesis.

Chapter 2

Background and Related Work

IN THIS CHAPTER, the relevant terminology and concepts are introduced, starting from the overall Business Intelligence framework and the data warehousing environment and proceeding to the elements of the data warehouse system architecture, employed data models and operations, implementation alternatives, and the design methodology. The last section highlights the fundamentals of visual OLAP as an emerging paradigm for interacting with multidimensional aggregates.

Contents

2.1 Business Intelligence and its Components	9
2.1.1 Business Intelligence	9
2.1.2 Data Warehousing	11
2.2 OLAP and the Multidimensional Data Model	14
2.2.1 Elements of the Multidimensional Data Model	14
2.2.2 OLAP Operations	17
2.2.3 OLAP Implementation Alternatives	20
2.2.4 Data Warehouse Design Methodology	24
2.3 Visual Analysis and Exploration	29
2.3.1 Visual OLAP as an Emerging Trend	30
2.3.2 Visual Exploration Framework	31

2.1 Business Intelligence and its Components

Due to the fact that data warehousing and its related concepts have been influenced by the technical as well as by the business application experts, there exist discrepancies in the definition of some terms. As this thesis focuses on the database aspects of data warehousing research, we pursue a technical definition perspective.

2.1.1 Business Intelligence

Coined as a term by the Gartner Group analyst Howard Dresner in 1992 [10], *Business Intelligence (BI)* is a popularized umbrella term encompassing a set of concepts and methods to improve business decision making

by using fact-based support systems. Though often used synonymously with *decision support*, the former is technically much broader, potentially encompassing knowledge management, enterprise resource planning, and data mining, among other practices. In the abundance of definitions emphasizing various aspects of BI, such as information processing, logistics, assessment, alerting, etc., we adopt the following definition:

“Business intelligence (BI) is a broad category of applications and technologies for gathering, storing, analyzing, and providing access to data to help enterprise users make better business decisions. BI applications include the activities of decision support systems, querying and reporting, online analytical processing (OLAP), statistical analysis, forecasting, and data mining” [171].

Figure 2.1 shows an attempt to structure diverse BI perspectives by arranging them on a two-dimensional plane, as proposed by Gluchowski [46] and modified in [80]. The phases of the analytical data processing are ordered along the vertical axis, whereas the horizontal axis differentiates between the technology and the application focus. Based on the positioning of the application classes, Kemper et al. [80] propose to distinguish between three prevalent definitional levels:

- ◆ **Narrow BI definition** is limited to a few core applications with a straightforward decision support function, such as OLAP, Management (MIS) and Executive (EIS) Information Systems.
- ◆ **Analysis-oriented BI definition** encompasses the entirety of end-user tools and applications that enable interactive analysis. Such tools include data mining, reporting, balanced scorecards, etc.
- ◆ **Broad BI definition** goes beyond the tools of the presentation layer by comprising all applications employed for decision support, directly or indirectly. ETL software falls into this category.

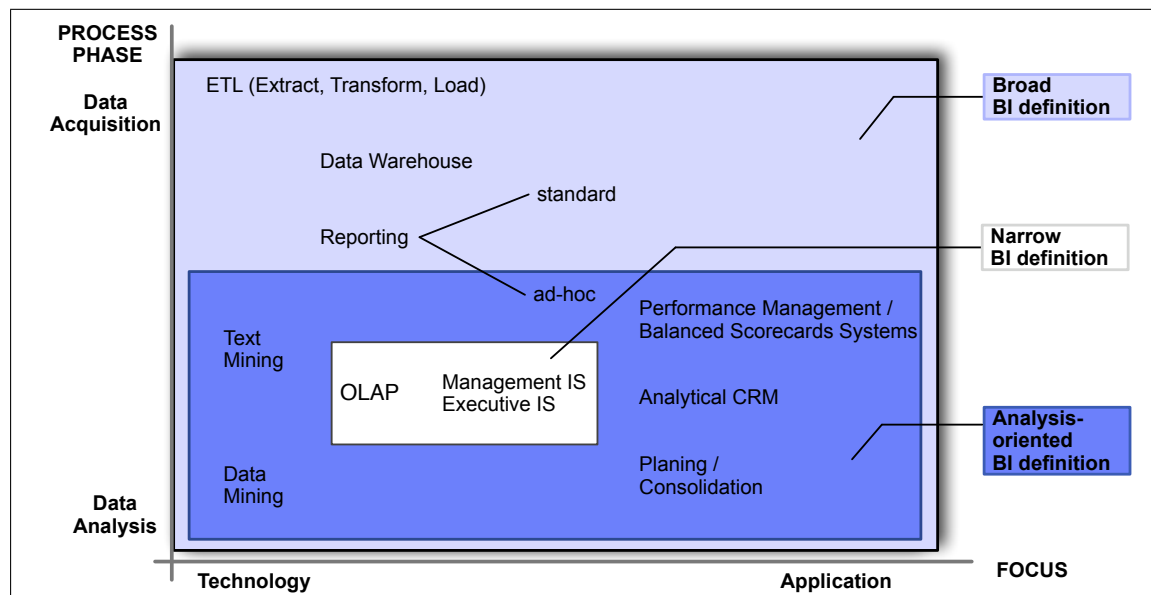


Figure 2.1: Definitional levels of Business Intelligence

2.1.2 Data Warehousing

Data warehousing is a field that has emerged from the integration of a number of different technologies and experiences over the last two decades. W. H. Inmon coined the term “*data warehouse*” as early as in 1990 with the following definition:

“A *data warehouse* is a *subject-oriented, integrated, non-volatile, and time-variant collection of data in support of management’s decisions. The data warehouse contains granular corporate data*” [68].

The four salient characteristics enumerated in the above definition are explicated in Table 2.1.

Table 2.1: Data warehouse characteristics according to W. H. Inmon

Characteristic	Explanation
<i>Subject-oriented</i>	The data is modeled according to the subject area of the respective enterprise, and not according to the application needs of operational systems. The topics of the analysis are enterprise-specific. Thereby, a proper perspective on the data from the decision-maker’s point of view is provided.
<i>Integrated</i>	The data fed from multiple sources has to undergo extensive transformations to be brought into a coherent state. The main challenges here are to ensure consistent formatting, naming, data coding, and measurement units.
<i>Non-volatile</i>	The data is loaded in a snapshot, static format; existing entries are not supposed to be further manipulated or deleted. Analytical operations are <i>read-only</i> .
<i>Time-variant</i>	Each data unit is accurate with respect to some point or period in time. The <i>time</i> dimension is used to characterize the validity of the facts. Aggregation along time and evolution in time are the core analysis types in data warehouses.

As the concept of data warehousing matured over time, other definitions were proposed. R. Kimball provided a rather simple but accurate definition of a data warehouse as “a copy of transaction data specifically structured for query and analysis” [81]. Finally, the end user perspective is stressed by Jarke et al. who define a data warehouse as a “collection of technologies aimed at enabling the knowledge worker (executive, manager, and analyst) to make better and faster decisions” [72].

The term “*data warehouse system*” comprises the data warehouse itself as well as its accompanying components, such as design and ETL tools, Operational Data Store, metadata repository, analysis and presentation tools of the end-user. A classical reference architecture of a data warehouse system [8, 24, 89, 92, 138, 151, 161] depicted in Figure 2.2 is a refinement of the simplified version from Figure 1.1. We use the 5-layer model proposed in [151], in which each layer encapsulates a different stage of the data flow in the system.

The *Data Sources Layer* encompasses all information sources, primarily the company’s own operational databases, which function as data suppliers for a warehouse. In addition to the internal data, external data sources, such as third-party demographic and statistical databases, market research reports, and web documents are frequently used to enrich the analysis base.

The task of the *ETL Layer* is to extract data from heterogeneous sources, cleanse it into a consistent state, transform it according to the target schema, and, finally, load it into the data warehouse. A set of activities required to populate data warehouses and OLAP applications with cleansed, consistent, integrated,

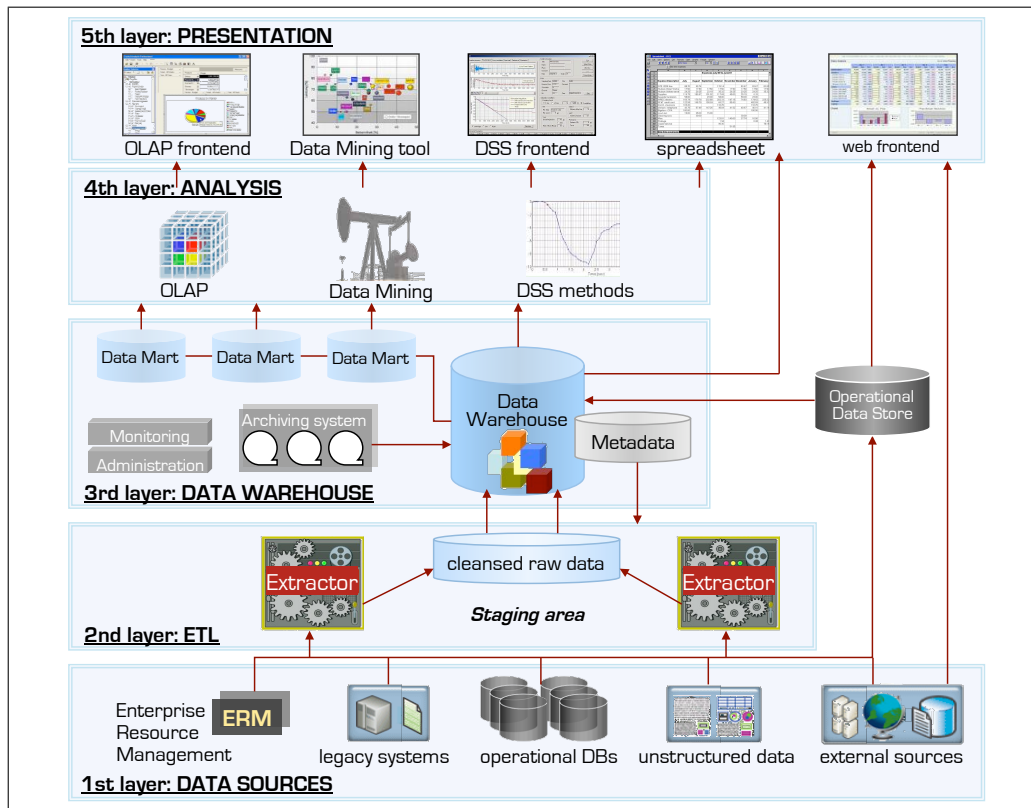


Figure 2.2: A multi-layer data warehousing system architecture

and probably summarized data is described by the term *ETL* (Extract, Transform, Load). The ETL process is subdivided into two phases: *i*) definition and *ii*) execution. The definition phase can be interpreted as a specification of the data warehouse metadata objects, whereas the execution phase uses the framework defined in the first phase to carry out the actual data loading routine (on a time-driven or even-driven basis) [89]. The entire latter phase takes place in a designated storage system called *staging area*, which is known as the “back room” portion of the data warehouse environment that lies out of bounds for end-users.

From the ETL layer, the transformed data is transferred to the *Data Warehouse Layer*, which is a special purpose database along with its metadata repository. Ideally, there exists a single centralized data warehouse consolidating the entire company’s data. In practice, however, organizations frequently switch to a decentralized data mart architecture. A *data mart* is a departmental data warehouse subset focused on a specific subject. As for the relationship between the data warehouse itself and the data marts, there exists two paradigms: Inmon suggests that data marts source their information from the enterprise-wide data warehouse, whereas Kimball defines the data warehouse to be the union of all data marts within the enterprise [67]. While current practices tend to be closer to Kimball’s approach, there is an emerging trend towards Inmon’s vision. Data marts can coexist with the main data warehouse or be fully decentralized, each disposing of its own ETL tools. The disadvantage of the latter approach is the danger of facing integration problems if the initial design does not reflect a complete business model [24, 151].

Another integral component of the data warehouse layer is the metadata repository that enables shared

access to metadata by various tools and processes. *Metadata* refers to the data required for managing the data warehouse and comprises administrative (setup, configuration, database objects and rules), business (definitions, ownership, access rights), and operational (origin, currency, usage statistics) metadata [24]. In the world of data warehousing, the administrative metadata, which describes the structure of the available data, functions as an index to the actual contents as the former allows the end-user to navigate through the data and analyze it interactively [68].

Operational Data Store (ODS) is a storage structure residing outside of the data warehouse environment and providing integrated real-time detailed data obtained from operational systems. In contrast to a data warehouse, which contains historical and summarized data, an ODS stores current operational data to support the demand for near-real-time data, e.g., for operational reporting or tactical decision making. Considering the growing analytical interest for fine-grained real-time data, Kimball proposes to relocate the ODS by coupling it tightly with the data warehouse as the “front edge” of the latter [84].

The *Analysis Layer* encompasses analysis methodologies and techniques, such as OLAP and data mining, which form the basis for the end-user BI tools. The term *OLAP* (On-Line AnalYTical Processing), synonymous to *multidimensional data analysis*, was coined in 1993 by the inventor of the relational data model E. F. Codd to describe a kind of software that analyzes business data in a top-down hierarchical fashion:

“OLAP is the name given to the dynamic enterprise analysis required to create, manipulate, animate, and synthesize information from exegetical, contemplative, and formulaic data analysis models. . . This includes the ability to discern new or unanticipated relationships between variables, the ability to identify the parameters necessary to handle large amounts of data, to create an unlimited number of dimensions (consolidation paths), and to specify cross-dimensional conditions and expressions” [30].

Data mining techniques provide advanced predictive and analytical functionality by identifying distribution patterns (segmentation), characteristic behaviors (classification) and relationships (association) within a dataset [24, 151].

Finally, the *Presentation Layer* consists solely of the frontend analytical applications, commonly referred to as BI tools, for reporting, querying, and mining the data. Presentation tools differ in the degree of freedom (pre-defined vs. ad hoc queries), complexity and customizability to individual requirements. With the recent advancements in the Internet technology, there is a clear trend towards browser-based frontend solutions. Another trendy development is a unification of diverse analysis toolkits in a comprehensive BI platform, based on an organization-wide business model. Examples of mature BI platform solutions are Oracle BI Suite [133] and BusinessObjects Enterprise [14]. A pioneering initiative in developing an open-source BI suite is Pentaho BI Platform [139], which is entirely web-based and highly customizable.

Obviously, each data warehouse solution has to be designed and implemented individually and in accordance with specific requirements of a given organization and the application domain. However, there are certain general characteristics that make out the advantages of the data warehousing approach [92, 151]:

- ◆ availability of an organization-wide unified and consistent data model,
- ◆ ability to benefit from external data sources,
- ◆ maintenance of historic and summarized data in a separate database,
- ◆ no interference with operational sources,
- ◆ optimized performance for complex queries,
- ◆ accessibility to a wide range of users,
- ◆ user-friendly frontend tools.

Subject-orientation guarantees applicability of data warehousing to virtually any application domain. Data warehouse solutions are employed in trade, finance, banking, insurance, production etc. Initially designed for satisfying decision support needs of business enterprises, the performance-oriented BI approach has recently found its way to a multitude of non-conventional applications, such as life sciences, health-care,

academia, and government, to name a few distinguished fields. In the next section we take a closer look at the OLAP technology and investigate what factors contribute to its nearly universal applicability.

2.2 OLAP and the Multidimensional Data Model

Data warehouses are targeted at enterprise decision support, which is based on analyzing key performance indicators presented as *measurable facts*, or in short *measures*, e.g., sales volume, turnover, ROI (return on investment), etc. Complex analytical queries aggregate over large volumes of consolidated data performing a series of expensive table scans and joins and thus result in heavy workloads. Apparently, data warehouse performance requirements are quite different from those of the OLTP (On-Line Transaction Processing) systems. These requirements are met by employing the OLAP approach and its underlying multidimensional data model. Standard terminology for OLAP is provided by the OLAP Council [134].

2.2.1 Elements of the Multidimensional Data Model

The multidimensional data model is aggregation-centric, i.e., it uses numeric measures as its analysis objects [24]. A fact entry is identified at the finest available granularity and normally corresponds to a single transaction or an event. Parameters that determine the values of a measure are referred to as dimensions. Consider an example from the domain of academic administration, with the number of enrollments as the measure of interest. The associated dimensions are Semester, Degree, and Country. Often, the members of a dimension are organized in a containment-type hierarchy to enable additional aggregation levels. For instance, single countries in Country dimension can be grouped into subcontinents and, subsequently, into continents.

DATA CUBES

A natural representation of a set of fact entries along with the associated dimensions and their hierarchies is known as a *multidimensional data cube*, or a *hypercube*. The number of dimensions in a cube corresponds to the dimensionality of its contained measure(s). Each dimension can be interpreted as an axis in a multidimensional space with the dimension's member values as its coordinates. Finally, each cell contains a value of the measure defined by that cell's dimensional coordinates.

Depending on the application, cubes may range from dense (each cell has a non-null measure value) to sparse (significant portion of empty cells) ones. The sparsity tends to increase with the increasing number of dimensions and with the increasing granularity within a dimension. Figure 2.3 shows a sample 3-dimensional cube (fragment) with student enrollment numbers. In addition to the original fact entries (white cells), slices of subaggregates, computed for each combination of dimensions (colored cells) as well as the absolute total value (a dark-grey cell), are shown. Despite the implied 3-dimensionality of the term *cube*, in OLAP there is no limitation on the number of dimensions in a cube.

DIMENSIONS

Dimensions represent a crucial concept in OLAP as they provide the context for analyzing the facts – analysts use them to filter the data and aggregate it to the desired level of detail. Each dimension of a cube corresponds to an axis in a multidimensional data space. The values of a dimension are called *dimension elements*, or *members*. Members may be arranged into a *classification hierarchy*, composed of multiple levels, with each level representing a distinct granularity within the dimension. Each member is mapped to a *classification node* in the hierarchy, with the members of the finest grain as the leaf nodes [8].

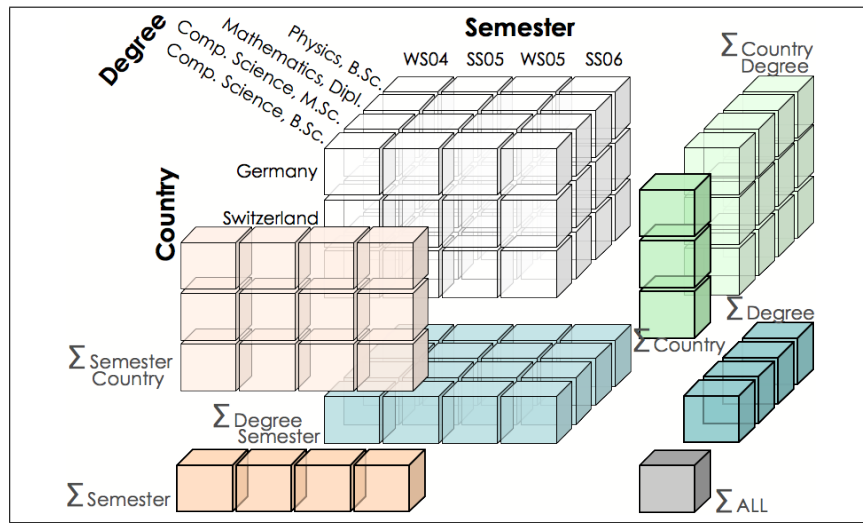


Figure 2.3: A sample 3-dimensional cube (fragment) storing student enrollment numbers (white cells) along with its 2-dimensional, 1-dimensional, and 0-dimensional projections (colored cells)

In the multidimensional data model dimensions are represented via their *classification schemes*, in which each level of the hierarchy, denoted a *classification level*, or *category*, is a node connected by an edge to its parent level. In some literature, the classification scheme is termed *intension*, with the actual classification hierarchy building its *instance*, or *extension* [142]. Fact entries are linked to a dimension exclusively at the bottom level of the latter, whereas upper levels are used for querying the respective aggregates. Distinct paths in a classification scheme are referred to as *consolidation paths*. A classification level is allowed to be composed of multiple attributes.

The analysis may be further enhanced by defining multiple classification hierarchies within the same dimension. Consider Degree dimension in the aforementioned student enrollment example. The bottom-level members are the degrees into which students are enrolled. A degree is composed of the attributes Study Subject (e.g., “Physics”) and Degree Type (e.g., “Bachelor”). Intuitively, single degree values can be grouped by Study Subject, on the one hand, and by Degree Type, on the other hand. Figure 2.4 shows the graph of the resulting aggregation paths within Degree as well as the actual data hierarchy behind each aggregation path.

The attribute upon which the hierarchy is defined is called the *analysis criterion*. Hierarchies within a dimension may refer to the same or to different analysis criteria. Hierarchies depicted in Figure 2.4 refer to various criteria: one classification is based on Degree Type, while the other draws upon Subject. Attributes holding non-hierarchical characteristics of the category’s members are called *properties*. In our example, Department category in Degree dimension may have properties such as Dean, Location, and Foundation Date. Property role of any attribute is not global but is limited to the context of a given hierarchy. For example, Location attribute of Department is a property in the context of the subject hierarchy in Degree. However, Location may be used as an analysis criterion for defining a hierarchy of department locations, such as Location \nearrow District \nearrow City.

Notice that dimension hierarchies are normally defined in terms of the partial ordering, i.e., as parent-child relationships between its members with no ordering between the members of the same level. However, for some dimensions, such as time, the total ordering on its members is given. The conventional OLAP approach requires dimension hierarchies to be strict, regular and balanced to ensure correct aggregation.

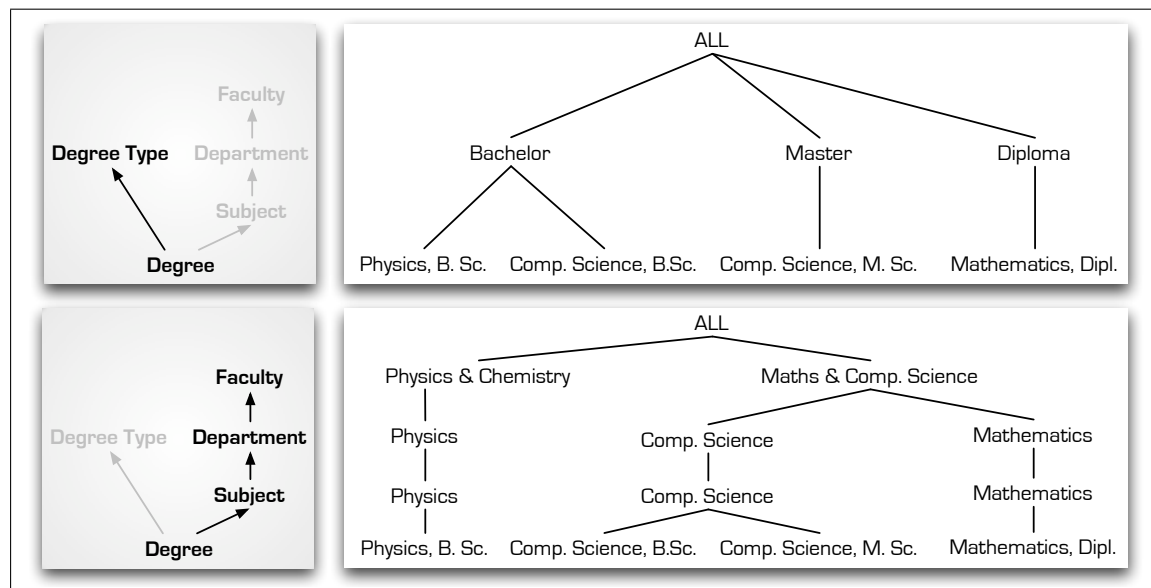


Figure 2.4: Dimension Degree with multiple hierarchies: scheme (left) and instances (right)

FACTS

Facts represent the subjects of the analysis. The *granularity*, or level of detail, of the facts corresponds to the atomic level of the modeled business subject. For example, the granularity of the student enrollment facts from the cube depicted in Figure 2.3 is given by degree, country, and semester. The finest granularity and homogeneity are ensured by requiring each fact entry to map to the bottom level in each of its dimensions.

Kimball identifies three fundamental types of facts [81]:

- ◆ *Transactional facts* track the occurrence of events, with each detailed event captured into a fact entry and the measures being additive across all or most of the dimensions.
- ◆ *Periodic snapshots* capture the states of an entity at given points in time, such as inventory levels or account balance. The same entity (e.g., a product) and its state is registered as a new fact for each point in time. The measured states are not additive across time but additive across other dimensions.
- ◆ *Accumulating (cumulative) snapshots* capture the states of an entity up to a certain point in time with respect to some initial point, common for all facts. An example of cumulative snapshots is the total number of website visitors recorded at daily basis. The measures are non-additive over time (as they already hold accumulated values) but are additive across other dimensions.

Apart from the measurable fact types listed above, there exist useful fact types that do not contain any measures, i.e., consist of nothing but a set of dimensions. Kimball denotes such facts *factless* and names their common usage scenarios [83]:

- ◆ *Event-tracking facts* simply store the occurrence of an event itself (e.g., every single student enrollment case) without any specific measure of interest.
- ◆ *Coverage facts* are useful for tracking whether something has or has not happened. An example of a coverage fact type is the entirety of individual student application records that have or have not resulted

in an enrollment. Such facts are useful for computing the acceptance rate.

- ◆ Any *many-to-many relationship* is a fact by definition.

Many real-world scenarios combine various types of facts to support complementary analysis tasks.

MEASURES

Measure is a fact property that the users want to analyze, predict, or optimize. The “holy grail” of the multidimensional design is that the most useful measures are *numeric*, *continuously valued*, and *additive* [81]. A measure of a query is defined by specifying a formula, usually a simple aggregate function, such as sum, that combines several measure values into one [143]. The measure attribute(s) and the formula should be chosen as to provide a meaningful value for all aggregation levels. Calculated, or derived, measures result from applying a function to one or more measure values pertaining to the same fact entry. An example of a derived measure is amount, obtained by multiplying price with quantity.

With respect to its aggregation behavior, each measure falls into one of the following three classes:

- ◆ *Fully additive* measures can be summed up through all of the fact’s dimensions. The number of enrollments in the cube in Figure 2.3 is an example of such a measure.
- ◆ *Semi-additive* measures may be totalled along some of the dimensions but not all of them. This kind of measures is typically encountered in snapshot facts, in which the values may not be aggregated over time.
- ◆ *Non-additive* measures may be added up along no dimension at all. Values of type measurement or metric, such as age, height, or density are examples of numeric values non-additive in many contexts.

Measure additivity is determined at design time and stored in the data warehouse as metadata to be used for checking the validity of attempted queries.

Aggregate functions can be classified into three categories [51]:

- ◆ *Distributive* functions, such as COUNT(), MIN(), MAX(), and SUM() can be computed by partitioning their input into disjoint sets, aggregating each set individually, and then aggregating individual subaggregates into the final result.
- ◆ *Algebraic* functions can be expressed as a scalar function with M arguments (where M is a bounded integer), each of which is obtained by applying a distributive aggregate function. For example, AVG() can be expressed as SUM()/COUNT().
- ◆ *Holistic* functions cannot be computed by partitioning as the whole input is required for computing each output value. Common examples of such functions are MEDIAN(), MostFrequent(), and RANK().

Characterization of aggregate functions is necessary for determining their *self-maintainability*, i.e., whether new aggregates can be computed directly from the old output of the function and from the changes to the base data [100].

2.2.2 OLAP Operations

Cube structure builds the foundation of OLAP applications. A variety of cross-dimensional calculations and aggregations for real-time data analysis can be performed within a cube or across multiple cubes by applying specialized query operations. OLAP operators pursue different tasks, from specifying the subset of interest and manipulating its dimensionality and granularity to filtering the dataset, obtaining new measures by linking multiple cubes, ranking the aggregates according to some function, specifying user-defined hierarchies, etc. The operators can be subdivided by function into the following groups:

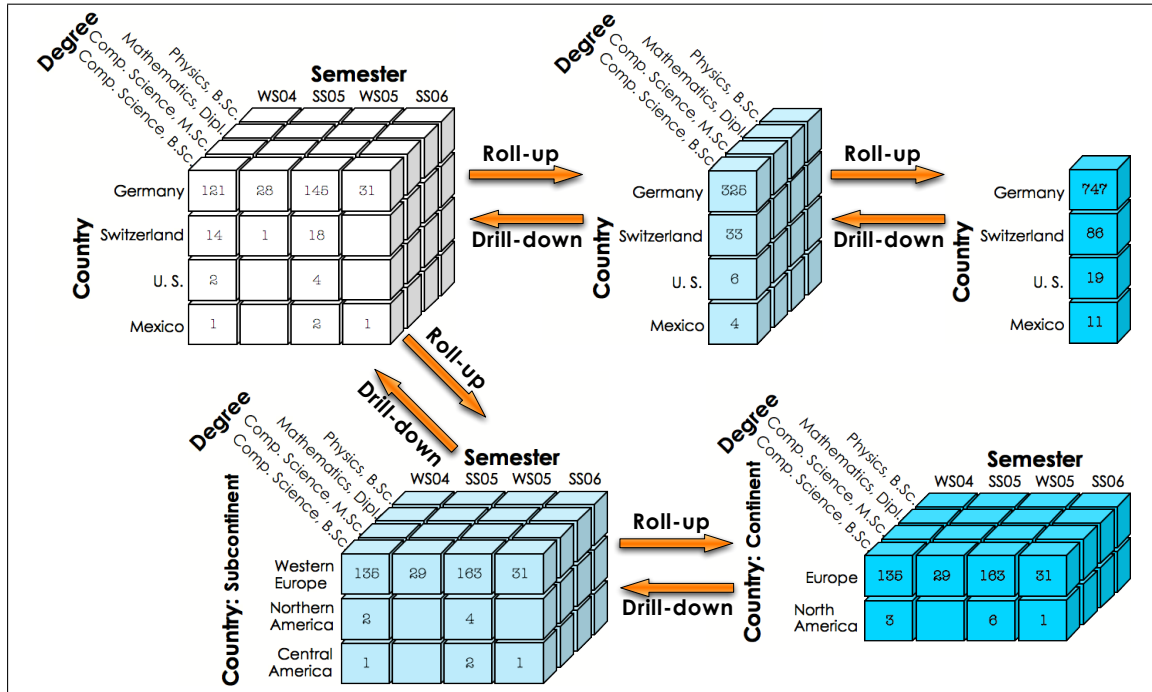


Figure 2.5: DRILL-DOWN and ROLL-UP across dimensions (top) and within a dimension hierarchy (bottom)

1. *Aggregation* operators DRILL-DOWN and ROLL-UP and their variants (DRILL-THROUGH, DRILL-WITHIN, DRILL-ASIDE, PROJECT) manipulate dimensionality and granularity of the output data cube. The ability to drill down into any level of any dimension is known as “drilling anywhere”.
2. *Filtering* operators SLICE&DICE and its special cases (SLICE, DICE, SELECT, FILTER, CONDITIONAL HIGHLIGHTING) reduce the size of the data set by adding predicates on dimensional characteristics. RANK operator applies filtering and ranking to the aggregated values themselves.
3. *Reordering* operators PIVOT (ROTATE) and SWITCH enable visual rearrangement of the output without any changes in the underlying dataset.
4. *Restructuring* operators DRILL-ACROSS, DRILL-AROUND, PUSH&PULL, and INSERT/DELETE LEVEL transform the cube’s schema to obtain new measures, dimensions, or hierarchy levels.

To gain an insight into the specifics of OLAP queries, we provide a detailed description of the popular operators from the above enumeration:

- ◆ DRILL-DOWN and ROLL-UP are inverse operations that use dimension hierarchies to perform aggregation steps. ROLL-UP aggregates a measure to a coarser granularity whereas DRILL-DOWN navigates from aggregated data to a higher level of detail. Drilling down and rolling up by respectively adding and eliminating dimensions from the drill path changes the dimensionality of the resulting cube, as shown in the upper part of Figure 2.5. However, the dimensionality remains unaffected when drilling within a dimension, as presented in the lower part of Figure 2.5 at the example of Country dimension.
- ◆ DRILL-THROUGH allows to jump back to the original fact data behind the selected aggregate values.

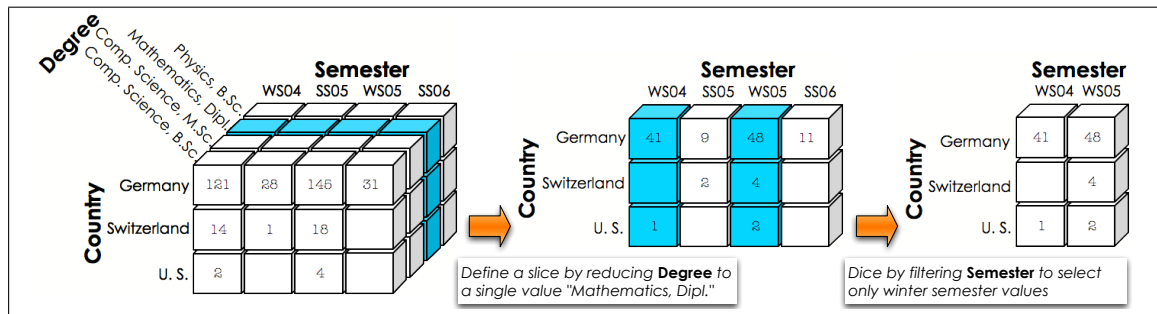


Figure 2.6: Example of a SLICE&DICE operation

- ◆ **DRILL-WITHIN** refers to switching from one classification to another within the same dimension. Drill-within prompts the user to select the path for drilling down at each level where multiple hierarchies are encountered within a dimension. For instance, at the Year level the user may choose to drill either into Month or into Week.
- ◆ **DRILL-ASIDE** enables navigation in a many-to-many mapping of the aggregation path. For instance, when drilling from Year down to Week, the last week of a year may partially belong to the next year and, therefore, the user is prompted to decide how the respective subaggregates should be handled.
- ◆ **SLICE&DICE** corresponds to reducing the cube's dimensionality by projecting the data onto a subset of dimensions while setting other dimensions to selected values, and is composed of two sub-operations: SLICE selects a subcube corresponding to a single value for some dimension in the drill path, while DICE reduces the size of a slice by filtering its data along any dimension(s) in the drill path. Figure 2.6 visualizes the effects of a slice&dice in the student enrollments cube: Degree dimension is "sliced" down to a single value "Mathematics, Dipl." with subsequent "dicing" of Semester dimension by selecting values "WS04" and "WS05".
- ◆ **SELECT** is the dual of dicing: rather than selecting the elements of a dimension to be included, the user is prompted to specify the condition on eliminating the data from the result.
- ◆ **RANK**, or top n /bottom n queries, retrieves only the first/last cells in the result sorted by the aggregate's value – for example, the 10 most popular degrees in the academic year 2004/05.
- ◆ **PIVOT**, or **ROTATE**, is a visualization operation that rotates the dimension axes in the view in order to provide an alternative presentation of the data. The data subset itself remains unaffected. Figure 2.7 gives an example of pivoting the original data cube by means of two rotations.
- ◆ **DRILL-ACROSS** allows to query multiple cubes that have at least one common dimension, combining the results into a single data set [48, 82, 145]. The join is preceded by the **ROLL-UP** of all participating cubes to the set of their common dimensions to ensure compatible granularity of the measures to be combined. Consider a sample task of deriving a new measure acceptance ratio (i.e., the number of enrollments divided by the number of applications), which can be achieved by combining enrollment facts with the corresponding application facts. The latter is available in form of a 2-dimensional cube with dimensions Semester and Country. Figure 2.8 demonstrates the intermediate operation of aggregating the enrollment facts by Semester and Country – the set of dimensions common to both cubes – and the final step of computing a new cube. Advanced drill-across options are elaborated in [4].
- ◆ **PUSH** and **PULL** operations, proposed in [5], enable the interchange of dimension and measure roles in a query. The push operator is used to convert a dimension into a measure to be aggregated. The pull

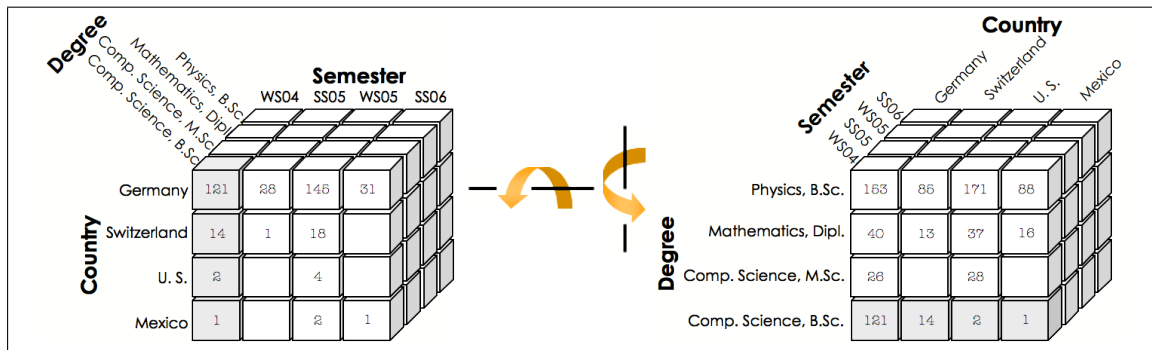


Figure 2.7: Example of a PIVOT operation with two rotations (shaded cells are visible in both views)

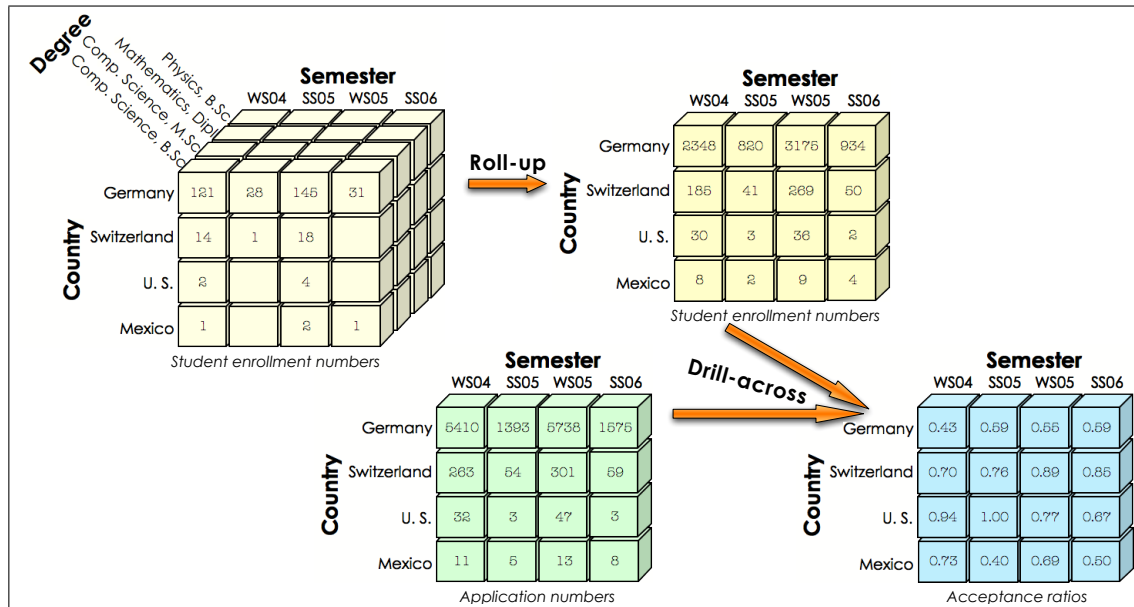


Figure 2.8: Example of a DRILL-ACROSS operation with two input cubes

operator is the converse of push: it converts a measure attribute into a dimension. Combined, push and pull enable uniform treatment of measure and dimension characteristics.

2.2.3 OLAP Implementation Alternatives

OLAP tools do not indicate how the data actually has to be stored. Hence, there exist multiple ways to implement a data warehouse, with the following two prominent architectures:

- ◆ *Relational OLAP (ROLAP)* [65] systems store data in relational DBMS and employ SQL extensions and specialized access structures to efficiently implement OLAP functionality.

- ◆ *Multidimensional OLAP (MOLAP)* [38] systems store data in specialized multidimensional data structures (such as arrays or cubes) and implement OLAP operations over these structures.

Apart from the fundamental distinction in data storage and processing capabilities, there is a conceptual difference between the two implementation options: MOLAP pursues a top-down approach by first focusing on business problems and then identifying measures and dimensions of interest, so that the metadata model may be built prior to the acquisition of the relevant data sources; ROLAP, in contrast, encourages a bottom-up analysis to identify candidate facts and dimensions in the relational data models of existing data sources [37].

Both paradigms have their benefits and weaknesses – the latter, however, being rapidly addressed by the respective vendors. Currently, data warehouses are predominantly built using ROLAP, especially when dealing with huge data volumes [89]. ROLAP attributes its success to such factors as the established and proven technology, good scalability in terms of the number of facts and their dimensionality, flexibility under cube redefinitions, and support for frequent updates [143]. MOLAP needs less storage space and generally delivers better performance due to specific indexing, compression and storage optimizations. An overview of the pros and cons of ROLAP and MOLAP based on the findings in [35, 56, 66, 85] is given in Table 2.2.

Hybrid OLAP (HOLAP) encompasses a range of solutions that combine ROLAP and MOLAP to benefit from the high scalability of ROLAP and a superior performance of MOLAP. For example, a HOLAP system may store large volumes of fine-grained facts in a relational database while materialized aggregation results are kept in a MOLAP store [54].

Table 2.2: ROLAP versus MOLAP

Criterion	ROLAP	MOLAP
<i>Performance</i>	✗	✓
<i>Maintenance</i>	✗	✓
<i>Compactness</i>	✗	✓
<i>Analytic Capabilities</i>	✗	✓
<i>Data Loading</i>	✗	✓
<i>Scalability</i>	✓	✗
<i>Portability</i>	✓	✗
<i>Standardization</i>	✓	✗
<i>Flexibility</i>	✓	✗
<i>Stability & Recoverability</i>	✓	✗
<i>Schema Evolution and Updates</i>	✓	✗
<i>Join and Star Join Operation</i>	✓	✗
<i>Complex Predicates</i>	✓	✗
<i>Table Scans</i>	✓	✗
<i>Data Partitioning</i>	✓	✗
<i>Parallel Query Execution</i>	✓	✗
<i>Dynamic Consolidation</i>	✓	✗
<i>Batch Consolidation</i>	✓	✓
<i>Indexing and Hashing</i>	✓	✓

This thesis focuses on extending the capabilities of relational OLAP as the latter appears a more suitable option for handling large volumes of complex high-dimensional data encountered in novel application scenarios. In fact, our solutions in part of handling non-summarizable and heterogeneous classification hierarchies explicitly exploit the flexibility of the relational database technology as will be shown in Chapter 4.

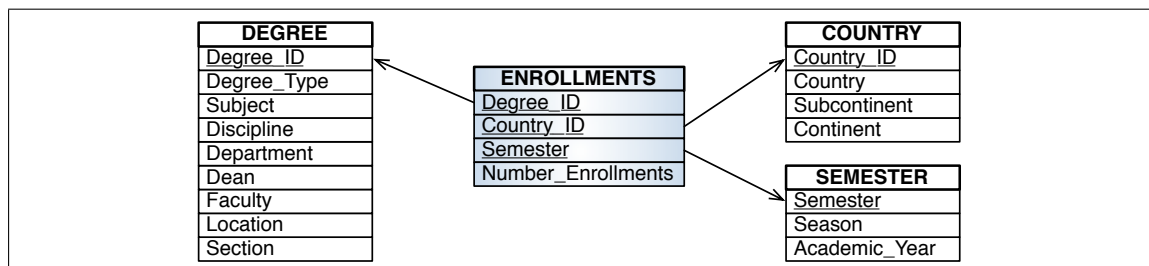


Figure 2.9: A sample star schema

ROLAP stores data cubes in relations of two types: *i*) fact table and *ii*) dimension table. A *fact table* stores a set of uniformly structured facts of the same grain, one row per fact entry. The facts may be either of the finest granularity or aggregated. Fact tables that contain aggregated facts are called *summary tables*. A fact table is composed of two types of columns – measures and dimensions – where each dimension column is a foreign key (generated foreign keys are preferred for efficiency reasons) to the respective dimension table. The primary key of a fact table is usually a composite key made up of all its foreign keys, i.e., the dimension columns. A *dimension table* is used for storing the members of a dimension along with its classification hierarchies. Dimension tables are sometimes called *lookup* or *reference tables*.

In the context of ROLAP systems, the two logical data warehouse design options are the star schema and the snowflake schema [81]. Both schemata differ solely in the way they handle dimension hierarchies. *Star schema*, used in most data warehouses, places each dimension with all its hierarchies into exactly one denormalized relation. The relational schema of a dimension table is given by the set of all level and property attributes of the respective dimension. Thereby, the star schema provides no explicit support for classification hierarchies and is prone to update anomalies. The de-normalized structure of the dimension tables is beneficial for browsing the dimensions and improving the query performance via a star join. Figure 2.9 shows the resulting star schema of the sample 3-dimensional cube with student enrollment numbers.

Snowflake schema is a refinement of the star schema, in which each dimension hierarchy is normalized according to the rules of the relational design to avoid redundancy: each dimension level is placed into a separate table containing the respective dimension level attribute and its property attributes. Lower-level tables also have a foreign key attribute referencing the containing level. Normalized storage is advantageous for avoiding redundancies and thus facilitating update consistency, as well as for handling complex data and for sharing dimension levels. Figure 2.10 shows the snowflake schema obtained by decomposing the dimension tables of the star schema from Figure 2.9.

Multiple fact tables with dimensions modeled using either the star or the snowflake schema may be arranged into a *galaxy schema* [86]. This schema is constructed by allowing dimension tables to be shared amongst multiple fact tables: each fact table is explicitly assigned to the dimensions, relevant for that fact table. This solution is very flexible and powerful, however, it comes at the expense of high design overhead because many variants of aggregation must be considered. Figure 2.11 shows an example of a galaxy constructed from the snowflake schemata of cubes **ENROLLMENTS** and **EXPENDITURES**, which have shared dimension levels **Semester** and **Department**.

When built on top of de-normalized dimension tables, the galaxy schema is limited to full dimension sharing as each dimension table may only be referenced via its primary key, i.e., the bottom-level category. Partial dimension sharing, where facts may refer to the same dimension at different granularity levels, as in the case depicted in Figure 2.11, can only be supported when dimension tables are normalized.

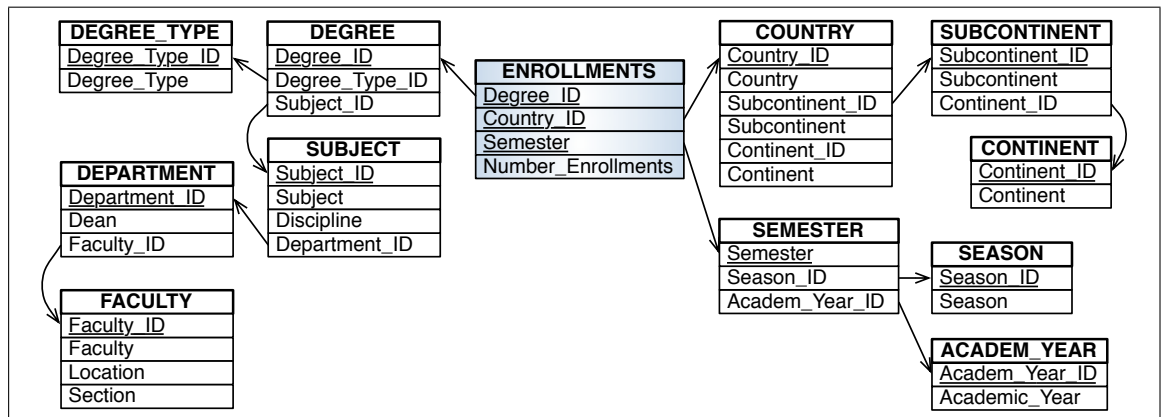


Figure 2.10: A sample snowflake schema

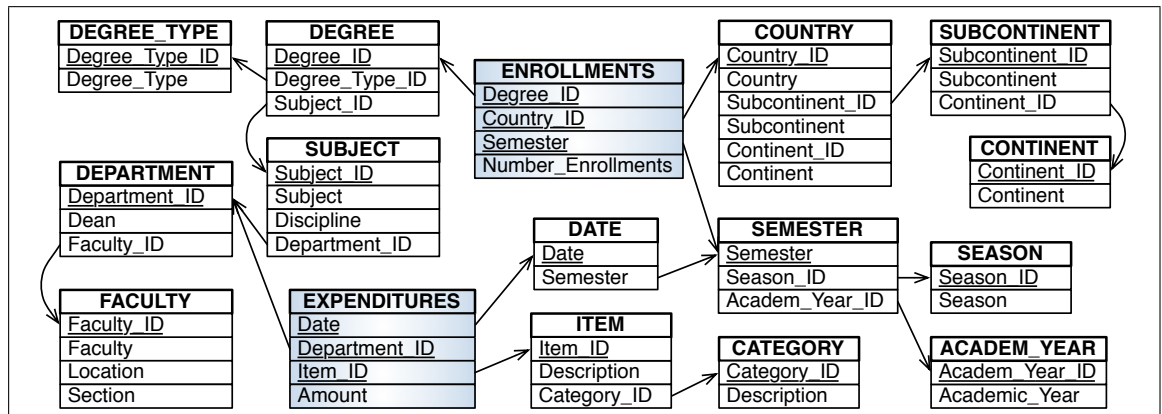


Figure 2.11: A sample galaxy schema with normalized dimension tables

A special case of a galaxy schema, in which multiple facts tables with shared dimensions are hierarchically linked to each other, is called a *fact constellation schema* [86]. Dimensions are modelled following the star schema and multiple related fact tables are the result of pre-aggregating the measures in form of summary tables. Figure 2.12 shows an example of a fact constellation with one base fact table and two summary tables.

Advantages of the star schema (simple design, optimized performance) and the snowflake schema (elimination of redundancies, especially via sharing at subdimension level) can be joined by employing a *star cluster schema* proposed in [125]. This schema has the minimal number of tables while avoiding overlap between dimensions by selectively “snowflaking” dimension tables to separate the segments or subdimensions shared between different dimensions. Figure 2.13 shows the star cluster schema corresponding to the galaxy schema depicted in Figure 2.11.

An evaluation of conceptual models for OLAP can be found in [2, 145] and a survey of logical data warehouse models is published in [180].

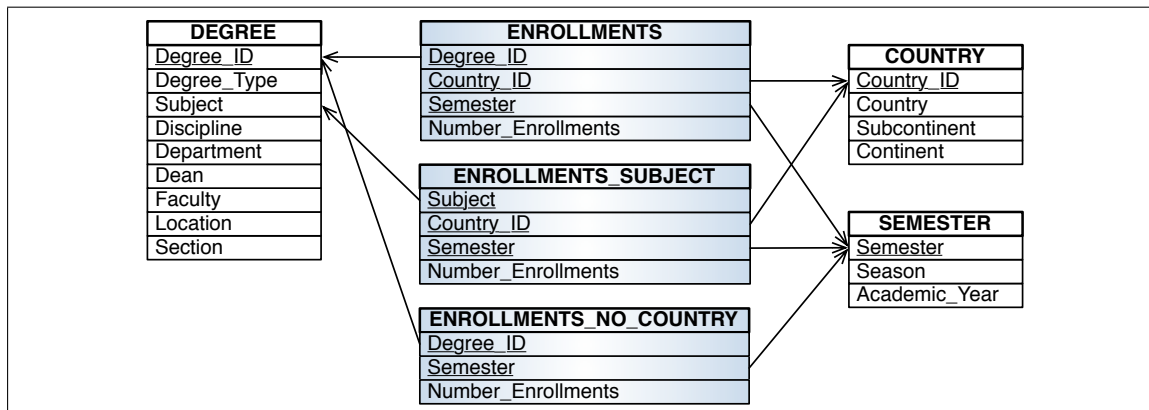


Figure 2.12: Fact constellation schema

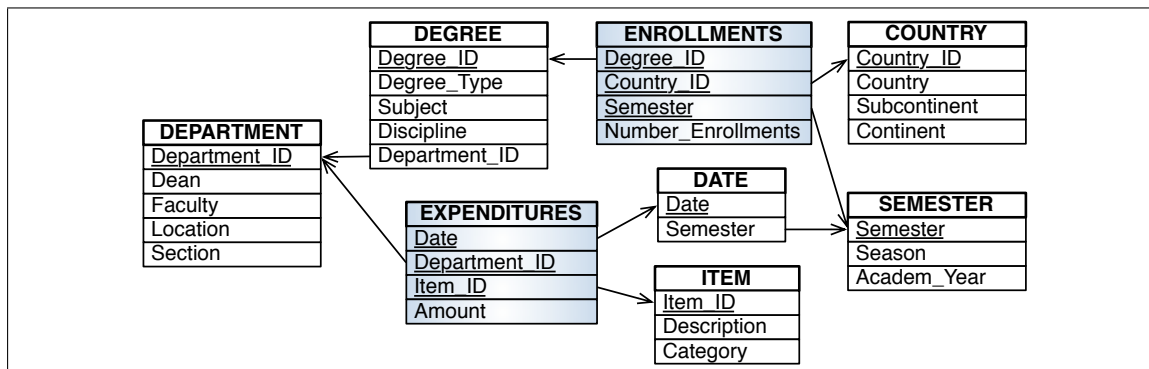


Figure 2.13: Star cluster schema

2.2.4 Data Warehouse Design Methodology

Three abstraction levels recommended by ANSI/X3/SPARC, namely *conceptual*, *logical* and *physical* design, are widely accepted as a sound framework to guide the database modeling process. There is a general acknowledgement of the validity of this framework in the context of data warehouse design [8, 48, 89, 93, 102, 156]. However, there is no agreement as to how these levels should be actually translated into data warehouse design phases [149]. Despite a variety of existing methods, the data warehouse community still lacks universally accepted methods and standards covering all aspects of data warehouse design.

In addition to the three design phases mentioned above, Golfarelli and Rizzi [48] identify two additional phases that precede the conceptual modeling phase:

1. *Analysis of the information system* is aimed at obtaining the (conceptual or logical) schemes of the pre-existing information system.
2. *Requirement specification* consists in collecting the user requirements and outputting the specification concerning the choice of facts, dimensions, measures, and aggregations, on the one hand, and indicating the preliminary workload, on the other hand.

A wealth of approaches for conceptual and logical data warehouse design have been proposed in recent years, introducing different models, formalisms and notations [2]. In this section we present the major contributions in this field and identify the most appropriate techniques to be used in the remainder of the thesis.

CONCEPTUAL DESIGN

Conceptual modeling provides a high level of abstraction for capturing relevant relationships in the application domain and the data to be stored and analyzed in an implementation-independent fashion. The output of this phase is a set of *fact schemes* and the prevailing techniques are based on graphical notations understandable for both designers and end-users. However, generic database design techniques, such as the Entity-Relationship (E/R) model [25] and the Unified Modeling Language (UML) standard [136], are not detailed enough to capture the specifics of the multidimensional data model [8, 89]. Rizzi et al. [156] frame the existing approaches to multidimensional modeling into three categories: *i)* extensions to the Entity-Relationship model, *ii)* extensions to UML, and *iii)* ad hoc models. All those models dispose of the same core expressivity, however, there are significant differences in their ability to handle more advanced concepts. The remainder of this section enumerates prominent contributions from each of the above model types.

Cabibbo and Torlone [18] present a design method based on restructuring the existing E/R schemes in order to explicitly express facts and dimensions as well as classification hierarchies. In the target scheme, the facts are presented as entities, the dimensions of interest are added thereupon, derived either from the existing scheme or from the external sources. Each dimension is refined by defining aggregation levels and property attributes. The restructured E/R scheme serves as input for deriving a dimensional graph. The latter can be obtained automatically and distinguishes between four kinds of nodes: *fact nodes* corresponding to fact entities, *level nodes* representing dimension hierarchy levels, *descriptive nodes* for level properties, and *measure nodes* outgoing from the fact nodes.

Franconi and Sattler [42] propose an extended E/R formalism, which allows for the description of the explicit structure of multidimensional aggregations. The E/R model is extended to represent the structure of *aggregated entities* and multiple classification hierarchies within a dimension. Thereby, aggregations turn into the “first-class citizens” of the representation language, i.e., they may have their own properties and be related to other entities, such as dimensions and other aggregations.

The above two approaches are based on “encoding” the multidimensional semantics into the original E/R constructs. Other authors argue that the E/R model itself has to be extended in order to provide adequate multidimensional modeling constructs. Prominent examples of this class are starER and ME/R models.

Tryfona et al. [175] proposed starER as an extension the E/R model, which offers a specialized entity construct of type *fact set*, relationship set constructs of types *generalization*, *aggregation*, and *membership* (subtyped into *complete*, *non-complete*, and *strict*), as well as the attribute construct of type *fact property*, or *measure*, (subtyped into *stock*, *flow*, and *value-per-unit*).

Sapia et al. [159] present a specialization of the E/R model, called Multidimensional Entity Relationship (ME/R) model. This model defines a specialized entity set *dimension level* and two specialized relationship sets: a binary “*rolls-up-to*” relationship set and an n-ary *fact* relationship set. Unlike the approach of Franconi and Sattler [42], ME/R captures only the static structure of the application domain. Derived and functional information, such as applicable aggregate functions, computed and aggregated measure values, are not part of this conceptual model. Phipps and Davis [146] propose an algorithm for automatic generation of candidate ME/R schemes from the E/R schemes of the operational data sources.

UML-based methods employ object-oriented (O-O) concepts and self-extension mechanisms of this modeling standard to map the constructs of the multidimensional data model.

An O-O multidimensional model for OLAP based on metacube is proposed by Nguyen et al. [131]. UML is used for modeling both the data cubes and the metadata. The authors provide a rigorous formalization of

cubes and their elements as well as of the cube operators. A UML class diagram is used for modeling the conceptual model itself in form of a metamodel. The model is flexible and powerful with respect to handling complex hierarchical relationships within dimensions.

Trujillo et al. [174] propose an O-O multidimensional modeling (OOMD) approach that uses class diagrams to model multidimensional schemes. The approach is not restricted to flat UML class diagrams, but can benefit from the package grouping mechanism of UML to assign classes into higher-level groupings and create different levels of abstraction. The model is capable of handling advanced concepts, such as derived measures, many-to-many mappings, measure additivity properties, and multiple hierarchies.

Another approach, which customizes UML for the multidimensional realm, called YAM² (“Yet Another Multidimensional Model”), is proposed by Abelló [1]. The ultimate goal of YAM² was to enrich the multidimensional modeling with semantic relationships offered by the O-O paradigm. The proposed model comprises data structures, integrity constraints, and operations. Data structures are mapped to UML extensions using the stereotype mechanism of the latter. Since data cubes are defined as functions, a closed and complete algebra of available operations is also provided.

The work of Luján-Mora [102] fits into the framework of UML-based design approaches. The proposed method comprises various design phases and utilizes the profile mechanism of UML (i.e., extensibility for specific application domains) to define specialized profiles for each of the following data warehouse design aspects: i) *Multidimensional Modeling*, ii) *Data Mapping*, iii) *ETL*, and iv) *Database Deployment*. The UML profile for a unified conceptual multidimensional data model expresses the context of a measure in terms of its dimensions and their classification hierarchies and can handle such properties as many-to-many relationships between facts and dimensions, dimension and fact degeneration, multiple and non-strict hierarchies, etc.

An interesting proposal on adapting UML to the multidimensional paradigm can be found in [57]. At the conceptual level, it distinguishes between the language and the graphical representation. A multidimensional meta language called *MML* (Multidimensional Modeling Language) is proposed for flexible, implementation-independent modeling. *MML* provides multidimensional semantic constructs and pursues strict distinction between the metamodel, the schema and the instance. *MML* as a language can be used with different graphical notations. The author’s own *MML*-based extension of the UML, called *mUML* (multidimensional UML), defines new stereotypes to model the different types of classes and to mark the connections for building hierarchies and uses the UML extension mechanism of tagged values to model derived attributes.

Totok [173] also uses UML in his O-O modeling framework. Great flexibility of the proposed approach with respect to dynamic aspects and special cases is achieved by the use of corresponding methods for linking measures to dimensions. The model also differentiates between original and derived measures. The power of this framework consists in the ability to graphically assign each measure to the dimensions that provide valid aggregation paths in that measure’s context [8].

Among recently proposed data warehouse design methods there is one proposed by Prat et al. [149], whose framework spans all three design phases – conceptual, logical and physical. The central element of the method is a so-called *unified multidimensional metamodel* that describes the elements of the conceptual model (facts, dimensions, measures, aggregate functions, etc.). The authors also define their own graphical notation using built-in extensibility mechanisms of UML, such as stereotypes and tagged values. The defined constructs are rather similar to those of the ME/R. Besides, this model explicitly distinguishes between temporal and non-temporal dimension levels.

The third class of conceptual multidimensional data models provide proprietary frameworks.

ADAPT (Application Design for Analytical Processing Technologies) proposed by Bulos in [11] has evolved into the most prominent example in the category of revolutionary notations for data warehouse design. ADAPT offers a detailed notation consisting of a large number of graphical primitives. However, no formal definition of the constructs’ semantics is provided. The method introduces new features for multidimensional modeling, such as dimension scope and dimension context. The provided graphical framework is

limited to regular hierarchy types commonly used in OLAP tools. Apparently, the ADAPT method is rather qualified for logical modeling of MOLAP systems, and is less appropriate for the conceptual design [89].

The Nested Multidimensional Data Model (NMDM) proposed in [97] focuses on extending the multidimensional model to handle complex dimensional structures. Functionally dependent attributes within single dimensions are grouped, yielding in real orthogonal dimensions, which are easy to create and to maintain. All schema constructs in this model refer to dimensions. Cubes are considered to be a recursive nesting of all computable aggregates by classifying multidimensional objects into *primary* (dimension levels of the finest granularity) and *secondary* (upper classification levels). During the analysis phase, this technique results in nested data cubes with flexible navigation in dimension hierarchies.

Lechtenböcker presents a comprehensive data warehouse design methodology in [92]. His approach to conceptual modeling is defined as the process of obtaining “good” schemata, i.e., schemata that satisfy certain quality measures. The quality criteria are specified in terms of multidimensional normal forms. An algorithmic approach to constructing fact schemata that satisfy the Third Multidimensional Normal Form (3MNF) from the user requirement specification by analyzing functional dependencies is also proposed.

The first pragmatic scientific approach to the graphical design of multidimensional data is given in [47]. The proposed methodology is aimed at obtaining a multidimensional scheme from the operational schemes (E/R or relational ones). The methodological framework is based on the conceptual model, called Dimensional Fact Model (DFM). The representation of the multidimensional data built using DFM is called a *multidimensional scheme* and consists of a set of *fact schemes* whose basic elements are facts, dimensions and hierarchies. Due to its expressiveness, compactness and user-friendliness, DFM appears to be the most adequate model for solving the conceptual design tasks in the context of this thesis. In Section 3 we present our own extension of this model, called *X-DFM*. The extensions are aimed at making the graphical notation fully coherent with the formal model. At this point, just a brief description of the original DFM is provided, with example schemes depicted in Figure 2.14.

A fact scheme in DFM is a *quasi-tree*, i.e., a directed, acyclic, weakly connected graph, in which multiple directed paths may converge on the same vertex [47]. A fact scheme is rooted at the fact node, presented as a box labeled by the fact’s name and containing its measures. Dimension levels are represented by circle nodes, with the bottom level attached to the fact. Property attributes are shown as lines attached to the respective dimension levels. Dimension levels are connected to their upper levels, thus forming hierarchies rooted at the bottom level. Arcs connecting pairs of nodes represent many-to-one relationships between them. Directed arcs are used to resolve the ambiguity in case of multiple hierarchies. Dash-marked arcs express optional relationships. Measures are assumed to be additive. Exceptions to additivity are specified by connecting the measure to the respective dimension with a dashed line labeled with all applicable aggregate functions.

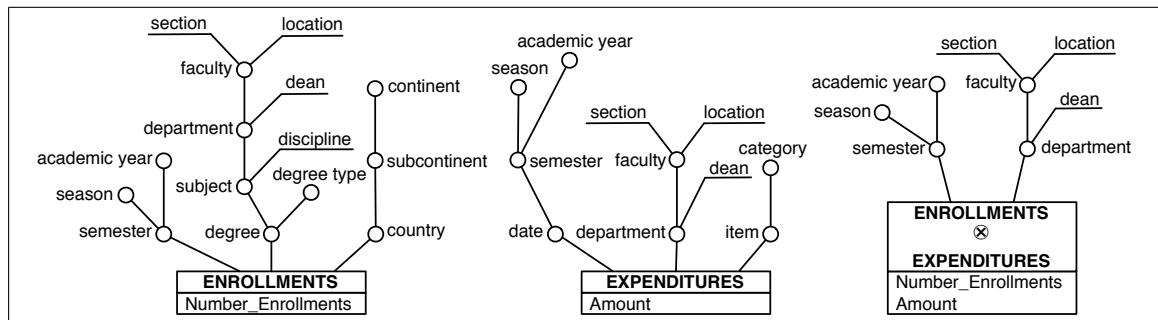


Figure 2.14: A pair of compatible fact schemes (left) and their overlap (right) modeled in DFM

DFM supports galaxies and fact constellations along with the corresponding drill-across operation by formalizing the concept of overlapping fact schemes. Fact schemes overlap if they are compatible, i.e., share at least one dimension level. Figure 2.14 illustrates the use of DFM to model the galaxy scheme from Figure 2.11. Each of the two fact schemes are modeled separately, followed by the resulting fact scheme overlap.

LOGICAL DESIGN

The goal of logical design is to detail the data as much as possible without considering physical implementation issues. However, logical models are clearly tailored towards a specific architecture. Since this thesis focuses on extending ROLAP systems, we focus our attention on the works relevant in the context of relational data warehouse design. The classical way to obtain a logical model is by means of mapping the conceptual model to logical constructs, such as relations, keys, and constraints.

The logical model may adopt the popular star schema or a less popular but more flexible snowflake schema, already discussed in Section 2.2.3. Both design options were coined by Kimball in his book [81], where he also documents his approach by presenting various design challenges and how those can be solved.

Some efforts have been made to improve Kimball's work, for instance, by employing object-oriented concepts. *O3LAP*, which is a hybrid OLAP approach aimed at combining the advantages of both ROLAP and MOLAP, is presented in [15]. The authors employ the O-O paradigm to overcome the mismatch between multidimensional operations and SQL. A mechanism called *Object-Relational View* (ORV), which is an O-O semantically-rich frontend to relational or object-relational data sources, is proposed in [49].

Mangisengi et al. [110] introduce two approaches to multidimensional modeling based on extended relational concepts. The first approach is based on the concept of *nested relations*. Nested relations (Non First Normal Form relations) [107] with relation-valued attributes and nest/unnest operators are used here to store measure values aggregated at different granularity. The second approach uses the concepts of Codd's extended relational model [29], such as *object identifiers*, a typology of *object types*, and *relationships* between different types, as well as the PATT (partitioning by attribute) operator for modeling multidimensional cubes.

Moody and Kortink [125] propose a methodology for obtaining logical schemata from E/R models and describe several variants of relational schemata, from a fully denormalized *flat* to a hybrid *star cluster* schema.

Lechtenbörger proposes a methodology of obtaining a fact constellation schema from semantic schemes as a sequence of transformation phases applied to *i*) dimensional levels, *ii*) property attributes, *iii*) contexts of validity (i.e., modeling of generalization hierarchies), and, finally, *iv*) fact schemata [92].



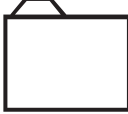
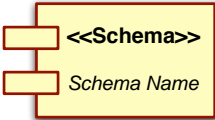

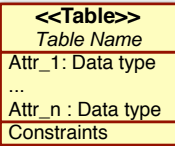

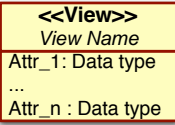

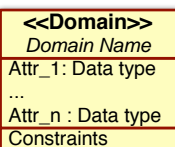
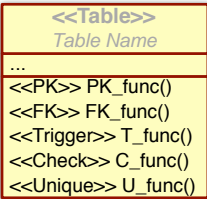
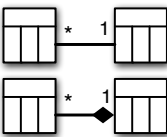
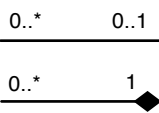
Some authors suggest that the relational data warehouse design also encompasses such phases as view materialization, vertical and horizontal fact table partitioning, and enforcing update independence [48, 92].

Luján-Mora [102] describes an approach to logical modeling based on the *UML Profile for Database Design* supplied by Rational Software Corporation [50]. In this thesis, we adopt this graphical notation for logical design. The main elements defined by this profile along with their graphical representations are summarized in Table 2.3. Icon representations can be used as a part of the base symbol or instead of the base symbol for a "collapsed" view of the element.

PHYSICAL DESIGN

The goal of the physical design is to transform the logical model obtained in the second phase into a physical schema, i.e., to refine its implementation mechanisms, such as storage structures, partitioning, access methods, and performance optimization techniques. Physical implementation of data warehouse systems is a vast field of research that is orthogonal to the scope of this thesis. We refer interested readers to [68, 90, 103] for further insights.

Table 2.3: Diagram elements of the UML Profile for Database Design

Name & Icon	Notation	Description
Database  Database name		Database is a system for data storage and controlled access to stored data. Database defines the type of the database and the data modeling constraints such as data types, stored procedures, syntax, etc.
Schema  Schema name		Schema is the basic unit of organizing tables in a database. Schemata are assigned to a database component at the next level of detail.
Table  Table name		Table represents a relational table in a database, defined as a set of data records of the same structure. A table may be associated to a particular schema.
View  View name		View is a virtual table. It is structured exactly like a table, with the only difference that the physical source of the data is in other table(s).
Domain  Domain name		Domain is a mechanism used to create user-defined data types that can be applied to columns across multiple tables.
Constraints <<PK>> <<FK>> <<Trigger>> <<Check>> <<Unique>>		Constraint is a rule applied to a column and/or table. PK constraint defines a primary key of a table. FK constraint is a foreign key implementing a relationship to another table. Trigger is an activity automatically executed to database consistency. Check constraint verifies whether the values of a column remain within a certain range. Unique assures that all values of a column are distinct.
Relationships 		Relationships model foreign key dependencies between tables. A relationship is non-identifying if the tables can exist independent of each other. A relationship is identifying when the child table cannot exist without the parent table.

2.3 Visual Analysis and Exploration

Analysts query multidimensional data interactively using visual interfaces, called *OLAP tools*, which build the Presentation Layer of the data warehouse system architecture shown in Figure 2.2.

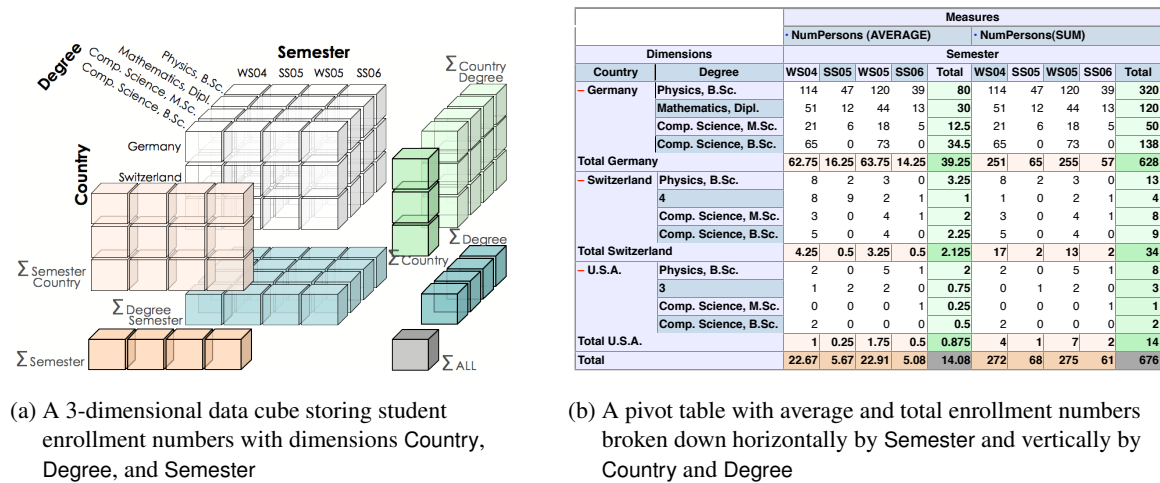


Figure 2.15: Exploring a multidimensional data cube with a pivot table

A traditional interface for analyzing OLAP data is a *pivot table*, or *cross tab*, which is a 2-dimensional spreadsheet with associated totals and subtotals. A pivot table is populated with data by specifying the measure(s) of interest and selecting dimensions to serve as vertical (and, optionally, horizontal) axes for summarizing the measures. The power of this presentation comes from its ability to summarize detailed data along various dimensions and arrange the aggregates computed at different granularity levels into a single view preserving inclusion relationships between the aggregates. An additional advantage is the ability to “nest” multiple dimensions within an axis. Figure 2.15 exemplifies the idea of “unfolding” the sample 3-dimensional student enrollments cube (left) into a pivot table (right). For convenience, table cells at each granularity level are filled with the same background color as the corresponding cube’s slices.

The pivot table technique is a powerful straightforward presentation of multidimensional aggregates, however, it is heavily criticized by researchers and practitioners for disgraceful handling of large datasets and for its inefficiency for non-trivial analytical tasks, such as recognizing patterns, discovering trends, identifying outliers, etc. [39, 95, 164]. Advanced OLAP tools overcome the limits of the cross tab interface by offering a multitude of powerful visual alternatives for retrieving, displaying, and interactively exploring the data. Visualization has the power to save time and reduce errors in analytical reasoning by utilizing the phenomenal abilities of the human vision system to recognize patterns [55]. Advances in visualization technologies enable use of human visual abilities to solve analytical tasks. Furthermore, cognitive fit theory shows that decision making is improved when the information representation matches the problem-solving task [181].

2.3.1 Visual OLAP as an Emerging Trend

The first generation of OLAP tools was tailored towards the needs of routine reporting by providing a *managed query environment* for navigating within a set of pre-defined queries [23]. With recent achievements in the information technology, the scope of analytical tasks has expanded far beyond interactive report generations. Comprehensive analysis includes examining the data from multiple perspectives, extracting useful information, verifying hypotheses, recognizing trends, revealing patterns, and discovering new knowledge from arbitrarily large and complex data sets. Conventional frontends display a number of deficiencies, such

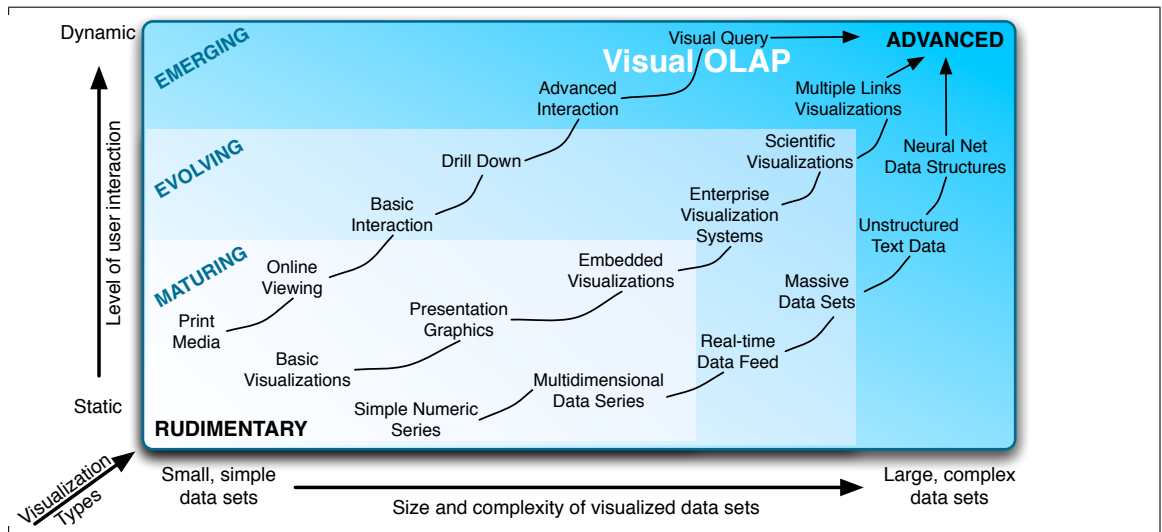


Figure 2.16: Trends in data visualization for business analysis (adopted and modified from [158])

as ineffective and inefficient data presentation, cumbersome usability, and poor functionality. These deficiencies have triggered the emergence of fundamentally new ways of interacting with multidimensional data.

Next-generation frontends achieve a new quality of analysis by unlocking the synergy between the performance-oriented *BI* techniques and the achievements in the areas of *Information Visualization*, *Human-Computer-Interaction*, and *Visual Analytics*. To efficiently analyze huge data volumes and uncover the “hidden gems” therein, novel tools enable free-wheeling and unconstrained data exploration allowing users to navigate to the desired view, experiment with various layout options, thus, supporting the process of incrementally refining a question into an answer. We refer to this flexible data exploration approach as *Visual OLAP*:

“Visual OLAP is an umbrella term encompassing a new generation of OLAP end-user tools for interactive ad hoc exploration of large volumes of multidimensional data by providing a comprehensive framework of advanced visualization techniques for representing the retrieved data set along with a powerful navigation and interaction scheme for specifying, refining, and manipulating the subset of interest.” [160].

The key distinction between the traditional and the novel OLAP tools is the role of visualization: the former use visualization merely for expressive *presentation* of the data, whereas the latter employ visualization as a *method* of interactive ad hoc analysis. In addition to conventional OLAP operations, visual OLAP supports further interaction techniques, such as zooming and panning, filtering, brushing, collapsing, etc.

Continuous efforts are put into providing new approaches to visual exploration for OLAP. Russom [158] summarizes the trends in business visualization as a progression from rudimentary data visualization to advanced forms and recognizes three life-cycle stages of visualization techniques, such as maturing, evolving, and emerging, as depicted in Figure 2.16. Within this classification, visual OLAP clearly fits into the emerging techniques for advanced interaction and visual querying.

2.3.2 Visual Exploration Framework

Figure 2.17 illustrates the data exploration process, also denoted *knowledge crystallization*, adopted from [20] and slightly modified for matching the context of OLAP. In this cycle, visualization clearly plays the key role in providing insight into the data and, thus, solving the task at hand. Also notice that searching for a solution may evolve in multiple iteration cycles.

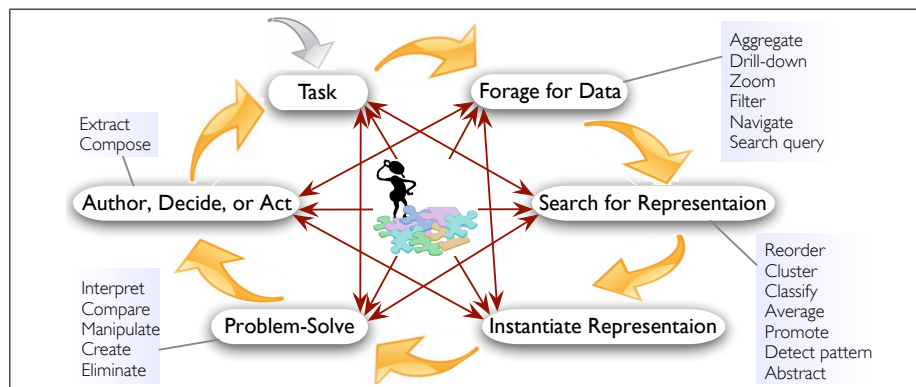


Figure 2.17: Data exploration cycle

A general approach to enabling the above flexible data exploration scheme is to implement a comprehensive framework, in which the users can browse through the data and interactively generate satisfactory visual presentations using “point-and-click” and “drag-and-drop” interactions. The overall query specification cycle includes *i*) selecting a data source of interest, *i*) choosing a visualization technique (e.g., a scatterplot or a bar-chart), and *i*) mapping various data attributes to that technique’s structural elements (e.g., horizontal or vertical axis) as well as to other visual attributes, such as color, shape, and size. The main elements of the framework are a data navigation structure, or “data browser”, for visual querying of data sources, a taxonomy of available visual layouts and their attributes, and a toolkit of interaction techniques for dynamic query refinement and visual presentation of the output. A unified framework is obtained by designing an abstraction layer for each element and providing mapping routines (e.g., navigation events to database queries, query results to a visual layout, etc.) that implement the interaction between different layers. Figure 2.18 shows an example of an advanced frontend tool for visual data analysis provided by Tableau Software [169]. Data navigation on the left-hand side enumerates available dimensions and measures. The central area is occupied by the visual display of the current query results. Additional windows and menu provide further options for refining the dataset itself (e.g., filter) and adjusting the visualization (e.g., color assignment, labeling, resizing).

DATA NAVIGATION SCHEME

Visual OLAP disburdens the end-user from composing queries in “raw” database syntax (e.g., SQL or MDX). Instead, queries are specified visually (i.e., by means of using a computer mouse). Multidimensional data is represented as a browsable structure. Visual interface does not trade off advanced functionality for simplicity, it rather facilitates the process of specifying ad hoc queries of arbitrary complexity.

A common data browsing paradigm is that of a navigation tree, i.e., as a recursive nesting of element nodes. The nodes in OLAP navigation scheme may be of types *database*, *schema*, *table (cube)*, *dimension*, *classification level*, and *measure*. In simplified configurations, the navigation may be limited to single data cubes and, thus, consist solely of dimension and measure attributes of a selected cube.

Data cubes are navigation object consisting of measures and dimensions. A dimension is represented either as a lattice of its aggregation levels or directly by the data hierarchy. Logical OLAP operations are incorporated into a visual framework in the form of navigation events. For example, a drill-down is performed by dragging the respective dimension category node into the visualization.

Mapping of a multidimensional structure to the navigation scheme, translation of navigation events into

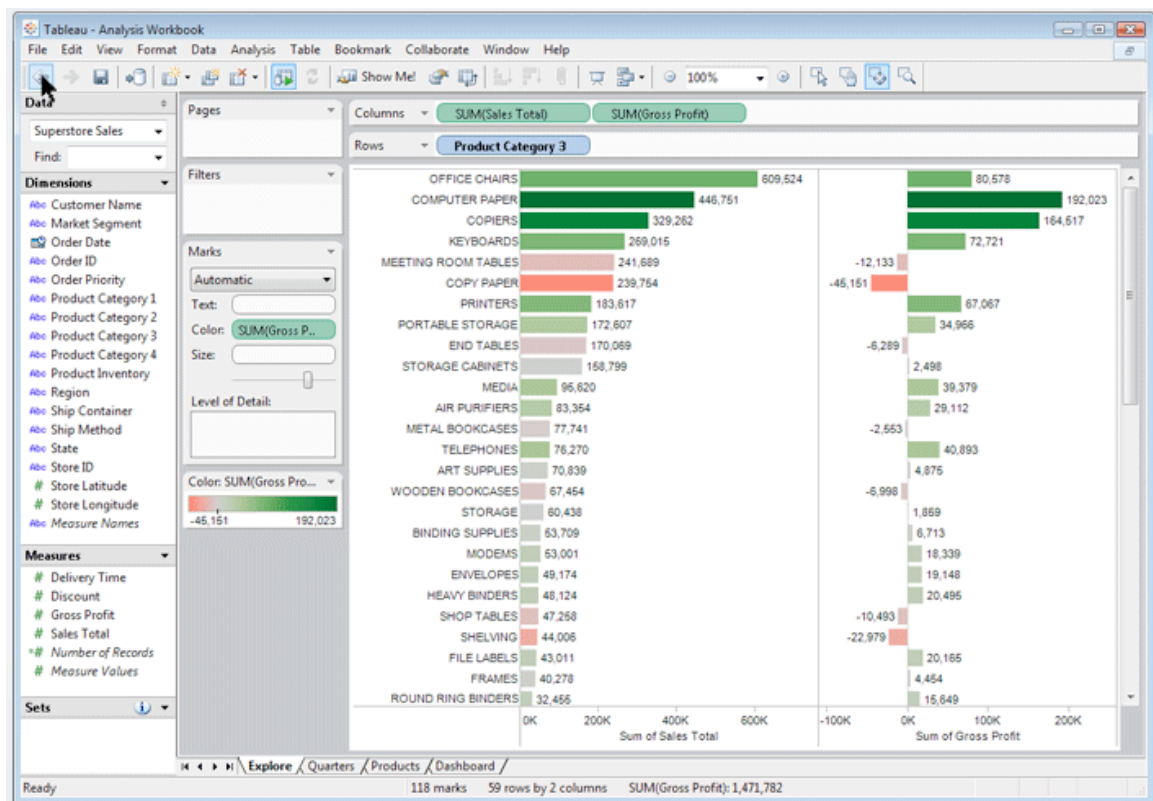


Figure 2.18: Tableau Software as a powerful frontend for visual analysis

database queries and mapping the query output to a visual layout rely on the metadata of the underlying data warehouse system. Metadata describes the structure of the cubes and their dimensions, measures and their additivity. In an advanced user interface, the analysts are able to define new measures in addition to the ones configured via the metadata. New measures may be obtained by applying a different aggregate function (e.g., average, variance, count) or a more complex formula over a single or multiple data fields (e.g., computing a ratio between two aggregated values).

VISUALIZATION INTERFACE

The task of selecting a proper visualization technique for solving a particular problem is by far not trivial as various visual representations (metaphors) may be not only task-dependent, but also domain-dependent. Successful visual OLAP frameworks need to be based on a comprehensive taxonomy of domains, tasks, and visualizations. The problem of assisting the analyst in identifying an appropriate visualization technique for a specific task is a still unsolved issue in state-of-the-art OLAP tools. Typically, a user has to find an appropriate solution manually by experimenting with different layout options.

A common approach to visualization in OLAP application relies on a set of templates, wizards, widgets, and a selection of visual formats. Hanrahan et. al [55] argue, however, that an open set of questions cannot be addressed by a limited set of techniques, and choose a fundamentally different approach for their visual

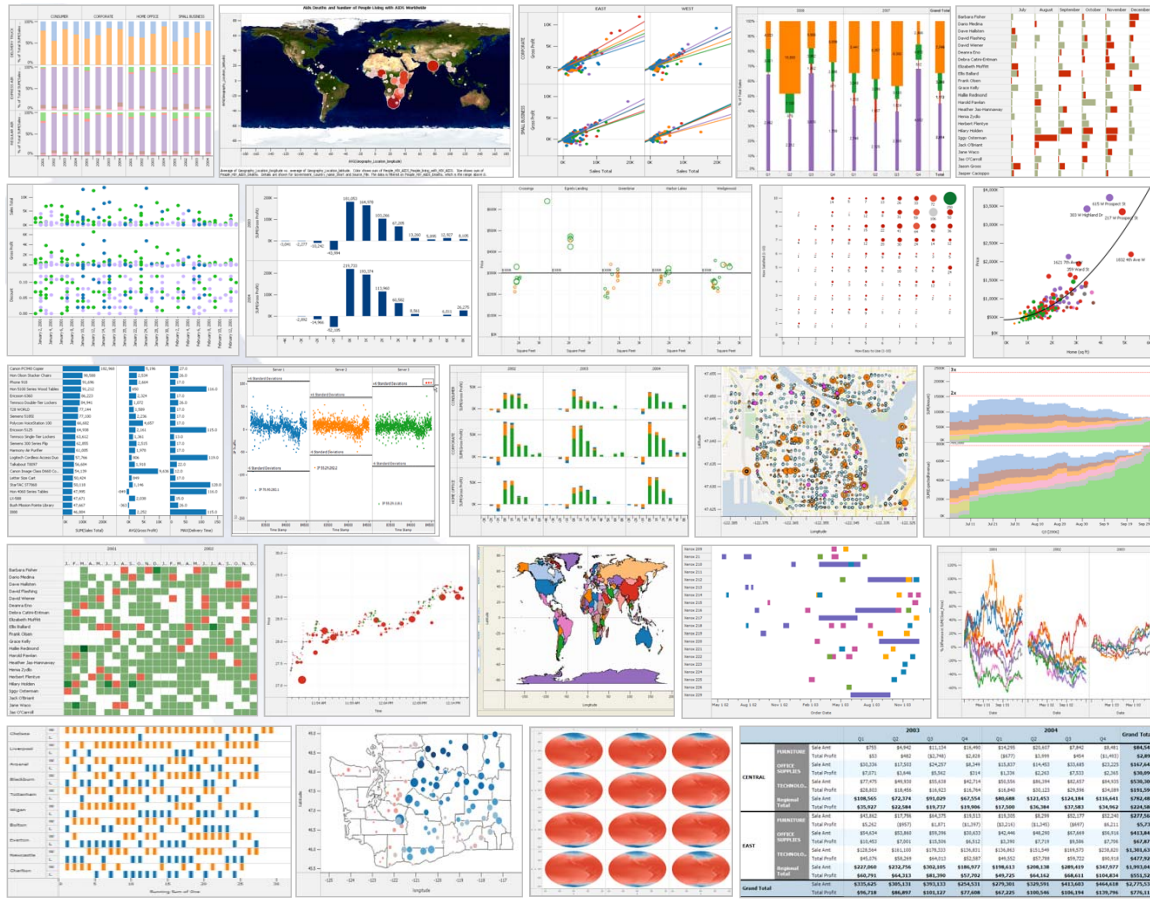


Figure 2.19: Examples of sophisticated visualizations generated by VizQL statements (permission granted by Tableau Software, Inc.)

analysis tool commercialized as Tableau Software [169]: a declarative visual query language VizQLTM offers high expressiveness and composability allowing users to create their own visual presentations by combining various visual components. Figure 2.19 illustrates the visualization approach of Tableau by showing a subset of sophisticated visual presentations created using simple VizQL statements and not relying on any pre-defined template layout. In a more recent work [106], the same authors propose a pioneering solution to automating the visual presentation based on the user experience in the Tableau system. Other prominent examples of advanced visual systems based on research findings are Advizor by Visual Insights [39] and MagnaView Explorer by MagnaView [185].

Chapter 3

Extending the Multidimensional Data Model

THIS CHAPTER IS DEDICATED to the conceptual data warehouse design for advanced applications. We revise the state of the art of the multidimensional data model, identify the major “bottle-necks” of the latter, and formulate a set of requirements towards the extended model. The proposed conceptual modeling framework consists of a formal model and an appropriate graphical notation. This chapter focuses on defining the core elements of the extended model. Formal definitions are accompanied by the introduction of the corresponding graphical constructs and are supplied with illustrative examples.

Contents

3.1	State of the Art of Multidimensional Modeling	35
3.1.1	Fundamental Constraints of the Multidimensional Data Model	36
3.1.2	Related Work	37
3.2	Requirements of Comprehensive Multidimensional Analysis	38
3.3	Conceptual Model: Graphical Notation and Formalization	41
3.3.1	X-DFM as the Graphical Modeling Notation	41
3.3.2	Formalization	47
3.4	Terminology and Definitions	48
3.4.1	Unified Multidimensional Space	48
3.4.2	Facts and Dimensions	49

3.1 State of the Art of Multidimensional Modeling

The claimed universality of OLAP bears on the concept of “analyzability”: the data should be homogenized, integrated, and preprocessed to enable efficient and goal-oriented analysis [8]. This claim of the universal applicability finds sufficient confirmation in practice: in the last years, the deployment of data warehouses has reached out for a variety of non-business domains and non-conventional applications, such as government, academia, life sciences, bio-informatics, education, research, medicine, etc.

Even though data warehousing is an established and widely adopted practice in the modern information technology platform, there exist numerous open research issues in this area. Many of those issues arise due to the attempts to apply business performance oriented OLAP techniques to non-conventional application scenarios. The causes of deficiencies and failures are manifold, from the underlying conceptual model to the frontend “bottle-necks”. At this stage, we investigate the types of constraints that lie at the very heart of OLAP, i.e., the ones that are imposed by the conceptual model itself.

3.1.1 Fundamental Constraints of the Multidimensional Data Model

Any attempt to extend an existing model should be preceded by the realization of that model’s essential assumptions and constraints. We define a constraint to be *essential*, or *inherent*, if it is fundamental for the definition and basic functionality of the model; violation of such a constraint results in the malfunction of the entire approach. The conventional multidimensional data model along with the associated OLAP operators rely on such constraints as prohibition of many-to-many relationships and NULL values, fact scheme homogeneity, uniform grain within a fact type, and summarizability for all dimension hierarchies. Further fundamental issues that aggravate straightforward applicability of the OLAP technology to complex data are the following ones:

- ◆ *“Roll-up” as the only relationship type.* This relationship expresses inclusion between facts and dimensions as well as between hierarchy levels. The model provides no mechanism to explicitly model any other relationships.
- ◆ *Many-to-many relationships must be mapped to facts.* It is one of the so-called Kimball’s Laws [81], which implies prohibition of non-strict dimension hierarchies and many-to-many relationships between facts and dimensions.
- ◆ *Fact homogeneity* implies that all fact entries fully adhere to the fact’s scheme, i.e., have the same dimensional characteristics and the same grain within each dimension.
- ◆ *Homogeneous aggregation* requires that all entries within the same fact type roll up along the same aggregation paths. This requirement implies prohibition of partial roll-up relationships.
- ◆ *Prohibition of NULL values* is an important guarantee for correct aggregation behavior.
- ◆ *Duality of facts and dimensions* forces to distinguish between fact schemes and dimension schemes and to statically assign each characteristic to a particular scheme.
- ◆ *Absence of object-oriented features*, such as generalization or inheritance.
- ◆ *Isolation of fact schemes* means that each scheme is modeled in its own multidimensional space, not related to other schemes. As a result, identical or semantically related attributes are maintained redundantly for each fact scheme. Besides, scheme isolation aggravates straightforward application of the drill-across operation.
- ◆ *Summarizability* requires distributive aggregate functions and dimension hierarchy values, or, informally, that *i)* facts map directly to the lowest-level dimension values and to only one value per dimension, and *ii)* dimensional hierarchies are balanced trees [98].
- ◆ *Duality of measure and dimension roles.* Measures reflect the focus of the analysis and, therefore, they should be known at design time and should be explicitly specified in the fact scheme.
- ◆ *Duality of category and property roles.* A dimension category consists of a single category attribute and may have further attributes, called *properties*. Property attributes may not be used as aggregation levels, even though the relationship between a category attribute and its property is equivalent to roll-up.

3.1.2 Related Work

Deficiencies of the original multidimensional data model and proposals of extended models have become an active data warehousing research issue in the last decade. The necessity to develop novel concepts was emphasized [190], and a series of extensions have been proposed in the literature. Most of the proposals were coined by a set of requirements drawn from specific application scenarios and, thus, do not claim to be ultimate or universal. Disclosure of novel applications continues to impose new modeling challenges and will undoubtedly continue to encourage further contributions.

Pedersen et al. [145] formulated 11 requirements of comprehensive data analysis and evaluated 14 state-of-the-art data models for data warehousing from both the research community and the commercial systems against those requirements. As none of the investigated models appeared to provide more than 6 of the 11 features, the authors proposed an extended model for capturing and querying complex multidimensional data according to the defined requirements. The evaluation criteria specified in [145] were not claimed to be universal as those were inspired by a specific case study from the healthcare domain. Nevertheless, the proposed extended model, supporting such features as non-summarizable hierarchies, many-to-many relationships between facts and dimensions, handling temporal changes and imprecision, remains one of the most powerful among the existing multidimensional models.

A similar attempt to classify and evaluate existing multidimensional models is presented in [2]. However, the authors defined two orthogonal sets of classification criteria, namely, according to the kind of constructs/concepts they provide and according to the design phase at which they are employed. Another assessment of conceptual models is provided in [104], in which 6 prominent multidimensional models are evaluated against an exhaustive set of requirements regarding facts, dimensions, measures, operators etc. Based on the stated requirements, the authors propose a UML-based multidimensional modeling toolkit that uses self-extensibility mechanisms of UML to enable adequate modeling support for a variety of multidimensional properties. Trujillo et al. propose an O-O multidimensional modeling (OOMD) approach that provides a theoretical foundation for the use of object-oriented features in data warehousing and OLAP applications [174]. This approach introduces a set of minimal constraints and extensions to UML for representing multidimensional modeling properties for these applications.

Major research efforts in the field of multidimensional modeling are focused on handling complex dimensions [61, 109, 118, 132, 145], which is but comprehensible: traditional models appear rather rigid by enforcing dimension hierarchies to be homogeneous, complete, strict, and balanced. As a result, irregular data hierarchies encountered in many real-world applications appear inadequate as dimensions of analysis. In a survey on open issues in multidimensional modeling [63], Hümmer et al. identified unbalanced and irregular hierarchies and missing data as the most pressing challenges of dimensional modeling.

Hurtado and Mendelson [60] proposed integrity constraints for inferring summarizability in heterogeneous dimension hierarchies and defined a formal framework for constraint-conform hierarchy modeling [61]. An approach to modeling dimension hierarchies with no enforcement of balancedness or homogeneity along with the corresponding SQL extensions, called $\text{SQL}(\mathcal{H})$, is described in [70]. Niemi et al. [132] analyzed unbalanced and ragged data trees and demonstrated how dependency information can assist in designing summarizable hierarchies. Lehner et al. [96] relaxed the condition of summarizability to enable modeling of generalization hierarchies. A remarkable contribution to the conceptual design was made by Malinowski and Zimányi who presented a comprehensive classification of dimensional hierarchies including those not addressed by current OLAP systems in [108] and formalized their conceptual model and its mapping to the relational schema in [109].

To the best of our knowledge, most of the extensions proposed in the above contributions have not been implemented by any existing data warehouse systems. In a previous work [182], we presented a prototypical analysis interface capable of supporting a subset of irregular dimension hierarchies and allowing interactive

data exploration using hierarchical visualization techniques. A more recent work [118] builds upon the classification framework of [109] and extends it by providing a more formal and comprehensive categorization of dimension hierarchy types. All enumerated classes are inspected for summarizability and a two-phase transformation algorithm for deriving a logical schema is proposed. As a proof of concept, all introduced model extensions were implemented in a visual interface with a schema-based dimensional navigation structure for exploring data cubes along complex dimension hierarchies.

Another branch of research in the multidimensional modeling focuses on challenges other than complex dimensions. These other issues address non-conventional requirements concerning facts, measures, fact-dimension mappings, and multi-fact schemes. The extended model of Pedersen et al. [145] accounts for such features as symmetric treatment of dimensions and measures, many-to-many relationships between facts and dimensions, awareness of the aggregation semantics, and variable granularity of facts. Abelló et al. [3] clarify some concepts related to multidimensionality in general and fact modeling in particular. The authors demonstrate the convertibility of fact and dimension roles in multi-fact schemes.

Ravat et al. [153] demonstrated how OLAP can be applied for analyzing semi-structured data, such as XML documents. As this data type is non-additive and non-numeric, the whole analysis framework needs to be adapted. The authors proposed a conceptual model based on a “factless multi-dimension” representation and define a set of multidimensional operations and aggregate functions specialized on this type of analysis.

In [114] we applied the data warehousing approach to business process analysis. The requirement to store the original process execution data rather than pre-defined performance measurements helped us identify new types of facts, factual and fact-dimensional relationships and aggregation behaviors. Besides, absence of predefined measures in the scheme raises the issue of enabling interactive measure specification.

3.2 Requirements of Comprehensive Multidimensional Analysis

A multitude of multidimensional models proposed in recent years is a result of specifying different sets of requirements a model has to meet. In this section, we integrate diverse requirements and properties defined by various authors with respect to comprehensive multidimensional analysis over complex data into a unified framework. This framework will serve as a reference for designing an extended data model.

The requirements can be subdivided into *i) static* properties dealing with the structuring of the multidimensional data space and *dynamic* properties dealing with the supported analysis tasks.

A set of major static multidimensional modeling properties, proposed in the research literature [2, 104, 145] including our own works [113, 114, 115, 116, 118, 120], can be summarized as follows:

1. *Explicit separation of cube structure and its contents.* The structure of a data cube is modeled as a fact-dimensional scheme. The actual content is crucial for refining the scheme in order to identify irregular hierarchies, partial roll-ups, etc.
2. *Facts with no measures.* The terms “fact” and “measure” are often used interchangeably in the data warehousing literature. However, some applications deal with multidimensional data structures without explicitly defined measures. Besides, according to one of Kimball’s laws, any many-to-many relationship should be modeled as fact [81]. Therefore, it is necessary to refine the concept of a “fact” and enable modeling of facts with no measures.
3. *Complex measures.* The model should support composite and derived measures as well as the specification of each measure’s additivity, i.e., aggregation semantics.
4. *Complex facts.* The model should be capable of handling deviating behavior within facts, such as heterogeneity, variable granularity, and missing values.

5. *Multi-fact schemes.* Some application scenarios require the data to be modeled as multiple related fact schemes in order to preserve the actual relationships in the data. Inter-factual relationships take the form of shared dimensions in a unified multidimensional space.
6. *Fully and partially shared dimensions.* Data cubes may be compatible to one another at a non-bottom granularity level. This happens when their schemes have at least one pair of partially shared dimensions, i.e., converging or overlapping at a category, non-bottom for at least one of them. To recognize partial sharing,
7. it is imperative to provide a methodology for identifying scheme overlap.
8. *Multiple roles of dimension categories.* In multi-fact schemes, the same dimension or category may be used in multiple roles (e.g., date dimension may be used as order date and payment date characteristics of a fact). Therefore, it is imperative to distinguish between the categories and their roles.
9. *Many-to-many fact-dimensional relationships.* Many-to-many mappings between facts and dimensions are common in practice and, therefore, should be manageable by the model.
10. *Explicit hierarchies in dimensions.* Hierarchies should be presented explicitly in the multidimensional scheme as the former determine valid aggregation paths. Furthermore, the model has to distinguish between dimension level attributes and property attributes belonging to a particular level.
11. *Complete hierarchies.* In a complete hierarchy, all child-level members fully roll-up to one parent-level and the extension of the parent-level consists of those child members only [104]. The model should provide constructs to specify the completeness, i.e., non-expandability, of a hierarchy.
12. *Multiple hierarchies.* A dimension can have multiple aggregation paths, which may be mutually exclusive or compatible and which may or may not converge at some upper level.
13. *Distinction between alternative and parallel hierarchies.* Multiple alternative hierarchies refer to the same hierarchical property and thus represent mutually exclusive aggregation paths. Parallel hierarchies are based on mutually independent hierarchical properties and may be used in combination as aggregation criteria.
14. *Complex dimensions.* To support complex dimensions, the model should capture the causes of the complexity, such as non-covering, non-onto, and non-strict mappings, heterogeneity, etc.
15. *Partial roll-up behavior.* Roll-up relationship between a fact and a dimension or between dimension categories may be full (each member participates in the relationship) or partial (members are allowed not to participate in the relationship). Partial roll-up may be a result of optionality, heterogeneity, or specialization. The model should distinguish between various kinds of roll-up relationships.
16. *Totally ordered hierarchies.* A dimension hierarchy is normally defined in terms of partial ordering (parent-child relationships within pairs of members). However, in some hierarchies, members of the same hierarchy level may have to be ordered to reflect some semantics and enable default sorting according to this ordering.

As for the dynamic properties of the extended multidimensional model, we identify the following ones:

1. *Symmetric treatment of facts and dimensions.* In multi-fact schemes, the duality of fact and dimension roles fades since one fact may act as a dimension of another fact, or a dimension may be turned into a fact of a specific query.
2. *Symmetric treatment of measure and dimension attributes.* Any attribute within a fact scheme may be converged into a measure of a specific query.

3. *Measure used as dimension*. Some queries may need to use some measure attribute as a dimension w.r.t. another measure within the same fact scheme.
4. *Drill-across*. Drill-across is a logical operator for constructing a multicube by joining a set of multiple related cubes, projected to the subset of their common dimensional characteristics, in order to explore their measures in parallel or to derive new measures.
5. *Ad hoc measure derivation*. Measures, not available in the static model, can be added at query time by specifying their derivation formulas.
6. *Ad hoc dimension derivation*. Dimensions, derivable from the existing one but not included into the original scheme, can be added at query time by specifying their derivation formulas.
7. *Ad hoc hierarchy specification*. The user should be able to manipulate the existing hierarchies (e.g., merge multiple categories into one) as well as to arrange dimensional values into a new hierarchy.
8. *Resolution of many-to-many mappings*. In the presence of non-strict hierarchies, users should be prompted to resolve multi-parent relationships for correct aggregation (drill-aside operation).

Table 3.1 summarizes the above mentioned static and dynamic properties grouping them according to the construct of the conceptual model, to which they belong.

Table 3.1: Overview of the desirable multidimensional properties

Concept / Construct	Static property	Dynamic property
<i>Fact galaxy</i>	<ul style="list-style-type: none"> ▷ Fully and partially shared dimensions ▷ Dimension category in multiple roles ▷ Fact in dimension role 	<ul style="list-style-type: none"> ▷ DRILL-ACROSS operation ▷ Fact overlap scheme
<i>Fact</i>	<ul style="list-style-type: none"> ▷ Non-measurable facts ▷ Heterogeneous facts ▷ Variably grained facts ▷ Missing values 	<ul style="list-style-type: none"> ▷ Dimension treated as a fact ▷ Derived fact
<i>Measure</i>	<ul style="list-style-type: none"> ▷ Complex measures ▷ Derived measures ▷ Aggregation semantics 	<ul style="list-style-type: none"> ▷ Ad hoc derived measures ▷ Combined measures ▷ Measure from a dimension (PUSH)
<i>Dimension</i>	<ul style="list-style-type: none"> ▷ Explicit hierarchies ▷ Multiple hierarchies ▷ Parallel vs. alternative hierarchies ▷ Complex/irregular hierarchies ▷ Heterogeneous hierarchies ▷ Complete hierarchies ▷ Derived categories ▷ Totally ordered hierarchies 	<ul style="list-style-type: none"> ▷ Ad hoc defined categories ▷ Ad hoc defined dimensions ▷ Dimension from a measure (PULL) ▷ Ad hoc hierarchies
<i>Roll-up relationship</i>	<ul style="list-style-type: none"> ▷ Partial roll-up ▷ Optional roll-up ▷ Generalization/specialization ▷ Many-to-many mappings 	<ul style="list-style-type: none"> ▷ Resolution of non-strict roll-ups

3.3 Conceptual Model: Graphical Notation and Formalization

The aim of the conceptual modeling is to capture relevant data and relationships in the application domain in a semantically rich and implementation-independent fashion. Two major components of the semantic multidimensional model are the formalization and the graphical notation. A formal model is necessary for providing a rigorous theoretical framework in form of definitions, constraints, classifications, lemmas, and proofs, whereas the role of a graphical notation is to support the practice of user-friendly design of conceptual schemata. Most of the multidimensional existing models focus either on the formalism or on the graphical toolkit, but not both. Formal models usually employ some existing graphical notation (e.g., ER, UML or their variants) or do not employ any.

In our opinion, the graphical notation should be fully aligned with the formal model in order to correctly capture the semantics of the latter. A prominent contribution in providing a comprehensive methodology for conceptual data warehouse design is Dimensional Fact Model (DFM) of Golfarelli et al. [47], already introduced and briefly explained in Section 2.2.4. DFM is based on a pragmatic scientific approach, in which the graphical framework emanates from the formal conceptual framework. Wide acceptance of this model in research and practice can be attributed to its simplicity, elegance, and expressiveness. As we proceed with extending the conceptual model, we will face the necessity to revise the original DFM notation and extend it to account for novel concepts.

Our model emerged gradually as we accumulated practical experiences of dealing with various application domains. Earlier stages of the resulting framework can be found in a series of previous works [113, 116, 118, 120]. Fundamentally, our formalization relies on the extended multidimensional models proposed by Pedersen, Jensen et al. [73, 145] as well as on the above mentioned DFM framework. In the remaining two sections of this chapter, we introduce our graphical notation and the underlying formalization, respectively.

3.3.1 X-DFM as the Graphical Modeling Notation

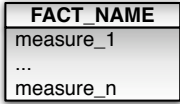
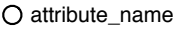
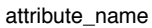
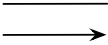
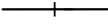
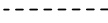
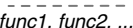
We require the graphical model to be as discriminative as the formal one, i.e., the former is not allowed to map different concepts of the latter to the same graphical element. At the same time, we avoid over-discrimination of the graphical model, i.e., providing multiple ways of expressing the same semantic element. Semantic richness is preferred over minimality to ensure that the graphical model is self-sufficient for extracting the metadata about the multidimensional structure.

Throughout the thesis, we undertake numerous extensions of the original DFM with the objective of insuring its coherence with the formal definitions of the respective concepts. The resulting notation is denoted X-DFM (Extended Dimensional Fact Model) to be distinguished from the original DFM.

THE ORIGINAL DFM NOTATION

In DFM, data cubes are presented as fact schemes, which show the fact's measures and dimension hierarchies as a structured quasi-tree rooted at the fact node. Dimension hierarchies are shown as directed graphs of aggregation paths with hierarchy levels as nodes and roll-up relationships between them as edges. Table 3.2 gives an overview of the provided notation elements. DFM identifies three types of nodes: *i*) facts, *ii*) dimension attributes (levels), and *iii*) non-dimension attributes (properties). A fact node is shown as a box bearing the fact's name and containing its measures. Dimension attributes are represented by labeled circles while property attributes are terminal nodes represented by labeled lines. Non-dimension attributes may be associated either with a fact or with a dimension attribute.

Table 3.2: Graphical constructs of the original DFM notation

Element	Description
	A fact is a box-shaped node labeled by the fact's name and containing the set of the fact's measures .
	A dimension attribute , or level, is a circle-shaped node labeled by the attribute's name.
	A non-dimension attribute , or property, is an additional characteristic of a dimension attribute.
	A roll-up relationship is a many-to-one relationship between a pair of attributes. A directed edge (the bottom one) is used to resolve ambiguities in case of multiple incoming roll-up relationships of the same attribute.
	An optional relationship is a many-to-one relationship with partial participation.
	Non-aggregability of measure M along dimension D means inapplicability of any aggregate function along that dimension's hierarchy.
	Non-additivity of measure M along dimension D results in inapplicability of the SUM operator along that dimension's hierarchy. Allowed aggregate functions (operators) are listed as the edge's labels.

An arc connecting a pair of nodes represents a many-to-one, or a *roll-up*, relationship between two nodes. Such a relationship may exist between *i*) a fact and a dimension level, *ii*) dimension levels, and *iii*) between a dimension level and a property attribute. DFM uses undirected arcs to represent directed edges as the direction can be unambiguously derived from the position of the fact node: an edge between two nodes is outgoing with respect to the node that has a shorter path to the root. The only ambiguity arises in the presence of multiple alternative or parallel hierarchies and related partial roll-up relationships, as in those cases distinct paths converge at the same dimension level. To resolve such cycles, DFM uses directed arcs. Optional relationships between nodes are marked with a dash crossing the corresponding arc. Such relationships may occur between any types of nodes and imply a partial roll-up. Whenever a measure is non-additive along a dimension, a dashed line is drawn between the two. If there exist further applicable aggregate functions (e.g., COUNT, MAX, etc.) those are listed as labels of the respective non-additivity edge.

Figure 3.1 demonstrates the usage of DFM for modeling a data cube that captures purchasing transactions within a university. The resulting 5-dimensional fact scheme PURCHASE describes purchasing transaction records in terms of two measures – amount and number of items – characterized by dimensions funding, product, project, date, and unit. Shaded bubbles enclosing each dimension's hierarchy are used solely for facilitating the perception and are not part of DFM.

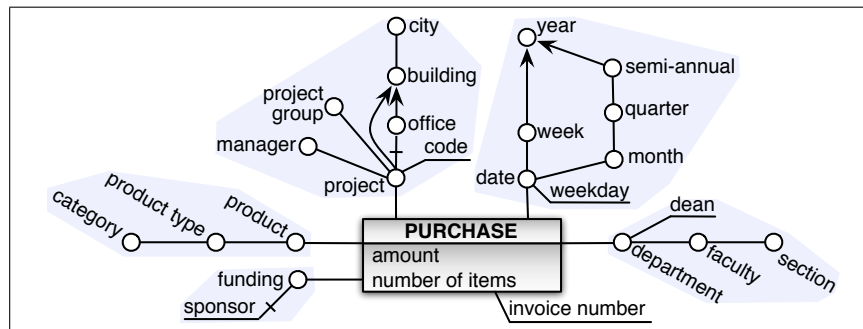


Figure 3.1: A 5-dimensional fact scheme PURCHASE in the original DFM notation

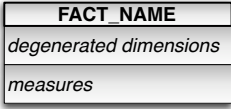
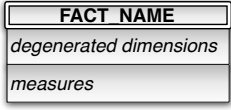
FROM DFM TO X-DFM

With respect to the requirements of the extended multidimensional data model and the fundamental definitions provided in Section 2.2, DFM displays a number of deficiencies, which can be summarized as follows:

- ◆ Facts are allowed to have non-dimension attributes, as is the case with invoice number in Figure 3.1. However, by definition, facts are composed solely of measures and dimensions.
- ◆ There is no construct for modeling many-to-many and one-to-one relationships between elements.
- ◆ Directed (i.e., many-to-one) relationships between the nodes are shown by non-directed edges. In our opinion, that is somewhat counter-intuitive for the interpretation of the scheme. Besides, DFM uses directed and undirected edges as alternative notations for the same type of relationship (many-to-one), thereby producing unnecessary differentiation.
- ◆ There is no distinction between hierarchical and non-hierarchical relationships: a roll-up relationship between levels does not visually differ from an associating a level with a non-dimension attribute.
- ◆ There is no distinction between optional properties and partial roll-up relationships.
- ◆ There is no construct for modeling heterogeneous roll-up relationships.
- ◆ There is no way to specify a totally ordered category or hierarchy.
- ◆ Top-level dimension categories are not shown in the scheme.
- ◆ DFM does not differentiate between alternative and parallel hierarchies in a dimension. However, the distinction is crucial for automatic recognition of valid aggregation paths. Multiple alternative hierarchies like the ones given by dimension levels week and month offer alternative, i.e., mutually exclusive, aggregation paths for date. Parallel hierarchies like the ones given by manager and project group are defined on independent characteristics within project dimension, and thus, can be used as aggregation axes in arbitrary order. Parallel hierarchies behave like different dimensions due to their orthogonality.
- ◆ Measure attributes inside the fact are presented as text labels. However, each attribute is a node of the scheme and, therefore, should be identifiable as nodes.
- ◆ There is no concept for modeling derived elements (facts, dimensions, measures).

In X-DFM, the above issues are resolved by modifying the affected constructs or introducing new ones. Table 3.3 provides an overview of the resulting elements of type node, defined by applying the following rationale:

Table 3.3: Graphical node type constructs of *X*-DFM

Element	Description
	A fact is a box-shaped node labeled by the fact name and containing two sets of elements: <i>i</i>) degenerated dimensions and <i>ii</i>) measures . Both sets are allowed to be empty.
	A degenerate fact is a many-to-many fact-dimensional relationship extracted into a separate fact, shown by placing a double-lined frame around the cell of the fact name.
● measure_name	A measure attribute is shown as a black circle-shaped node labeled by the measure's name. Measure nodes appear in the designated area of the fact node.
○ attribute_name	A dimension category corresponding to a non-abstract hierarchy level is a circle-shaped node labeled by the category's name.
◎ attribute_name ● measure_name	A derived dimension/measure attribute is shown as a double-lined circle-shaped node. Optionally, a dashed-line annotated with the derivation formula connects the derived element with its base element(s).
○ <u>attribute_name</u>	A fact identifier is a degenerated dimension with a one-to-one relationship to the fact, shown by underlining the attribute's name with a double line.
● category_name ● T category_name	An abstract dimension category is a circle-shaped node filled with grey color and labeled by the attribute's name. In case of a top-level category, the name is shown as a subscript of the T symbol.
◎ attribute_name ● T category_name	A totally ordered dimension category is marked by a dot in the node's center. A totally ordered dimension can be specified by placing a dot in the top category's node.
<u>attribute_name</u>	A property attribute is a characteristic associated with some dimension category, shown as an underlined attribute's name, connected by an undirected edge to its category node.
<u>attribute_name</u>	A “degree-of-belonging” attribute is a property associated with a child category of a non-strict weighted roll-up relationship.

- ◆ Non-dimension attributes of a fact should be treated as dimensions, as proposed in the literature: a dimension “stripped-off” to a single attribute is called *degenerated* [81].
- ◆ All attributes existing only in the context of their fact (i.e., measures and degenerated dimensions) are placed inside the fact node. Therefore, the fact's box is partitioned into two respective areas.

- ◆ Degenerate facts, i.e., many-to-many mappings extracted from other facts, are marked by double-lining the border of the fact's name (similar to a weak entity in the E/R notation).
- ◆ A dimension attribute with a one-to-one relationship to the fact (i.e., fulfilling the primary key property) is double-underlined.
- ◆ Measures are considered as a special kind of dimensions residing inside the fact. This assumption provides a basis for the interchangeability of measure and dimension roles. A measure's name is preceded by a circle node filled with black color.
- ◆ Abstract dimension categories consisting of a single value `all`, such as a top level of the whole dimension hierarchy or an abstract top node of a homogeneous subtree within a heterogeneous hierarchy, are shown as shaded circle nodes. The name of a top-level category is preceded by the "T" symbol.
- ◆ Derived measures and dimension categories are shown by double-lining the border of their nodes.
- ◆ Total order within a category is shown by placing a dot in the center of the category's node.


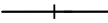
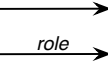


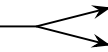
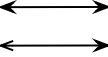
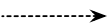
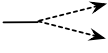
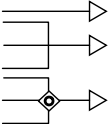
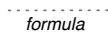
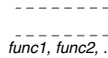
Different kinds of relationships, such as aggregation, composition, association, and generalization, are mapped to respectively different edge types, similar to those provided by UML, and are summarized in Table 3.4:

- ◆ An undirected edge is used for specifying non-hierarchical associations, such as the one between a property attribute and its category or a one-to-one relationship between a fact and a dimension. Optionality of an association is specified by placing a dash across the edge, as in DFM.
- ◆ A directed edge, or an arrow, stands for a many-to-one roll-up relationship.
- ◆ A bi-directed edge captures a many-to-many relationship. In case of a predominantly many-to-one relationship in a non-strict hierarchy, the stronger arrowhead indicates the roll-up direction.
- ◆ All kinds of related or alternative roll-up relationships (these relationships arise in case of multiple alternative hierarchies as well as in heterogeneous hierarchies of types non-covering and specialization) are visibly related by bundling their outgoing edge ends. Thereby, related hierarchies are distinguished from parallel ones.
- ◆ A dashed-line edge marks a partial roll-up relationship. In case of a set of related partial roll-up relationships, the bundled part of the edge is shown by a solid line to reflect the fact that the roll-up relationship of the child category to the set of those alternative parent categories is complete.
- ◆ A dotted-line edge links a derived element to its base element(s). A derivation formula can be shown as the edge's annotation.
- ◆ A generalization/specialization relationship between a category and its superclass/subclass is shown by an edge with a hollow triangle at the superclass end, adopted from the UML. A set of related specializations is shown in a shared-target style, i.e., as a tree rooted at the superclass.

Notice, that some of the edge properties may be used in combination, e.g., a roll-up relationship can be partial and non-strict, whereas other properties are mutually exclusive, e.g., a roll-up relationship can be either full or partial.

Figure 3.2 shows the results of adjusting the original PURCHASE scheme from Figure 3.1 according to the *X*-DFM notation. Graphical elements not explicated so far and not covered by this simple example are detailed in the next chapters. As we proceed with the formalization of the extended multidimensional model, the advantages of the proposed notation will become more apparent.

Table 3.4: Graphical edge type constructs of *X*-DFM

Element	Description
	An association relationship is an undirected edge connecting a property attribute with its category or connecting a fact with a dimension in case of a one-to-one relationship between the two.
	An optional association relationship is shown by putting a dash across the edge.
	A full strict roll-up is a many-to-one relationship between a fact and a category or between a pair of categories, shown as a edge directed towards the parent category. In case the same category is a target of multiple roll-up relationships, each roll-up edge can be labeled by the respective role of that category.
	A complete roll-up is a many-to-one relationship within a complete hierarchy, shown by a diamond at the outgoing end of the roll-up edge.
	A fuzzy roll-up relationship, in which child elements are assigned to parent elements dynamically based on some rules, is marked as a double-pointed arrow.
	Multiple alternative roll-up relationships are alternative, i.e., mutually incompatible, aggregation paths of the same child category, shown by bundling the roll-up edges into a common edge at the outgoing end.
	A many-to-many relationship between categories is shown as a bi-directed edge. In case of a non-strict roll-up relationship, the direction of the roll-up is indicated by a stronger arrowhead.
	A partial roll-up is an optional roll-up relationship of the child category, shown as a directed dotted-line edge.
	Related partial roll-ups are a set of mutually exclusive roll-up relationships in a heterogeneous hierarchy, shown by bundling the outgoing parts of the edges into a single solid-line edge.
	Generalization/specialization is shown as a solid-line edge with a hollow triangle at the superclass end. The edges of related specializations are shown in a shared-target style. By default, specialization is disjoint. Overlapping subclasses are specified by placing a diamond with “o” symbol onto the edge at the point where it branches into subclass edges.
	Derivation relationship is a dotted-line connecting a derived element to its input element(s).
	Non-aggregability/non-additivity edge is adopted from DFM.

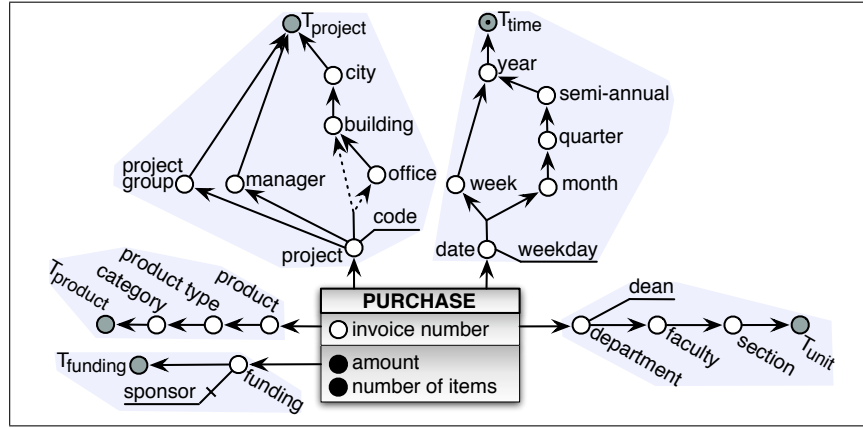


Figure 3.2: The revised fact scheme PURCHASE in the X-DFM notation

3.3.2 Formalization

According to the classification framework of Abelló [2], existing conceptual multidimensional data models can be grouped into three categories according the level of semantic details they provide:

1. **Upper Level (UL)** models use the constructs *fact* and *dimension* without further subdivision, thus enabling modeling of star-shaped fact schemes. In this models, the terms *fact* and *measure* are used synonymously, i.e., each measure is modeled as a fact of its own.
2. **Intermediate Level (IL)** models recognize decomposition of facts and dimensions into *cells* and *levels*, respectively. Different levels in a dimension are arranged into hierarchies and a cell contains the measure's value for a given set of dimensional values.
3. **Lower Level (LL)** models structure the attributes constituting dimension levels and fact cells into *classification attributes* and *measures*, respectively. At this level, the term *fact* is no longer a synonym of *measure*, but is rather a set of measures of the same granularity.

The use of term “lower level” in this classification may appear confusing as it actually stands for the higher level of detail. Supposedly, the term “level” in the above classification is to be interpreted as the level of abstraction, which is the opposite of detail. Obviously, LL models provide the finest level of detail necessary for obtaining the logical model from the conceptual one. Besides, the other two levels are comprised by LL and can be obtained by eliminating the elements that go beyond that level's scope. Less accurate IL and UL models, even though insufficient for deriving full-fledged logical schemata, are useful at initial design stages as well as for modeling simple OLAP applications as their constructs are sufficient for specifying the overall structure of a cube.

We decided to define and formalize the basic multidimensional model at each level of detail in a top-down fashion. This way, we ensure consistency and downward compatibility of all introduced concepts, and the proposed basic model is applicable at any of the three defined levels. The scheme in Figure 3.2 is an example of an LL model as it explicitly captures each attribute as a separate node. Figure 3.3 contains the same data fragment, modeled in X-DFM according to (a) UL and (b) IL.

Notice that only the basic elements of the multidimensional model can be specified at all three levels of abstraction. Advanced concepts, such as overlapping fact schemes and complex dimension hierarchies, presented in Chapters 5 and 4, respectively, premise the conceptual framework of the LL model.

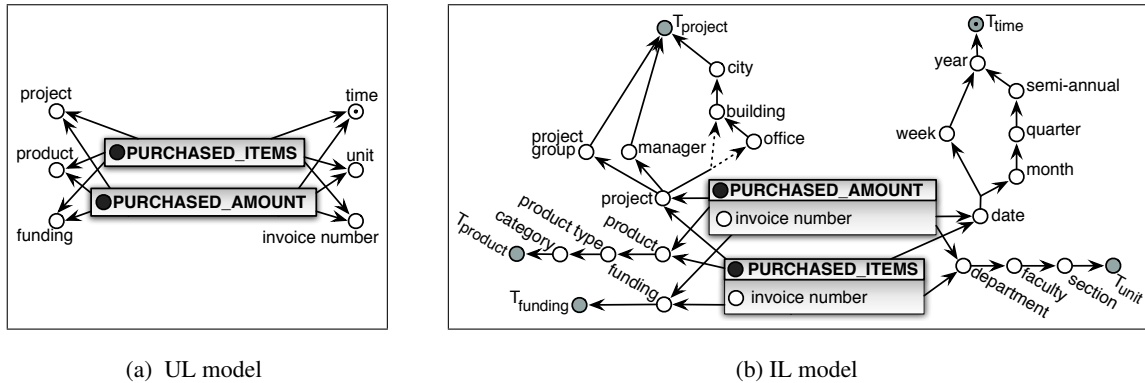


Figure 3.3: Modeling purchasing facts at higher levels of abstraction

In the remainder of this chapter, we provide the formalization of the fundamental elements of the conceptual model as a foundation for defining extensions and specializations of those elements described in the next two chapters.

3.4 Terminology and Definitions

In this section, the elements of the multidimensional model are defined successively at the upper, the intermediate, and the lower level of abstraction. Once the basic definitions and general concepts are provided, further characteristics and special cases are deduced.

3.4.1 Unified Multidimensional Space

One fundamental definitional issue to be clarified prior to the formalization is whether the semantics, i.e., relationships between fact schemes as well as between dimensions, should be captured by the conceptual model. A conventional approach would be to model each n -dimensional fact scheme in its own isolated n -dimensional space. The output of such a model is a set of unrelated fact schemes. Advanced models however, such as YAM² [1], OOMD [174], and DFM [47], support inter-factual semantics by allowing facts to share dimensions. The major advantage of the latter approach is explicit support for the drill-across operation that allows to compare measures of related data cubes or even derive new measures by combining the existing ones. Further kinds of relationships, such as specialization, aggregation, composition, and degeneration, are also possible between multiple fact schemes, when their dimensions are defined in a non-redundant multidimensional space.

In UL models, the resulting conceptual scheme is called *multi-star*. A pair of dimensions is merged into one shared dimension, if the former are defined on a related semantic domain [1]. For example, dimensions customer city and sale city would be modeled as a common dimension city containing the union of values from both dimensions. IL and LL models recognize further types of dimension sharing besides full sharing by decomposing dimension structure into categories and attributes and considering semantic compatibility at the level of single categories. The resulting conceptual scheme is called *inter-stellar*, or *galaxy*.

Inter-factual relations are useful not only for the analysis, but also for the design itself as their recognition helps to reduce maintenance overhead and automatically detect valid operations. To fully capture these

relationships, our model employs the concept of a *unified multidimensional space*, in which dimensions (as well as dimension categories at IL and LL) with semantically related value domains are represented in a non-redundant fashion. Formal unification of the multidimensional space is achieved by defining the notions of related domains, compatible and conform dimensions, and related fact schemes.

The issue of dimension sharing, or conformation, was first brought up by Kimball in [81], who introduced the term *conformed dimensions* to refer to dimensions, which are not physically centralized but which have identical schemes. Our approach to dimension sharing differs from that of Kimball, as the latter addresses the logical design (e.g., centralization and normalization) whereas we are concerned with the conceptual modeling. A unified multidimensional space approach of our conceptual model does not impose any particular logical or physical implementation scheme. Moreover, this approach is beneficial for generating accurate metadata to support advanced OLAP operations and data navigation options in frontend tools irrespective of the backend implementation.

3.4.2 Facts and Dimensions

All definitions are first given for the UL model and subsequently refined into the IL and, finally, the LL model. Concepts defined at higher levels of abstraction represent coarsements of their counterparts at lower levels, thus preserving the coherence of the definitional framework from one level to another.

UPPER LEVEL MODEL

From the UL perspective, *fact* and *dimension* are the only available elements of the conceptual model.

DEFINITION UL-FACT. A *fact* F is a collection of uniformly structured data entries over a fact scheme \mathcal{F} , where \mathcal{F} is determined by a set of dimensions $\mathcal{D}^{\mathcal{F}} = \{D_i, i = 1, \dots, n\}$.

DEFINITION UL-DIMENSION. A *dimension* D is a nominal category with member values of type \mathcal{D} . A *dimension type* \mathcal{D} characterizes the semantics of all potentially possible values drawn from the same value domain, denoted $\text{Domain}(\mathcal{D})$.

Fact scheme \mathcal{F} related to n dimensions ($|\mathcal{D}^{\mathcal{F}}| = n$) is referred to as an *n-dimensional fact scheme*. In the UL fact definition, there is no mention of *measures* since the fact itself is understood as a measure along with its dimensional characteristics. However, our model needs a mechanism for specifying factless and derived facts. Since fact and measure types are detailed in the next chapter, at this point we only consider the problem of representing those properties in the graphical model with the respective examples found in Figure 3.4:

- ◆ A fact corresponding to a measure (default case) is shown by prepending the name of the fact node with a measure symbol. As an example, consider the fact ITEMS_SOLD in Figure 3.4b.
- ◆ A fact with a derived measure is shown by prepending the name of the fact node with a derived measure symbol and connecting it to its base fact(s) with a derivation edge. An example of a derived fact in Figure 3.4b is PROFIT, computed from ITEMS_SOLD.
- ◆ A fact with no measure is shown with no measure symbol next to the fact's name, as is the case with the fact RECRUITMENT in Figure 3.4a.

The only type of a UL roll-up relationship is a *fact-dimensional roll-up*, denoted $\mathcal{F} \sqsubseteq D_i$, where D_i is any dimension in fact scheme \mathcal{F} . By default, this relationship is assumed to be *full*, denoted $\sqsubseteq^{(\text{full})}$, implying full participation of fact entries in F in dimension D_i . Otherwise, D_i is an optional dimension in \mathcal{F} resulting

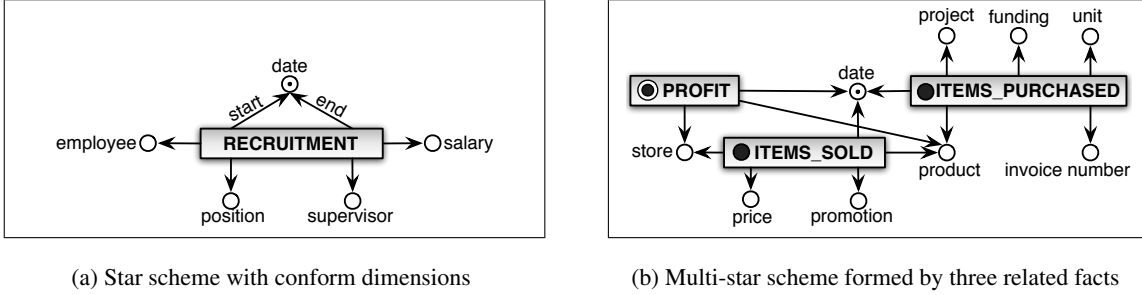


Figure 3.4: Examples of dimension sharing in star and multi-star schemes

in a *partial* roll-up relationship, denoted $\sqsubseteq^{(\text{part})}$. The entire set of the fact's roll-up relationships, denoted $\sqsubseteq_{\mathcal{F}}$, stands for the union of all full and partial roll-up relationships of \mathcal{F} .

Let e be a member value of D , denoted $e \in D$. The following conditions hold for e : $\forall e \in D : \text{Type}(e) = \mathcal{D} \wedge e \in \text{Domain}(\mathcal{D})$, or informally, all values are of the same data type. The above definition of a dimension distinguishes between the notions “dimension” and “dimension type” in order to admit existence of multiple dimensions of the same type. Consider an example of modeling temporal characteristics of a fact. Dimension type *date* describes the overall semantics of date as a characteristic and defines its value domain $\text{Domain}(\text{date}) = \text{'DD-MM-YYYY'}$. A fact may have more than one dimensional characteristic of type *date*, e.g., start date and stop date. We call such dimensions *conform*. Member sets of conform dimensions do not have to be identical or even overlap – the only requirement is that all member values are drawn from the same domain, i.e., possess the same semantics.

DEFINITION UL-CONFORM DIMENSIONS. Dimensions D_i and D_j are *conform*, if they are of the same dimension type: $\text{Conform}(D_i, D_j) \Leftarrow (D_i \neq D_j \wedge \mathcal{D}_i = \mathcal{D}_j)$.

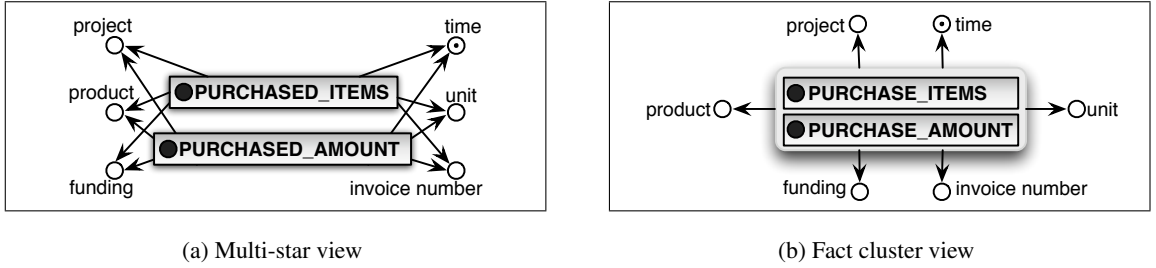
Defined as a nominal category, a dimension assumes no order between its members. However, some dimensions are defined on ordinal domains (e.g., time). In that case, the dimension is said to be *totally ordered*. We use operators $<$ and $>$ to specify the order for a pair of ordinal values.

DEFINITION UL-TOTALLY ORDERED DIMENSION. A dimension D is *totally ordered*, if there exists an order between its members: $\text{Ordered}(D) \Leftarrow (\forall e_i, e_j \in D : (e_i < e_j \vee e_i > e_j))$.

Back to the concept of conform dimensions, we have obtained a foundation for dimension sharing: in a unified multidimensional space, each set of conform dimensions is represented by a *shared dimension* of the respective type. Thereby, the graphs of fact schemes containing a shared dimension appear connected.

DEFINITION UL-RELATED FACTS. Fact schemes \mathcal{F}_k and \mathcal{F}_l are *related*, if they share at least one dimension:
 $\text{Related}(\mathcal{F}_k, \mathcal{F}_l) \Leftarrow (\exists D_i \in \mathcal{F}_k, \exists D_j \in \mathcal{F}_l : \text{Conform}(D_i, D_j))$.

In the UL model, fact schemes are also known as *stars*. A set of related facts form a *multi-star* scheme since their schemes are connected through shared dimensions. Dimension sharing may occur both within the same fact scheme and between different schemes. Figure 3.4 shows the respective examples of dimension

Figure 3.5: Arranging facts into a cluster in X -DFM

sharing in fact schemes of type single star and multi-star: fact RECRUITMENT in (a) has conform dimensions start and end of type date; facts ITEMS_SOLD and ITEMS_PURCHASED in (b) are related via shared dimensions of types product and date.

A special case of a multi-star is given by multiple facts with identical dimension sets. Intuitively, such facts represent different measures in exactly the same multidimensional space. We introduce the term “fact cluster” to refer to a set of identically structured facts.

DEFINITION UL-FACT CLUSTER. Facts F_k and F_l form a *cluster*, if their schemes are identical:
 $Cluster(F_k, F_l) \Leftarrow \mathcal{F}_k = \mathcal{F}_l$.

In X -DFM, fact nodes that form a cluster are put into a common cluster super-node in order to obtain a non-redundant presentation of their schemes. Figure 3.5 exemplifies the benefits of recognizing fact clusters in X -DFM: (a) is an unclustered view of two facts with identical schemes and (b) is a unified view of the same fact schemes obtained by putting both fact nodes into a common cluster node.

The output of the conceptual data warehouse design for the entire application domain is given by the set of its fact schemes and is denoted a *fact family* \mathcal{F} . This concept is crucial for the formalization of the multidimensional space. We first define the multidimensional space in the context of a single fact scheme and then generalize it for a family of fact schemes.

For any dimension $D_i \in \mathcal{D}^{\mathcal{F}}$, $Dim(D_i)$ stands for the projection of cube F over D_i . The resulting multidimensional space of a cube groups all valid combinations built up by considering the value sets of dimensions in $\mathcal{D}^{\mathcal{F}}$. Each dimension’s value set is enriched with an abstract value ALL, a generalization of all possible values in the dimension, which is considered the originating point of that dimension.

DEFINITION UL-MULTIDIMENSIONAL SPACE (FACT). A *multidimensional space* of fact F , denoted $Space(F)$, is defined as follows: $Space(F) = \{\times_{D_i \in \mathcal{D}} (Dim(D_i) \cup ALL)\} \cup \{\emptyset, \dots, \emptyset\}$, where \times is the Cartesian product and $\{\emptyset, \dots, \emptyset\}$ stands for the combination of empty values.

Informally, dimensions serve as axes of a data cube with ordered dimension values as coordinates. Fact entries are mapped to the points of the multidimensional space. The set of each point’s coordinates is called a *multidimensional pattern* and the point’s value is given by the corresponding measure value. A multidimensional pattern with no associated fact entry is considered *empty*. Intuitively, the multidimensional space of a family of facts \mathcal{F} is built up as a Cartesian product of the value sets of all dimensions in \mathcal{F} .

Figure 3.6 illustrates the concept of a multidimensional space at a simple example of three dimensions X , Y , Z , with 9, 6, and 7 member values, respectively. All dimensions originate at the common point $(\emptyset, \emptyset, \emptyset)$. Not surprisingly, an n -dimensional space has a shape of an n -dimensional cube.

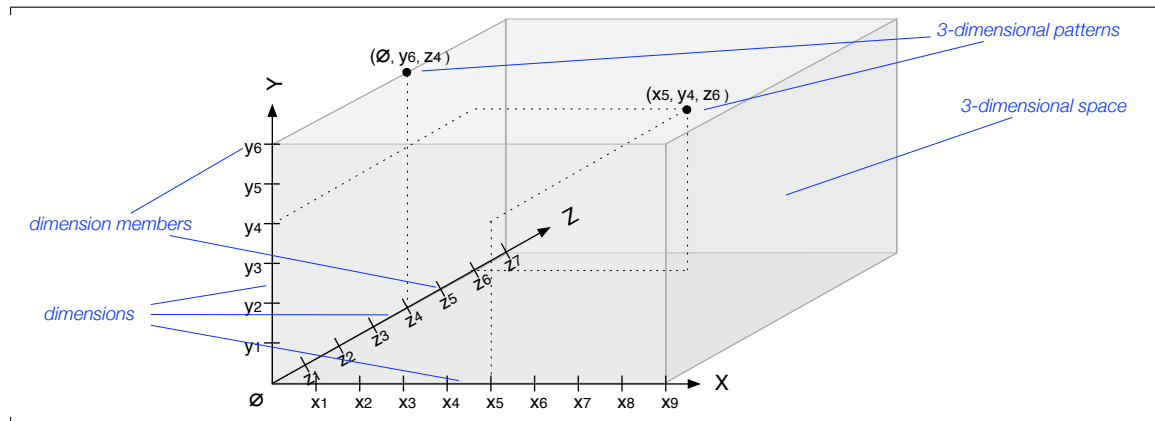


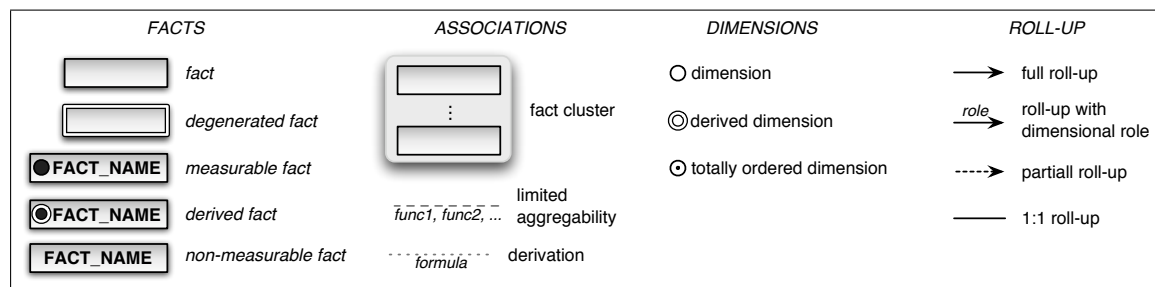
Figure 3.6: A 3-dimensional space produced by dimensions X, Y, and Z

DEFINITION UL-MULTIDIMENSIONAL SPACE (FACT FAMILY). A *multidimensional space* of a fact family \mathcal{F} , denoted $Space(\mathcal{F})$, is defined as follows: $Space(\mathcal{F}) = \{\times_{D_i \in \bigcup_{\mathcal{F}} (Dim(D_i) \cup ALL)}\} \cup \{\emptyset, \dots, \emptyset\}$ where $\bigcup_{\mathcal{F}}$ is a union of all dimensions of all facts in \mathcal{F} .

A *multidimensional space!unified* disallows co-existence of conform dimensions by replacing each conforming set by a single shared dimension type:

DEFINITION UL-UNIFIED MULTIDIMENSIONAL SPACE. A multidimensional space is *unified*, if none of its dimensions is conform with another dimension:
 $Unified(Space(\mathcal{F})) \Leftarrow \forall D_i \in Space(\mathcal{F}) : (\nexists D_j \in Space(\mathcal{F}) : Conform(D_i, D_j)).$

As can be seen from the set of definitions in this section, the UL model spans a rather limited set of elements and multidimensional properties: facts are not decomposable into measures, dimensions are non-hierarchical categories, and categories are not decomposable into attributes. As a result, only a small subset of X -DFM constructs are valid in the context of the UL model, as summarized in Figure 3.7. Apparently, even the valid graphical constructs had to be adjusted to comply with the respective UL definitions. For example, fact nodes have no subareas for measures and degenerated dimensions, and each dimension is represented by a single category node.

Figure 3.7: The UL construct set of X -DFM

INTERMEDIATE LEVEL MODEL

Unlike in the UL model, the IL allows to structure dimensions into aggregation hierarchies. In OLAP, dimensions are used for aggregating data to a desired granularity. Therefore, the task of the conceptual model is to capture the aggregation semantics and valid navigation paths for the analysis. The notions of fact and dimension are redefined to account for the concept of hierarchically structured dimension schemes:

DEFINITION IL-FACT. A *fact* F is a collection of uniformly structured data entries over a fact scheme \mathcal{F} , where \mathcal{F} is determined by a set of dimension schemes $\mathcal{D}^{\mathcal{F}} = \{\mathcal{D}_i, i = 1, \dots, n\}$.

DEFINITION IL-DIMENSION. A *dimension* D is defined by its hierarchy scheme (*intension*) \mathcal{D} and the associated member set (*extension*) E , so that $Type(E) = \mathcal{D}$.

DEFINITION IL-DIMENSION SCHEME. A *dimension scheme* is a quadruple $\mathcal{D} = (\mathcal{C}^{\mathcal{D}}, \sqsubseteq_{\mathcal{D}}, \top_{\mathcal{D}}, \perp_{\mathcal{D}})$, where $\mathcal{C}^{\mathcal{D}} = \{\mathcal{C}_k, k = 1, \dots, p\}$ is set of category types in \mathcal{D} , $\sqsubseteq_{\mathcal{D}}$ is a partial order in \mathcal{C} , whereas $\top_{\mathcal{D}}$ and $\perp_{\mathcal{D}}$ are distinguished as the top and the bottom element of the ordering, respectively.

Informally, a dimension scheme is a connected, directed graph, in which vertices correspond to hierarchy levels and edge represent roll-up relationships between the levels. $\perp_{\mathcal{D}}$ corresponds to the finest grain of \mathcal{D} , i.e., the one at which \mathcal{D} is connected to the fact scheme. Notice that the property of being a bottom element is not global, but is valid only in the context of a given fact scheme. Therefore, in multi-fact schemes, X -DFM shows the category's actual name instead of $\perp_{\mathcal{D}}$ notation. $\top_{\mathcal{D}}$ corresponds to an abstract root node of the dimension hierarchy and has a single value referred to as ALL: $\top_{\mathcal{D}} = \{\text{ALL}\}$. Function *Abstract*() returns true, if the input category is an abstract one.

Relation $\sqsubseteq_{\mathcal{D}}$ captures the containment relationships between category types. This containment may be *full*, denoted $\sqsubseteq_{\mathcal{D}}^{(\text{full})}$, or *partial*, denoted $\sqsubseteq_{\mathcal{D}}^{(\text{part})}$. Thereby, relation $\sqsubseteq_{\mathcal{D}}$ indicates the union of the two orders $\sqsubseteq_{\mathcal{D}}^{(\text{full})}$ and $\sqsubseteq_{\mathcal{D}}^{(\text{part})}$. Admission of partial containment, also known as *partial roll-up* relationship, between category types is crucial for specifying heterogeneous dimension hierarchies.

Predicates \sqsubseteq and \sqsubseteq^* specify *direct* and *transitive* containment relationship, respectively, between a pair of category types in \mathcal{C} . Partial and full direct containment predicates are denoted $\sqsubseteq^{(\text{part})}$ and $\sqsubseteq^{(\text{full})}$, respectively. Predicates \sqsubseteq and \sqsubseteq^* without fullness / partiality indication imply that the containment is either full or partial: $\mathcal{C} \sqsubseteq \mathcal{C}' \Rightarrow (\mathcal{C}_i \sqsubseteq^{(\text{full})} \mathcal{C}_j \vee \mathcal{C}_i \sqsubseteq^{(\text{part})} \mathcal{C}_j)$. Partial containment between any two categories \mathcal{C}_i and \mathcal{C}_j ($\mathcal{C}_i \sqsubseteq^{(\text{part})} \mathcal{C}_j$) occurs when members of \mathcal{C}_i are not required to have parent members in \mathcal{C}_j .

A pair of partial containment relationships of the same category \mathcal{C}_i (i.e., $\mathcal{C}_i \sqsubseteq^{(\text{part})} \mathcal{C}_j \wedge \mathcal{C}_i \sqsubseteq^{(\text{part})} \mathcal{C}_k$) is mutually *exclusive*, if any member of \mathcal{C}_i rolls-up either to \mathcal{C}_j or \mathcal{C}_k , but never to both. A set of exclusive partial roll-up relationships is denoted $\mathcal{C}_i \sqsubseteq^{\text{part}} (\mathcal{C}_j | \mathcal{C}_k)$.

Back to Figure 3.3b, dimension project has a set of categories $\mathcal{C}^{\text{project}} = \{\perp_{\text{project}}, \text{office}, \text{building}, \text{city}, \text{manager}, \text{project_group}, \top_{\text{project}}\}$ and a partial order $\sqsubseteq_{\text{project}} = \{\perp_{\text{project}} \sqsubseteq^{(\text{full})} \text{project_group}, \text{project_group} \sqsubseteq^{(\text{full})} \top_{\text{project}}, \perp_{\text{project}} \sqsubseteq^{(\text{full})} \text{manager}, \text{manager} \sqsubseteq^{(\text{full})} \top_{\text{project}}, \perp_{\text{project}} \sqsubseteq^{(\text{part})} (\text{office} | \text{building}), \text{office} \sqsubseteq^{(\text{full})} \text{building}, \text{building} \sqsubseteq^{(\text{full})} \text{city}, \text{city} \sqsubseteq^{(\text{full})} \top_{\text{project}}\}$. The roll-up between \perp_{project} and office is partial due to the implied heterogeneity of project instances: internal projects roll-up to office, whereas external ones roll-up to building. These two parent categories form an exclusive partial roll-up $\perp_{\text{project}} \sqsubseteq^{(\text{part})} (\text{office} | \text{building})$.

The following properties hold for the partial order relation $\sqsubseteq_{\mathcal{D}}$ and its predicates:

- ◆ Antireflexivity: $\nexists \mathcal{C}_j \in \mathcal{C}^{\mathcal{D}} : \mathcal{C}_j \sqsubseteq \mathcal{C}_j$.
- ◆ Antisymmetry: $\nexists (\mathcal{C}_i, \mathcal{C}_j) \in \mathcal{C}^{\mathcal{D}} : (\mathcal{C}_i \sqsubseteq \mathcal{C}_j \wedge \mathcal{C}_j \sqsubseteq \mathcal{C}_i)$.
- ◆ Transitivity : $\forall (\mathcal{C}_i, \mathcal{C}_j, \mathcal{C}_k) \in \mathcal{C}^{\mathcal{D}} : (((\mathcal{C}_i \sqsubseteq \mathcal{C}_j \vee \mathcal{C}_i \sqsubseteq^* \mathcal{C}_j) \wedge (\mathcal{C}_j \sqsubseteq \mathcal{C}_k \vee \mathcal{C}_j \sqsubseteq^* \mathcal{C}_k)) \Rightarrow \mathcal{C}_i \sqsubseteq^* \mathcal{C}_k)$.

Antireflexivity forbids reflexive roll-up, i.e., a relationship of a category with itself. A classical example of such reflexive relationship could be a supervisor hierarchy within the category manager. Antisymmetry disallows bi-directional rolls-up between any pair of categories, since that would result in cyclic aggregation paths. Thereby, the first two properties guarantee acyclic termination of all aggregation paths. Transitivity defines indirect roll-up relationships within a hierarchy. For instance, if date is contained in month and month is contained in year, then date is transitively contained in year.

C_j is said to be a category type in $\mathcal{C}^{\mathcal{D}}$, denoted $C_j \in \mathcal{C}^{\mathcal{D}}$. A dimension scheme defines a skeleton of the associated data tree, for which the following conditions hold:

1. $\forall C_j \in \mathcal{C}^{\mathcal{D}} \setminus \{\top_{\mathcal{D}}\} : C_j \sqsubseteq^{*(full)} \top_{\mathcal{D}}$ (non-top category types are fully contained in the top category).
2. $\forall C_j \in \mathcal{C}^{\mathcal{D}} \setminus \{\perp_{\mathcal{D}}\} : \perp_{\mathcal{D}} \sqsubseteq^* C_j$ (the bottom category type of a dimension is contained in all its other category types, either fully or partially).
3. $\nexists C_j \in \mathcal{C}^{\mathcal{D}} : C_j \sqsubseteq \perp_{\mathcal{D}}$ (the bottom category type is childless in the context of a given dimension).

In the simplest case, a dimension consists solely of the bottom and the top category types, i.e., is non-hierarchical. A scheme of a single hierarchy is a lattice, whereas dimension schemes of multiple hierarchies may result in rather complex graph structures. Multiple hierarchies in \mathcal{D} exist whenever there exists a category type at which at least two paths converge, or formally: $\exists C_i, C_j, C_k \in \mathcal{C}^{\mathcal{D}} : C_i \sqsubseteq^{(full)} C_k \wedge C_j \sqsubseteq^{(full)} C_k$. Figure 3.8 shows examples of dimension schemes of various complexity.

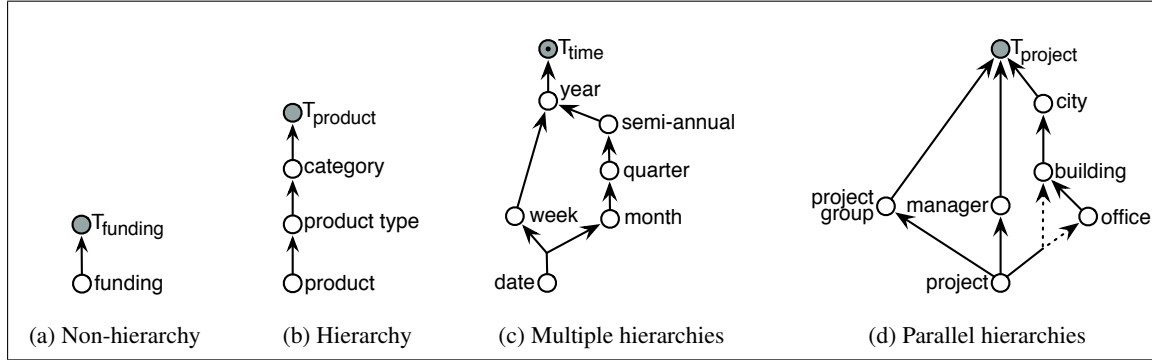


Figure 3.8: Dimension schemes as directed graphs of various complexity

Since the IL model structures facts into cells of measure values determined by the respective dimension values, scheme definitions at this level should be supplied with the corresponding instance definitions.

DEFINITION IL-DIMENSION INSTANCE. An *instance*, or *extension*, E associated with dimension scheme \mathcal{D} is a pair (C, \sqsubseteq_E) , where $C = \{C_j, j = 1, \dots, m\}$ is a set of categories such that $Type(C_j) = C_j$ and \sqsubseteq_E is a partial order on $\bigcup_j C_j$, the union of all dimensional values in the individual categories.

DEFINITION IL-CATEGORY. A *category* C_j of type \mathcal{C}_j is a set of member values e_k such that $\forall e_k \in C_j : Type(e_k) = \mathcal{C}_j$.

The definition of the partial order \sqsubseteq_E in the context of dimension values is as follows: given $(e_1, e_2) \in \bigcup_j C_j$, $e_1 \sqsubseteq e_2$, if e_1 is logically contained in e_2 . Predicates \sqsubseteq and \sqsubseteq^* specify the direct and the transitive

containment relationship, respectively, between members. The containment relationship at the category levels is expressed using the same predicates \sqsubseteq and \sqsubseteq^* , as used at category type level. The total number of member values in category C_j is denoted $|C_j|$. The following conditions hold for a dimension instance:

1. $\forall e_m \in C_i, \forall e_n \in C_j : e_m \sqsubseteq e_n \Rightarrow C_i \sqsubseteq C_j$ (connectivity). This condition ensures that the containment relationship between categories results from the containment relationship between the members of those categories and disallows roll-up relationships between members of unrelated categories.
2. $\nexists e_m \in C_i, \nexists e_n \in C_j : e_m = e_n \wedge C_i \neq C_j$ (disjointness of category types). Prohibiting any value to be a member of multiple category types enforces unification of conform categories into a shared category type as well as disjointness of categories referring to different types.
3. $\nexists e_m, e_n \in C_i : e_m \sqsubseteq e_n \vee e_m \sqsubseteq^* e_n$ (stratification, i.e., disallowing direct or transitive containment within members of the same category).
4. $(Type(C_j) = \top_D) \Rightarrow (|C_j| = 1 \wedge C_j = \{ALL\})$ (top category consists of a single value ALL).

As already mentioned, a roll-up relationship between category types may be either *full* or *partial*. The containment pattern actually originates in the dimension's extension as the former is determined by the roll-up behavior of member values, and is simply propagated to the respective category type.

DEFINITION IL-FULL ROLL-UP. A roll-up relationship between a pair of categories C_i and C_j is *full*, denoted $C_i \sqsubseteq^{(full)} C_j$, if each member in C_i has a containing member in C_j , i.e., $\forall e_m \in C_i : (\exists e_n \in C_j : e_m \sqsubset e_n)$.

DEFINITION IL-PARTIAL ROLL-UP. A roll-up relationship between a pair of categories C_i and C_j is *partial*, denoted $C_i \sqsubseteq^{(part)} C_j$, if C_i admits members with no containing member in C_j , i.e., $\exists e_m \in C_i : (\nexists e_n \in C_j : e_m \sqsubset e_n)$.

DEFINITION IL-PARTIAL RELATED ROLL-UPS. A set of partial roll-up relationships of category C_i ($C_i \sqsubseteq^{(part)} C_j, \dots, C_i \sqsubseteq^{(part)} C_n$) is called *exclusive*, if each member of C_i is directly contained in only one of the multiple parent categories. Partial related roll-ups are denoted $C_i \sqsubseteq^{(part)} (C_j | \dots | C_n)$.

So far, we assumed a partial order between the elements belonging to different hierarchy levels. However, if a category is defined on an ordinal value domain, its members are *totally ordered*.

DEFINITION IL-TOTALLY ORDERED CATEGORY. A category C_j is *totally ordered*, if there exists an order between its members: $Ordered(C_j) \Leftarrow (\forall e_k, e_l \in C_j, k \neq l : (e_k < e_l \vee e_k > e_l))$.

The property of the total order can be *inherent* to a category, i.e., imposed by the category's value domain, or *propagated* from a totally ordered category in the hierarchy path. For semantic correctness, a category in X -DFM should be marked as totally ordered only then, if that order is inherent. Apparently, a hierarchy is totally ordered, if each of its levels is totally ordered, and, similarly, a dimension is totally ordered if all its hierarchies are ordered. In X -DFM, such dimension is marked by a totally ordered top-level category symbol, as used for the time dimension in Figure 3.8c.

DEFINITION IL-TOTALLY ORDERED DIMENSION. A dimension D is *totally ordered*, if a total order exists in each of its categories: $Ordered(D) \Leftarrow (\forall C_j \in D : Ordered(C_j))$.

The concept of the multidimensional space also needs to be reconsidered, as the IL model allows dimensions to be arbitrarily complex (e.g., hierarchical, heterogeneous), whereas dimensions of the multidimensional space represent non-hierarchical value domains. Therefore, each dimension category should be considered a dimension of its own in the multidimensional space. For any category $C_i \in \mathcal{D}^{\mathcal{F}}$, $Dim(C_i)$ stands for the projection of cube F over C_i . The resulting multidimensional space of a cube groups all valid combinations built up by considering the value set of each category in $\mathcal{D}^{\mathcal{F}}$.

DEFINITION IL-MULTIDIMENSIONAL SPACE (FACT). A *multidimensional space* of fact F , denoted $Space(F)$, is defined as follows: $Space(F) = \{\times_{C_i \in \mathcal{D}} (Dim(C_i) \cup ALL)\} \cup \{\emptyset, \dots, \emptyset\}$, where \times is the Cartesian product and $\{\emptyset, \dots, \emptyset\}$ stands for the combination of empty values.

DEFINITION IL-MULTIDIMENSIONAL SPACE (FAMILY). A *multidimensional space* of a fact family \mathcal{F} , denoted $Space(\mathcal{F})$, is defined as follows: $Space(\mathcal{F}) = \{\times_{C_i \in \bigcup_{\mathcal{D}} \mathcal{D}} (Dim(C_i) \cup ALL)\} \cup \{\emptyset, \dots, \emptyset\}$ where $\bigcup_{\mathcal{D}}$ is a union of all dimension categories in all dimensions of all facts in \mathcal{F} .

Prior to proceeding with the definition of the unified multidimensional space, we have to reconsider the concept of conform dimensions in presence of dimension hierarchies, multiple hierarchies, and heterogeneous roll-ups within a dimension. IL models define dimensions in terms of directed aggregation graphs of their categories. Therefore, the compatibility of dimension schemes results from the compatibility at each dimension level. Thereby, dimension schemes may have different semantic relationships with each other, such as *full coincidence*, *convergence*, *overlap* or *disjointness*. At this stage, we provide definitions of compatibility patterns in dimension categories, and will reason about sharing patterns in entire dimension schemes in Section 5.4 of the next chapter. Since categories are involved into roll-up relationships, we propose to distinguish between two types of semantic relations, such as *compatibility* and *conformance*:

DEFINITION IL-COMPATIBLE CATEGORIES. Categories C_i and C_j are *compatible*, if they belong to the same category type: $Compatible(C_i, C_j) \Leftarrow (C_i \neq C_j \wedge Type(C_i) = Type(C_j))$.

DEFINITION IL-CONFORM CATEGORIES. Compatible categories C_i and C_j are *conform*, if they roll-up to compatible sets of category types: $Conform(C_i, C_j) \Leftarrow Compatible(C_i, C_j) \wedge (\forall C_m, C_i \sqsubseteq C_m : (\exists C_n, C_j \sqsubseteq C_n : Conform(C_m, C_n)))$.

Figure 3.9, showing a slightly modified variant of the multi-star scheme from Figure 3.3b, should serve as an illustrating example for the above concepts. Examples of conform categories are order date and receipt date as their hierarchy schemes are identical. Categories office and building in project dimension are compatible, but not conform, to their counterparts in unit dimension, as they roll-up along different paths.

From the compatibility of single categories, we infer the notion of *related* dimensions and facts:

DEFINITION IL-RELATED DIMENSIONS. Dimensions D_i and D_j are *related*, if they share at least one category: $Related(D_i, D_j) \Leftarrow \exists C_m \in E_i, \exists C_n \in E_j : Compatible(C_i, C_j)$.

DEFINITION IL-RELATED FACTS. Fact schemes \mathcal{F}_k and \mathcal{F}_l are *related*, if at least one dimension in \mathcal{F}_k is related to at least one dimension in \mathcal{F}_l : $Related(\mathcal{F}_k, \mathcal{F}_l) \Leftarrow \exists D_i \in \mathcal{F}_k, \exists D_j \in \mathcal{F}_l : Related(D_i, D_j)$.

Apparently, fact schemes may be related to each other at the bottom granularity or at some upper aggregation levels. The resulting multi-fact scheme is referred to as a *galaxy*. The UL definition of a fact

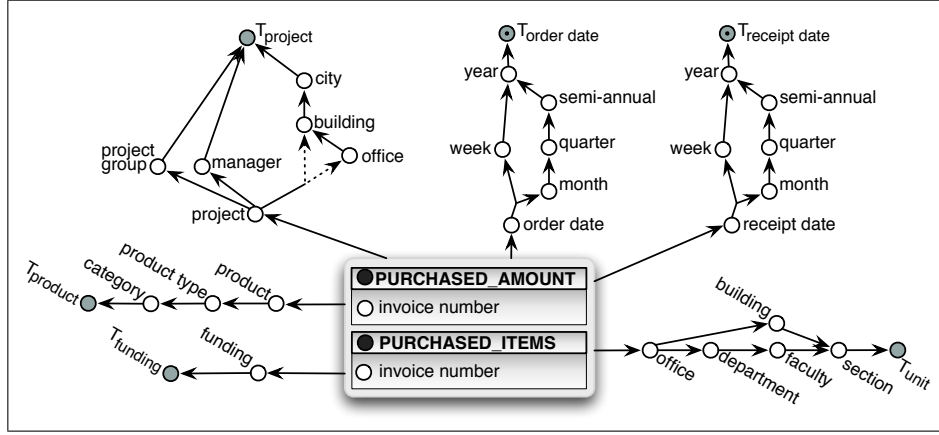


Figure 3.9: Clustered IL view of purchasing facts with compatible and conform categories

cluster remains valid in the IL context and is useful for arranging facts with identical sets of dimensions into a common super-node in X -DFM, as was done in Figure 3.9.

We suggest that top-level categories should be essentially treated as unique to account for the fact that compatible dimensions may have different member sets and that the abstract root value ALL covers only the respective dimension's member set. Therefore, top-level categories are exempted from the compatibility test. While the UL model unifies the multidimensional space by merging conform dimensions, the IL model does the same at the category level:

DEFINITION IL-UNIFIED MULTIDIMENSIONAL SPACE. A multidimensional space is *unified*, if it maps each set of compatible categories to a single shared category type:
 $Unified(Space(\mathcal{F})) \Leftarrow \forall C_i \in Space(\mathcal{F}) : (\nexists C_j \in Space(\mathcal{F}) : Compatible(C_i, C_j)).$

DEFINITION IL-CONFORMED MULTIDIMENSIONAL SPACE. A multidimensional space is *conformed*, if it enforces sharing of only conform category types:
 $Conformed(Space(\mathcal{F})) \Leftarrow \forall C_i \in Space(\mathcal{F}) : (\nexists C_j \in Space(\mathcal{F}) : Conform(C_i, C_j)).$

As can be seen from the set of definitions in this section, the IL model spans a rather broad set of elements and multidimensional properties by introducing the concept of hierarchically structured dimension schemes. This fact is reflected in the IL construct set of X -DFM, which includes all types of dimension category nodes and roll-up relationships. The only unsupported components in the IL are measure and property attributes.

LOWER LEVEL MODEL

The LL model lowers the abstraction of the defined multidimensional properties in two major respects, namely, *i*) defining the term “measure” as an attribute of a fact and *ii*) decomposing categories into their constituting attributes. This refinement is reflected in the according redefinition of the terms “fact” and “category type”.

DEFINITION LL-FACT. A *fact* F is a collection of uniformly structured data entries over a fact scheme \mathcal{F} , where \mathcal{F} is defined as a pair $(\mathcal{M}^{\mathcal{F}}, \mathcal{D}^{\mathcal{F}})$, where $\mathcal{M}^{\mathcal{F}} = \{M_j, j = 1, \dots, m\}$ is a set of measures and $\mathcal{D}^{\mathcal{F}} = \{D_i, i = 1, \dots, n\}$ is a set of corresponding dimension schemes.

The above fact definition supersedes the concept of a fact cluster in the UL and IL by assigning all measures of the same dimensionality to the same fact. Fact scheme PURCHASE from Figure 3.2 can now be formally defined in terms of its measure set $\mathcal{M}^{\text{PURCHASE}} = \{ \text{amount, number_of_items} \}$ and a set of dimension schemes $\mathcal{D}^{\text{PURCHASE}} = \{ \text{funding, product, project, time, unit, invoice_number} \}$.

The definition of a category type is modified as to specify its constituting attributes:

DEFINITION LL-CATEGORY TYPE. A *category type* \mathcal{C} is a pair $(\mathcal{A}^{\mathcal{C}}, \mathcal{A})$ where $\mathcal{A}^{\mathcal{C}}$ is the distinguished *dimension level* attribute, whereas $\mathcal{A} = \{ \mathcal{A}_r, r = 1, \dots, x \}$ is a set of *property* attributes belonging to \mathcal{C} and functionally dependent on $\mathcal{A}^{\mathcal{C}}$.

A category type \mathcal{C} has exactly one dimension level attribute $\mathcal{A}^{\mathcal{C}}$, which represents the essential characteristic (key property) of that category type. The entire set of attributes defines the structure of the category's members. With the above definition, members in a category of type \mathcal{C} are not atomic values, but tuples of values over the schema $\{ \mathcal{A}^{\mathcal{C}}, \mathcal{A} \}$.

Members of a dimension level attribute are required to be unique (key constraint) whereas the set of property attributes is functionally dependent from the former. Consequently, the relationship between the dimension level attribute and any of its properties is many-to-one or even one-to-one, i.e., it produces the same hierarchical behavior as a roll-up relationship between two categories. Technically, that implies that any property \mathcal{A}_r in \mathcal{C} can be alternatively modeled as a parent category type of \mathcal{C} . In the literature, there are no strict guidelines as to which of the modeling alternatives is to be preferred. However, from the semantic perspective, property attributes are declared as non-hierarchical characteristics of a specific category type and, as such, they may not be used as aggregation levels in OLAP queries. Considering this established rationale, we suggest that the parent-level option is “safer”, whenever the expected query pattern of an attribute is unknown a priori. However, if there relationship of a property to the dimension level attribute is one-to-one, it is not worth representing it as a parent category type.

Figure 3.10 demonstrates the two alternatives of dealing with property attributes in the LL model at the example of project dimension: in (a) manager is a property of the bottom category project and in (b) the former is promoted to a parent hierarchy level of the latter. In X-DFM, the dimension level attribute is implicitly shown as the name of the respective category node, whereas properties are modeled as attribute nodes attached to the respective category.

The definitional framework of the LL model provides the necessary level of semantic detail for identifying various classes of dimensions and facts, presented in the following two chapters.

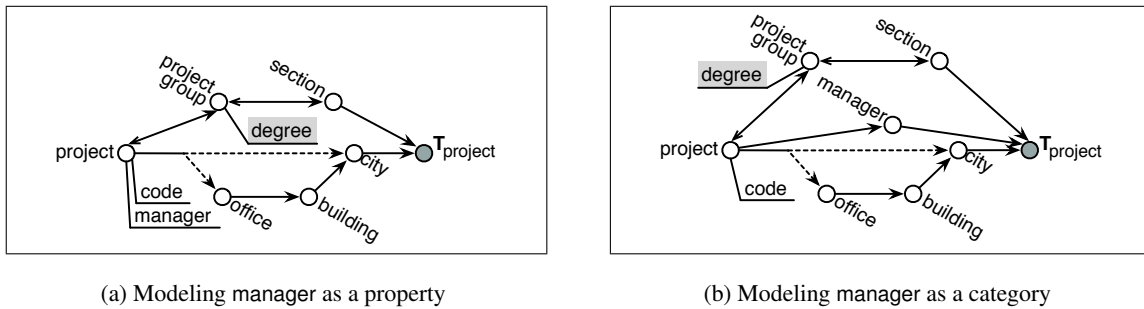


Figure 3.10: Alternatives of dealing with property attributes in the LL model

Chapter 4

Dimensions and Hierarchies in the Multidimensional Data Model

THIS CHAPTER FOCUSES on extending the multidimensional data model to handle complexity in OLAP dimensions and dimension hierarchies. We show that the underlying constraints of the conventional model, such as homogeneity, strictness, regularity, and others, appear too rigid for many real-world applications and, therefore, have to be overcome at the conceptual and, subsequently, at the logical level in order to provide adequate OLAP support for those applications. We present a formal framework for classifying dimension and hierarchy types and demonstrate the feasibility of the proposed extensions at a real-world case study from the domain of academic administration.

Contents

4.1	State of the Art of Dimensional Modeling	59
4.1.1	Rigidity of OLAP Dimensions	60
4.1.2	Related Work on Modeling Dimension Hierarchies	60
4.2	Academic Management as the Motivating Case Study	62
4.3	Categorization of Dimension and Hierarchy Types	66
4.3.1	Refining the Formal Framework	66
4.3.2	Dimension Types	68
4.4	Classification of Hierarchy Types	70
4.4.1	Strict vs. Non-Strict Hierarchies	71
4.4.2	Types of Homogeneous Hierarchies	73
4.4.3	Types of Heterogeneous Hierarchies	75
4.5	Classification of Multiple Hierarchies	84

4.1 State of the Art of Dimensional Modeling

Dimension hierarchy is a central concept in OLAP as it specifies valid aggregation paths for exploring the facts in a data cube at different levels of detail and in a hierarchical fashion: typically, data analysis evolves

from a more abstract view of coarsely-grained aggregates to a more precise view obtained through a sequence of drill-down and slice&dice operations. In OLAP tools, dimension schemes and instances are presented in the form of hierarchical data navigation structures, which enable purely visual specification of analytical queries: the user simply navigates to the desired categories and their members. Abstraction of a database query into a navigation interface raises the issue of reachability: each lower-level member must be reachable from each of its containing members at the upper levels. Another important issue is correct summation: subtotals produced by drilling down should sum up to the value of the decomposed total. The above issues are well investigated in the literature and are formalized as the concepts of aggregate navigation and summarizability.

4.1.1 Rigidness of OLAP Dimensions

Rigidness of the standard OLAP technology is primarily due to the requirement of *summarizability* for all dimension hierarchies. The concepts of summarizability and well-formedness were coined by Rafanelli and Shoshani [152] in the context of statistical databases and redefined by Lenz and Shoshani [98] for OLAP. Summarizability guarantees correct aggregation and optimized performance, as any aggregate view is obtainable from a set of precomputed views of superior granularity. However, the hierarchies in many real-world applications are not summarizable and, therefore, may not be used as dimensions in their original form. In case of minor irregularities, the data tree can be balanced by filling the “gaps” with artificial nodes. In highly unbalanced hierarchies, however, suchlike transformations may be undesirable. Yet in other scenarios, e.g., in taxonomy-based classifications, it is crucial to preserve the original state of hierarchical structures.

Another limitation of OLAP comes from enforcing *uniform granularity*, or precision, in the members of the same category type. As stated in [145], in some applications the data may be prone to naturally varying precision, e.g., the diagnosis of a patient, whereas in other scenarios mixed granularity arises as a result of combining data from different sources. These variations are eliminated by “cleansing” the data prior to loading it into a data warehouse.

The requirement of *completeness* prohibits missing (NULL) values in facts and dimensions, as those values aggravate the invocation of aggregate functions and correct interpretation of computed aggregates.

Correct aggregation is also enforced via the requirement of *homogeneity*. Even though it is admissible to define multiple hierarchies within the same dimension, each of those hierarchies must be homogeneous, i.e., each level of the tree corresponds to a single category and all members of a given category have ancestors in the same set of categories [61].

Analysts are frequently confronted with data, which violates the above restrictions and which, therefore, cannot be adequately supported by standard OLAP systems.

4.1.2 Related Work on Modeling Dimension Hierarchies

Despite the overall maturity and successful establishment of the OLAP technology, the domain of conceptual design still faces a lot of research challenges. New issues arise when this technology, tailored primarily towards the needs of business performance analysis, is applied in novel contexts that deal with rather complex data that cannot be trivially rearranged into homogeneous cubes of uniformly grained facts and perfectly balanced dimension hierarchies.

Early works on multidimensional data models, such as Kimball’s star schema model [81], cube operator of Gray et al. [51], the conceptual model for OLAP of Gyssens and Lakshmanan [52], and the cube data model of Datta and Thomas [36] tightly coupled the conceptual model with the logical, especially the relational one, considering dimension hierarchies as mere collections of attributes used as grouping criteria for aggregating measures. The underlying star schema approach trades semantic richness off against simplicity by not capturing hierarchical relationships at the schema level.

A step in the right direction was achieved by a concurrent branch of research on so-called structured cube models, in which dimension hierarchies are explicit in the scheme.

Li and Wang [99] introduced the notion of a *grouping relation* to reflect hierarchical relationships between the attributes of a dimension and defined *grouping algebra* as an extension of relational algebra that includes order-oriented and aggregation operators.

Agrawal et al. [5] also proposed a model along with a set of algebraic operators for multidimensional databases. Their model provides support for multiple hierarchies in a dimension and symmetric treatment of dimensions and measures by means of PUSH and PULL operators. However, the model does not distinguish between structure and contents, which results in the necessity of encoding structural and functional information into the query.

Cabibbo and Torlone [16] proposed a formal multidimensional model, which is truly implementation-independent and thus provides clear distinction between practical and conceptual aspects. Complex structured dimensions are modeled by specifying a partial order on dimension levels, thus accounting for the possibility of multiple aggregation paths. In the follow-up work [17], the authors extend their formal framework by presenting an algebraic and a graphical language for querying multidimensional databases.

Vassiliadis [179] provides a formal model that defines dimensions as lattices of levels, in which each path is a linear, totally ordered list of levels. The model also includes a set of useful cube operations. Furthermore, the authors present mappings of the conceptual model and its operators to an extended relational model and to multidimensional arrays.

A powerful approach to modeling dimension hierarchies along with SQL query language extensions called $SQL(\mathcal{H})$ was presented by Jagadish et al. [70]. The notion of a dimension is formalized by introducing the concepts of a *hierarchical domain*, *hierarchy*, *hierarchy scheme*, and *dimension scheme*. $SQL(\mathcal{H})$ does not require data hierarchies to be balanced or homogeneous. The data model allows capturing of structural heterogeneity at schema level.

The necessity of dropping the restriction of homogeneity has been recognized by the researchers who proposed respective extensions in form of dimension constraints [61], multidimensional normal forms [94, 96], transformation techniques [144], and mapping algorithms [109].

Hurtado and Mendelson contributed a series of works on summarizability for heterogeneous hierarchies. In [60], a class of constraints for inferring summarizability in a particular class of heterogeneous dimensions is presented. A follow-up work [61] proposes a class of integrity constraints and schemes that enable reasoning about summarizability in general heterogeneous dimensions. A more recent work [59] summarizes previous contributions of the authors and provides a survey of related work on heterogeneity in OLAP.

Lehner et al. [96] relaxed the condition of summarizability to enable modeling of generalization hierarchies by defining a Generalized Multidimensional Normal Form (GMNF) as a yardstick for the quality of multidimensional schemata. Lechtenböcker and Vossen [94] pointed out the methodological deficiency in deriving multidimensional schema from the relational one and extend the framework of normal forms proposed in [96] to provide more guidance in data warehouse design.

In spite of numerous contributions and competing multidimensional models, there is still no consensus in the community about the modeling standards and especially what concerns different hierarchy types.

Pedersen et al. [145] proposed an extended multidimensional data model for handling complex data and a corresponding algebra for multidimensional objects. Model extensions in part of dimension hierarchies address variable granularity due to imprecision, missing values, non-covering, non-strict, and non-onto hierarchies. The issue of heterogeneity is not considered by this model. Innovative features, such as handling imprecision and incompleteness, are supported by means of the proprietary algebra, which has not been implemented in any OLAP system.

Some researchers propose to handle complex hierarchies at the logical schema level. Bauer et al. [9] exploit the flexibility of the relational and, alternatively, the object-relational database constructs to obtain

normalized OLAP schemes. Lin et al. [101] describe an object-relational modeling approach for warehousing complex data. Complexity is handled by employing inheritance and complex objects.

The conceptual design approach of Hüsemann et al. [64] builds upon the multidimensional normal form framework of [96] and investigates the properties of multiple hierarchies, subdividing them into optional (i.e., generalized) and alternative ones. Non-balanced hierarchies are not considered by this framework. Pourabbas and Rafanelli [148] propose a characterization of hierarchy types, distinguishing between total and partial hierarchies, derived hierarchies, and different types of multiple hierarchies within a dimension. The model admits existence of irregular hierarchies, however, not discriminating between different kinds of irregularity.

Abelló et al. [1] proposed a conceptual multidimensional model YAM² based on the UML notation. The model provides a sound framework for classifying various hierarchy types, e.g., symmetric, non-strict, derived, multiple alternative, and generalized hierarchies. Besides, the model also formalizes the types of relationships between correlated dimensions, distinguishing between generalization, aggregation, and derivation. A conceptual model with very rich multidimensional semantics was proposed by Luján-Mora et al. in [104]: dimension hierarchies are classified using the properties of strictness, completeness, degeneration, optionality, multiplicity, derivation, etc. However, the model focuses on the graphical presentation of the extended set of constructs using UML rather than on formal aspects.

The works of Malinowski and Zimányi focus on extending the conceptual model to handle complex hierarchies, not addressed by standard OLAP systems. [108] presents a conceptual classification of hierarchies and proposes a graphical notation based on the E/R model. In [109], the authors formalize and extend their framework, presenting it as *MultiDimER*, a conceptual multidimensional model, and a relational mapping of all defined hierarchy types. The authors evaluate various approaches to enforcing summarizability by transforming complex hierarchies and propose their own mappings, which overcome the deficiencies of previously proposed solutions. However, the author's own conceptual model is presented in a rather informal fashion compared to other models of the state-of-the-art. Besides, this work does not consider the implications of extending the data model on the applicability of OLAP operators.

4.2 Academic Management as the Motivating Case Study

Conceptual extensions of the multidimensional model presented in this chapter emerged as a result of a collaborative effort of our research team with SuperX Project ¹. *SuperX* is the name of the data warehouse system for university administration, developed specifically for the needs of German public higher education sector and patronized by the Ministry of Education as a standardized reporting environment for public universities throughout Germany. Originated at the University of Karlsruhe and further developed at the University of Duisburg-Essen, SuperX has been officially taken over by HIS Ltd. ². HIS Ltd. is the principal provider of software solutions for university management (e.g., facilities, personnel, finance, teaching and research, time-scheduling, etc.) in Germany. SuperX automatically extracts operational data from various HIS systems, cleanses and transforms it into multidimensional cubes and makes the latter available for querying and reporting to decision-makers and other staff.

SuperX appeared an attractive and promising cooperation partner for our research for a number of reasons, such as the following:

- ◆ SuperX is an open-source product available free of charge.
- ◆ SuperX is being increasingly adopted by universities throughout Germany.

¹Project homepage: <http://www.superx-projekt.de/>

²Higher Education Information System Ltd., <http://www.his.de/english/>

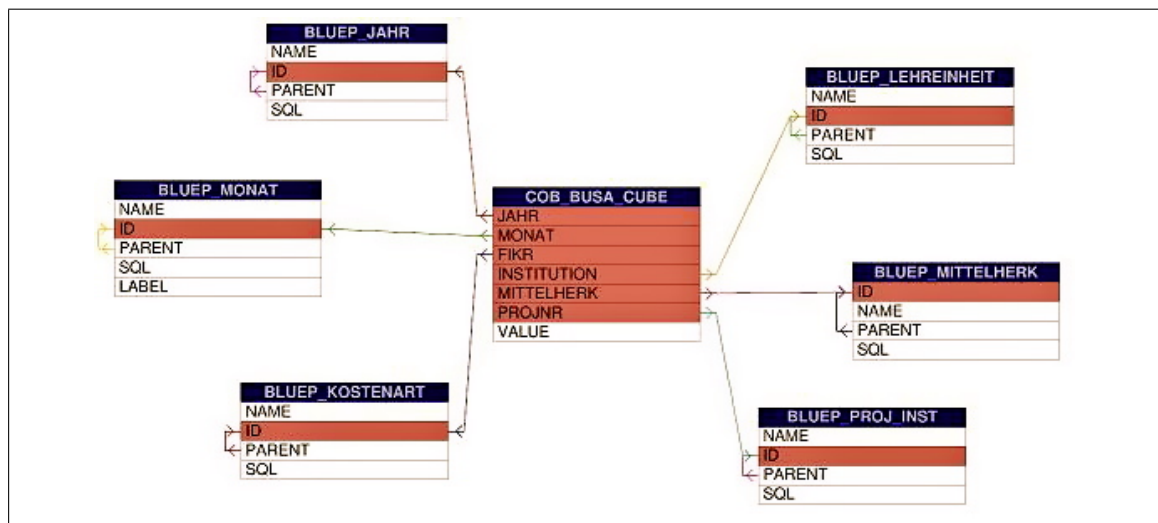


Figure 4.1: Original SuperX fact table COB_BUSA_CUBE (university expenditures) and its dimension tables

- ◆ SuperX is platform-independent and does not require any commercial software to run.
- ◆ SuperX is flexibly configurable with respect to the overall architecture (e.g., DBMS, web server).
- ◆ SuperX has numerous open issues, predominantly due to complex and/or inconsistent data delivered by some of the operational data sources

In addition to the above issues, we intended to enrich the functionality of SuperX by integrating our own software module *UniCap* [117, 183, 184], which is a decision support system for managing the admission capacity and teaching resource utilization in universities.

The SuperX team granted us access to test data extracted from HIS systems COB (cost and activity accounting) and SOC (student admission and performance records). The data was available in the relational form as 2 fact tables with 14 (partially shared) dimension tables. The cubes are as follows:

1. COB_BUSA_CUBE contains the household data, i.e., purchases of various administrative units.
2. STUD_ALLG_CUBE contains overall student statistics, broken down by age, sex, origin, semester, major, etc., i.e., grouped into cohorts rather than as individual records.

The original logical schemata of the above cubes are depicted in Figures 4.1 and 4.2, respectively: primary key attributes are shown with red background and arrows point to the targets of the foreign key relationships.

At the first glance, the logical schemata shown in Figures 4.1 and 4.2 seem to follow the star schema design – each dimension is stored in a single relation. Notice, however, that most of the dimensions have exactly the same set of attributes, namely {ID, NAME, PARENT, SQL}, with ID being the primary key and PARENT being a self-reference foreign key. In terms of the star schema design, those dimensions appear to contain no hierarchies. However, that is not the case as SuperX stores hierarchical relationships as a reference to the parent record (foreign key PARENT) – a well-known approach to storing recursive structures in relational databases, but not in data warehouses.

Table 4.1 shows an example of storing the hierarchy of the university's administrative units using the parent reference. The resulting data hierarchy is shown in Figure 4.3. Since the relational representation provides no clue about hierarchy levels, the node type information mapped to the nodes' background color had to

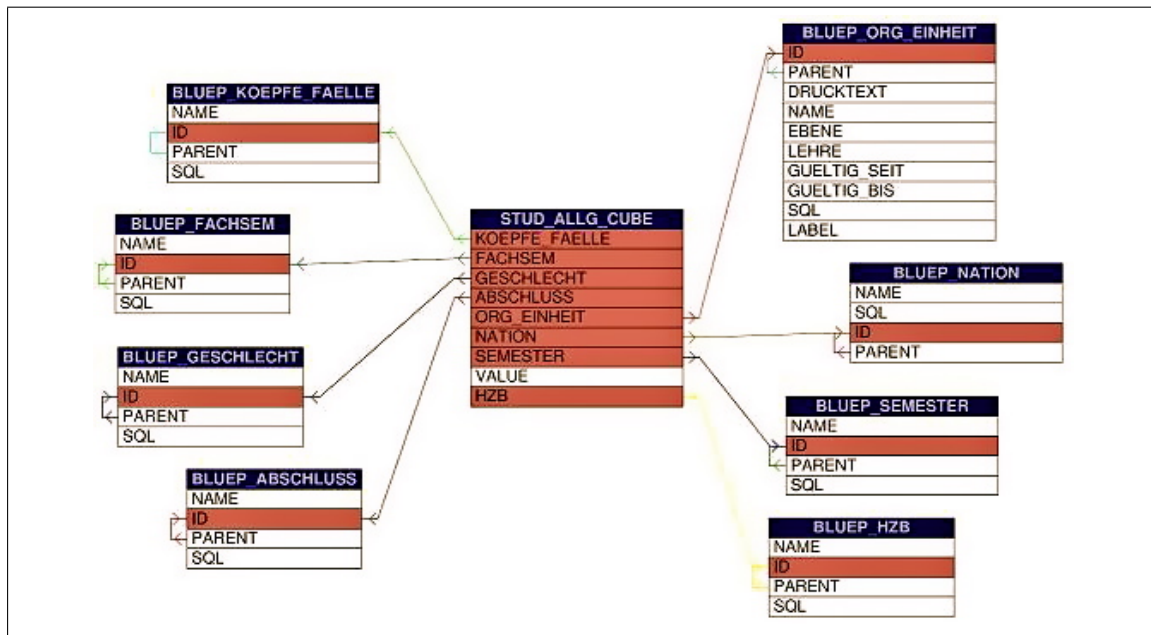


Figure 4.2: Original SuperX fact table STUD_ALLG_CUBE (student enrollments) and its dimension tables

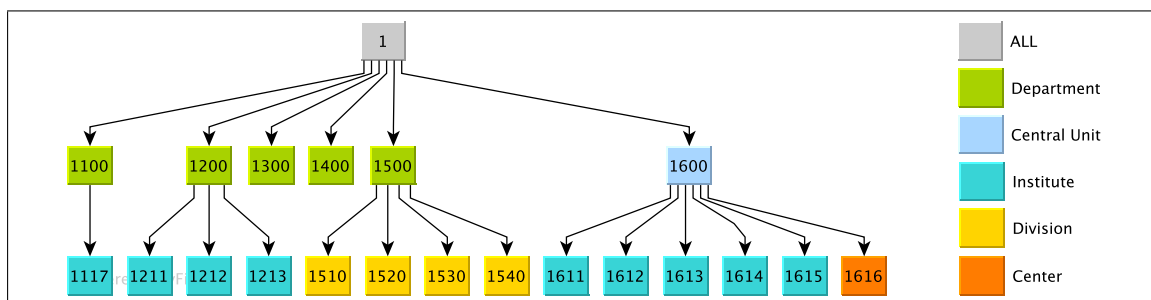


Figure 4.3: Organization hierarchy (fragment) of a university stored in Table 4.1

be extracted manually from the values of the attribute NAME. Without this tediously obtained node semantics it would be impossible to lay the hierarchy out into levels and, consequently, impossible to characterize or interpret it. Back to the data tree in Figure 4.3, based on the node type assignments, we can characterize it as heterogeneous (multiple category types at the same level) and asymmetric (childless non-bottom nodes).

We denote the above approach to storing dimension hierarchies a “*pseudo-star*” schema: the entire hierarchy is put into a single table, as in the star schema, however not by means of de-normalization, but rather by referencing the parent element. An attempt to map the above tiny organization hierarchy to a summarizable OLAP dimension would fail at the very bottom level as even that level consists of three node types and, therefore, may not be mapped to a single category. In the SuperX database, this kind of complex hierarchy structures is rather typical and is encountered in product and cost categorizations, personnel and organisation

Table 4.1: Example of a non-structured hierarchy storage

ID	NAME	PARENT
1	ALL	NULL
1117	Institut für Entwicklung und Frieden	1100
1211	Institut für Kulturwissenschaften	1200
1213	Institut für Fremdsprachliche Philologie	1200
1510	Abteilung für Elektrotechnik u. Informationstechnik	1500
1612	Institut für niederrheinische Kulturgeschichte	1600
1613	Institut für Verkehr und Logistik (IVL)	1600
1614	Institut für Automation und Robotik	1600
1615	Institut für Informatik	1600
1616	Zentrum für Lehrerbildung	1600
1100	Gesellschaftswissenschaften	1
1200	Geisteswissenschaften	1
1212	Institut für Germanistik	1200
1600	Zentrale wiss. Einrichtungen	1
1300	Wirtschaftswissenschaft	1
1400	Naturwissenschaften	1
1500	Ingenieurwissenschaften	1
1520	Abteilung für Maschinenbau	1500
1530	Abteilung für Informatik, Information/Medien	1500
1540	Abteilung für Materialtechnik	1500
1611	Institut für Ostasienwissenschaften	1600

hierarchies, course structures, etc. Besides, the underlying data sources (HIS applications) also employ the same relational storage approach to hierarchical data. Therefore, SuperX does not only avoid the challenges of re-modeling complex data into OLAP dimensions, but also facilitates the ETL process by simply taking over the original data hierarchies as dimensions of the respective facts.

The penalty of storing dimension hierarchies as unstructured data graphs is devastating—the OLAP technology, its operators, query languages, metadata, and frontend tools become fully inapplicable. SuperX provides proprietary end-user tools for analyzing “pseudo-star” schemata. These tools are rather rudimentary in their functionality, e.g., they allow to retrieve data via pre-defined masks or interact with a cube via a pivot table. The provided pivot table interface Joolap³ allows to explore available measures along at most two dimensions (no dimension nesting) whereas other dimensions may be used as filters. Joolap avoids recursive queries by allowing to drill down only one value at a time, e.g., to drill-down from quarters into months, each quarter’s element has to be expanded. Poor query and presentation functionality is the price SuperX pays for supporting non-structured and non-summarizable hierarchies.

In our opinion, the “pseudo-star” schema solution of SuperX is too disadvantageous due to its incompatibility with the established OLAP technology and, hence, inability to benefit from the techniques and tools of the latter. Therefore, we reconsidered the original intention to extend the existing SuperX framework and decided to build a new data warehouse in accordance with the general guidelines of data warehouse design. Complex hierarchies, which cannot be handled by the conventional multidimensional model, inspired the corresponding conceptual extensions of the model and transformation techniques for inferring summarizability in those hierarchies.

³Project homepage: <http://joolap.memtext.de/>

4.3 Categorization of Dimension and Hierarchy Types

A categorization of dimension types and their hierarchy types is obtained by systematically investigating various properties of dimensions and hierarchies at the scheme and at the instance level. To have an adequate example for illustrating diverse types of dimensions presented in this section, we generated a more detailed variant of the sample fact scheme PURCHASE from Chapter 3, shown in Figure 4.4.

Adherence to or violation of a particular multidimensional property (e.g., homogeneity, strictness, balancedness) is used for subtyping dimension hierarchies. The metamodel of the categorization is specified in terms of dimension and hierarchy classes with composition and specialization relationships between them. Since a dimension may consist of multiple hierarchies or be non-hierarchical, we distinguish between dimension and hierarchy classes. Metaclasses are identified starting from more general properties and proceeding to their subclasses and special cases, with the entire resulting categorization scheme depicted in Figure 4.5 in the form of a metamodel. The edges of the classification are labeled by the respective discrimination criteria.

4.3.1 Refining the Formal Framework

The formalization of the conceptual model, presented in Chapter 3, needs to be further refined to define the properties necessary for coping with complex dimension hierarchies. The concepts introduced in this section are definable at the IL and remain valid in the LL model.

A dimension hierarchy is based on a hierarchical characteristic, also referred to as an *analysis criterion*. For instance, in the dimension project (see Figure 4.4), a hierarchy given by $\text{office} \sqsubseteq^{(\text{full})} \text{building} \sqsubseteq^{(\text{full})} \text{city}$ orders projects according to their location criterion, whereas manager creates a supervision hierarchy, and project group groups projects thematically. To classify dimensions according to their hierarchy types, we provide an alternative definition of the concept *dimension* in terms of its constituting hierarchies:

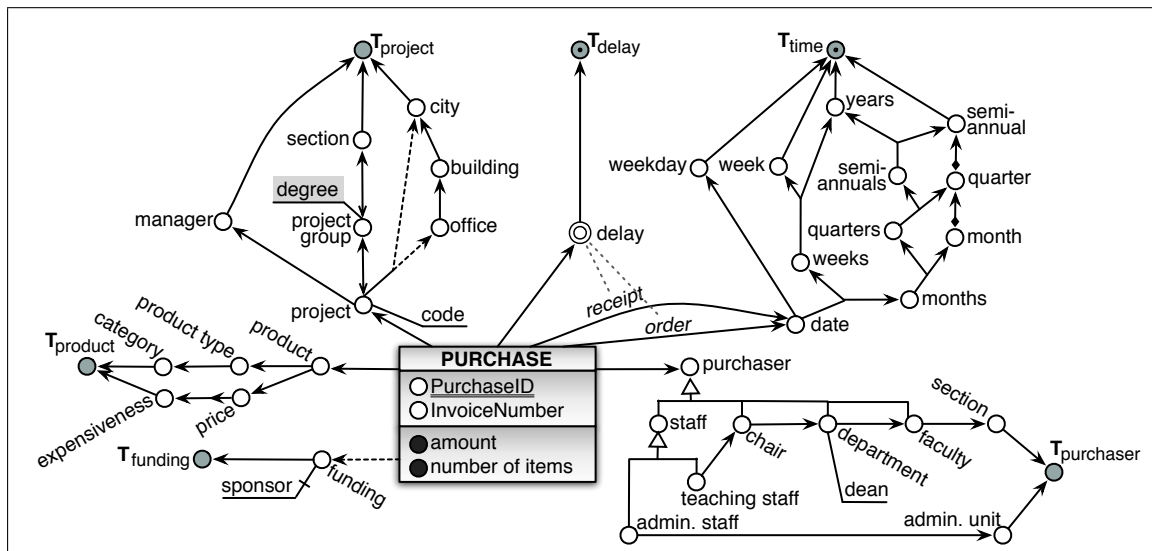


Figure 4.4: Fact scheme PURCHASE with added complexity

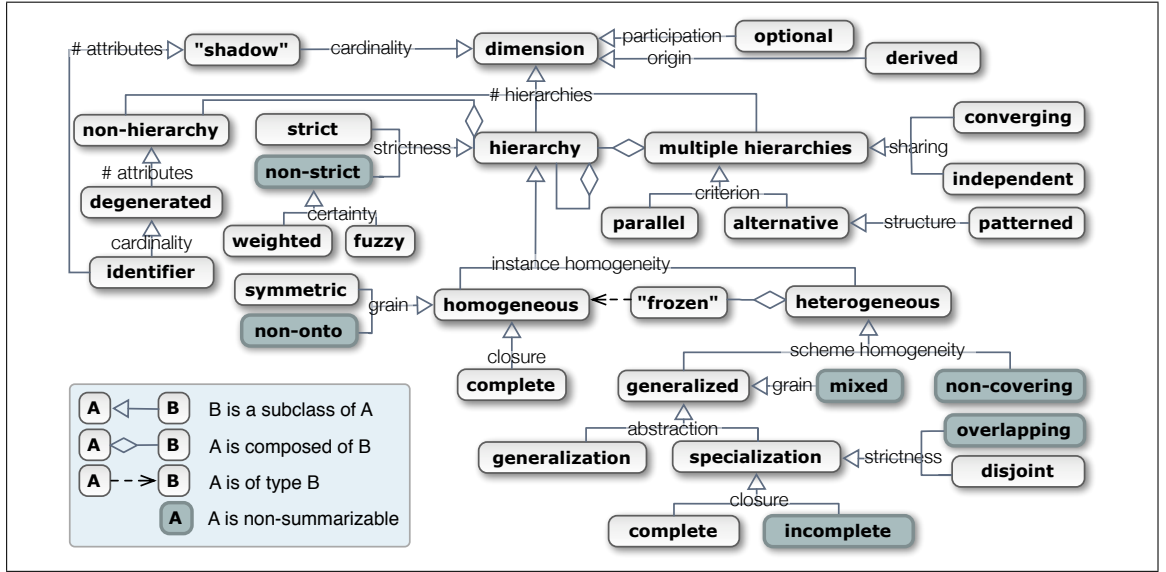


Figure 4.5: Categorization of dimension and hierarchy types

DEFINITION IL-HIERARCHICAL DOMAIN. A *hierarchical domain* is a non-empty set V_H with the only defined predicates $=$ (identity), \subseteq (child/parent relationship), and \subseteq^* (transitive closure, or descendant/ancestor, relationship) such that the graph G_{\subseteq} over the nodes $\{e_i\}$ of V_H is a tree.

In OLAP, only *structured* hierarchies, i.e., those adhering to a certain scheme, qualify as dimensions. A dimension may be organized into one or multiple hierarchies to provide additional aggregation levels.

DEFINITION IL-HIERARCHY SCHEME. A *hierarchy scheme* \mathcal{H} within a dimension scheme \mathcal{D} is a quadruple $\mathcal{H} = (\mathcal{C}^{\mathcal{H}}, \sqsubseteq_{\mathcal{H}}, \top_{\mathcal{D}}, \perp_{\mathcal{D}})$, where $\sqsubseteq_{\mathcal{H}}$ is a restriction of $\sqsubseteq_{\mathcal{D}}$ and each category has at most one outgoing full roll-up relationship: $\nexists (C_i, C_j, C_k) \in \mathcal{C}^{\mathcal{H}} : C_i \sqsubseteq^{\text{full}} C_j \wedge C_i \sqsubseteq^{\text{full}} C_k$, i.e., no category is allowed to have multiple full roll-up relationships.

Thereby, any hierarchy in a dimension has the same bottom and top category types as the dimension itself. According to the above definition, a scheme is guaranteed to yield a single hierarchy by excluding multiple full roll-up relationships of the same category type. Notice that it does not prohibit heterogeneous schemes produced by related partial roll-ups. The *instance* of a hierarchy results from specifying the members in each category and the partial order between them.

DEFINITION IL-HIERARCHY INSTANCE. A *hierarchy instance* H associated with hierarchy scheme \mathcal{H} is a pair $H = (C^H, \sqsubseteq_H)$, where $C^H = \{C_j, j = 1, \dots, m\}$ is a set of categories such that $\text{Type}(C_j) = C_j, C_j \in \mathcal{C}^{\mathcal{H}}$, and \sqsubseteq_H is a partial order on $\bigcup_j C_j$, the union of all dimensional values in the individual categories.

A *dimension extension*, or *instance*, E associated with dimension scheme \mathcal{D} is a merge graph of all its hierarchy instances H_i . Dimension scheme \mathcal{D} can similarly be obtained by merging its hierarchy schemes, however, in each case of multiple outgoing roll-up relationships, it is necessary to specify whether those

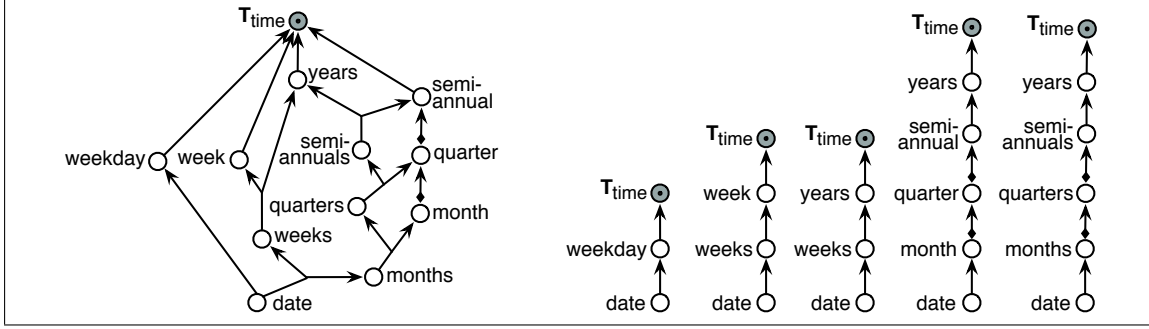


Figure 4.6: Dimension time (left) as a merge graph of its hierarchies (right)

are multiple alternative or parallel hierarchies. Dimension scheme is decomposable into a set of hierarchy schemes $\mathcal{H}^{\mathcal{D}} = \{\mathcal{H}_i, i = 1, \dots, n\}$. Figure 4.6 illustrates the concept of obtaining the dimension scheme by merging the schemes of all hierarchies found in that dimension, at the example of dimension time.

DEFINITION IL-DIMENSION SCHEME. A dimension scheme \mathcal{D} is a directed graph constructed by merging the graphs of all hierarchy schemes associated with \mathcal{D} . The set of vertices in \mathcal{D} is a duplicate-free union of category types from $\mathcal{H}^{\mathcal{D}}$ and the edges are the union of the partial orders from $\mathcal{H}^{\mathcal{D}}$. \mathcal{D} contains each scheme $\mathcal{H}_i \in \mathcal{H}$ as its subgraph.

The above dimension definition admits a broad variety of hierarchical behaviors within a dimension. For instance, it is not prohibited to have multiple hierarchies, shared category types, or partial roll-up relationships.

Decomposition of complex dimension schemes into their constituting hierarchy schemes is crucial for determining valid aggregation paths. Consider dimension project in Figure 4.4. Apparently, it is composed of multiple hierarchy schemes with the following sets of category types:

1. $\mathcal{H}_1 = \{\perp_{\text{project}}, \text{project group}, \top_{\text{project}}\}$,
2. $\mathcal{H}_2 = \{\perp_{\text{project}}, \text{manager}, \top_{\text{project}}\}$,
3. $\mathcal{H}_3 = \{\perp_{\text{project}}, \text{office}, \text{building}, \text{city}, \top_{\text{project}}\}$,
4. $\mathcal{H}_4 = \{\perp_{\text{project}}, \text{building}, \text{city}, \top_{\text{project}}\}$.

Multiple hierarchies in a dimension exist whenever its scheme contains a category that rolls-up to more than one destination. We distinguish between heterogeneous and truly multiple hierarchies. In heterogeneous hierarchies, multiple paths result from partial related roll-up edges, such as in the project location hierarchy, in which the members of project have parent members either in office or directly in the next-level category building. Therefore, hierarchies \mathcal{H}_3 and \mathcal{H}_4 can be considered parts of a single heterogeneous hierarchy.

4.3.2 Dimension Types

Dimension classes are identified by inspecting such properties as *i)* the cardinality of its relationship with the fact, *ii)* fullness of its participation in the fact, *iii)* derivability of its members, and *iv)* the number of constituting hierarchies. Figure 4.7 shows the fragment of the metamodel corresponding to dimension categorization, described in this section.

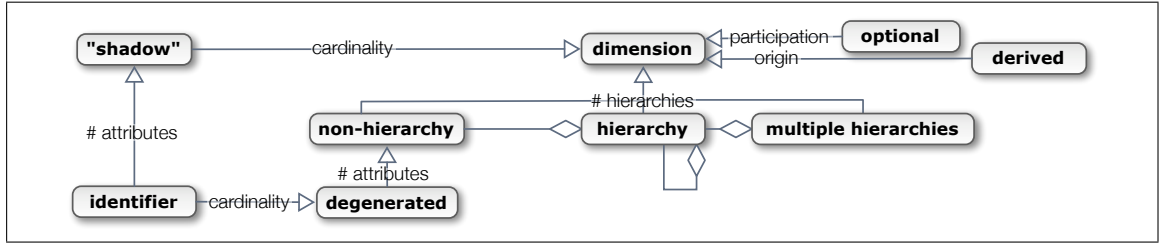


Figure 4.7: Categorization of dimension types

Cardinality of fact-dimensional relationship. Typically, a relationship between a fact and any of its dimensions is many-to-one, i.e., each fact maps to exactly one member in each dimension. Degeneration of a fact-dimensional roll-up to a one-to-one relationship produces a so-called “shadow” dimension [45]. Figure 4.8 shows three possible cases of modeling “shadow” dimensions in *X*-DFM at the example of a degenerated roll-up relationship between the fact PURCHASE and its dimension purchase record: (a) shows purchase record as a degenerated dimension, i.e., consisting of a single attribute, and, therefore, qualifying as a fact identifier of PURCHASE, (b) shows purchase record as a non-hierarchical non-degenerated dimension, i.e., consisting of multiple attributes, whereas (c) presents purchase record as a hierarchy. We denote a one-to-one fact-dimensional relationship a *degenerated roll-up* and represent it by a non-directed edge in *X*-DFM, as in Figures 4.8b and 4.8c.

Participation in the fact-dimensional relationship. The roll-up relationship between a fact and a dimension may be *full* or *partial*. Partial containment occurs when fact entries are not required to have the respective dimensional characteristic, i.e., admit NULL values in its place. A partially contained dimension is called *optional* with respect to the fact. In Figure 4.4, funding is an optional dimension, as indicated by the incoming partial roll-up edge towards its bottom category.

Derivability. A dimension, in which the members of the bottom category are derived from one or several other dimension categories, is called *derived*. In some scenarios, it may be feasible to model derived dimensions explicitly, if those are expected to be used frequently or if the frontend tools do not support ad hoc dimension specification. In fact scheme PURCHASE in Figure 4.4, an example of such a dimension is

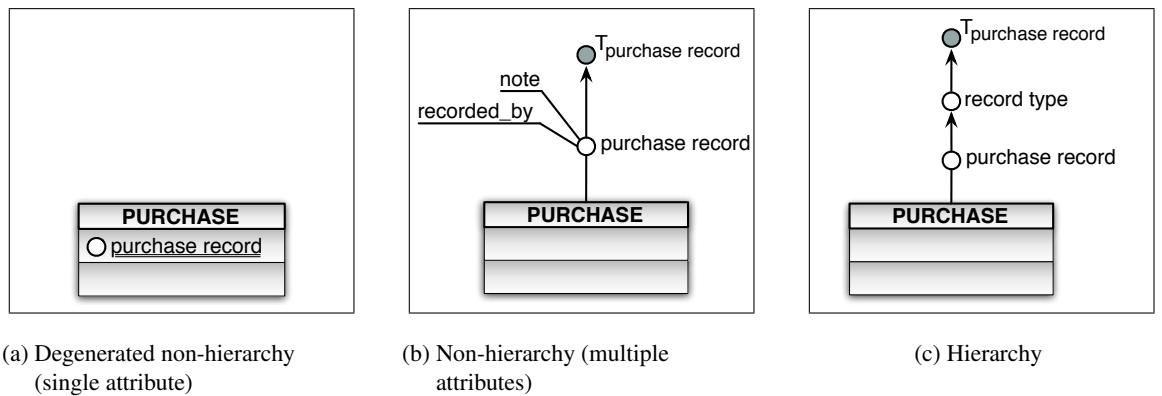


Figure 4.8: Examples of “shadow” dimensions

delay, as its values are derived by subtracting order date from receipt date. Notice that, if modeled explicitly, a derived category may be used just as a normal category, i.e., participate in roll-up relationships, serve as input for deriving further categories, etc.

Number of constituting hierarchies. Considering the number of hierarchies, a dimension can be a *hierarchy*, *non-hierarchy*, or *multiple hierarchies*, with funding, product, and project as the examples of the respective types shown in Figure 4.4.

DEFINITION IL-NON-HIERARCHY. A dimension D is a *non-hierarchy*, if its scheme is composed solely of the bottom and the top category types: $Non-hierarchy(D) \Leftarrow \mathcal{C}^D = \{\perp_D, \top_D\}$.

DEFINITION IL-HIERARCHY. A dimension D is a *hierarchy*, if its scheme consists of exactly one hierarchy: $Hierarchy(D) \Leftarrow |\mathcal{H}^D| = 1 \wedge \exists C_i \in \mathcal{D} : C_i \neq \perp_D \wedge C_i \neq \top_D$.

DEFINITION IL-MULTIPLE HIERARCHIES. A dimension D represents *multiple hierarchies*, if its scheme contains more than one hierarchy: $Multiple(D) \Leftarrow |\mathcal{H}^D| > 1$.

In a non-hierarchical dimension, no hierarchy levels are defined on top of the bottom category. A non-hierarchical dimension that has been “stripped off” to a single category with a single attribute is called *degenerated* dimension!degenerated, and a degenerated dimension is a *fact identifier* dimension!degenerated!fact identifier, if it uniquely identifies each fact entry. Formalization of the above two dimension types is provided in Section 5.2.2 of the next chapter. InvoiceNumber and PurchaseID are examples of a degenerated and a fact identifier dimension, respectively, in fact scheme PURCHASE in Figure 4.4.

A dimension with at least one roll-up relationship (besides the default roll-up to the abstract root category) is a hierarchical one. The remaining two sections of this chapter elaborate on the types of single and multiple hierarchies, respectively.

4.4 Classification of Hierarchy Types

Similarly to the way a hierarchical structure can be recursively decomposed into subtrees, hierarchical dimensions can be decomposed into constituent subdimensions. Intuitively, D_j is a subdimension of D_i , if both the scheme and the extension of the former are subgraphs in the scheme and the extension of the latter. Horizontal decomposition into subdimensions is done by recursively stripping off the bottom level. For example, a project location hierarchy $project \sqsubseteq^{(part)} office \sqsubseteq^{(full)} building \sqsubseteq^{(full)} city \sqsubseteq^{(full)} \top_{project}$ consists of the following subdimensions:

- ◆ $office \sqsubseteq^{(full)} building \sqsubseteq^{(full)} city \sqsubseteq^{(full)} \top_{project}$,
- ◆ $building \sqsubseteq^{(full)} city \sqsubseteq^{(full)} \top_{project}$,
- ◆ $city \sqsubseteq^{(full)} \top_{project}$.

Alternatively, a dimension can be decomposed into subdimensions vertically, i.e., along multiple aggregation paths, as was already shown in Figure 4.6 for the time dimension. Finally, vertical and horizontal decomposition may be used in combination for subdimension extraction. Decomposition of complex dimensions into simple subdimensions is useful for localizing the segments, in which a certain property (e.g., non-strictness or heterogeneity) occurs.

DEFINITION IL-SUBDIMENSION. A dimension D_n is a *subdimension* of D_m , if scheme \mathcal{D}_n is a subgraph in scheme \mathcal{D}_m (i.e., $\mathcal{C}^{\mathcal{D}_n} \in \mathcal{C}^{\mathcal{D}_m}$ and $\sqsubseteq \mathcal{D}_n$ is a restriction of $\sqsubseteq \mathcal{D}_m$) and extension E_n is a restriction of extension E_m to the corresponding categories.

Dimension hierarchies are categorized along two orthogonal properties of *homogeneity* and *strictness*.

Homogeneity of a hierarchy is assessed by testing it for partial containment relationships. Dimension project contains a homogeneous hierarchy $\text{project} \sqsubseteq^{(\text{full})} \text{manager} \sqsubseteq \top_{\text{project}}^{(\text{full})}$. However, the hierarchy $\text{project} \sqsubseteq^{\text{part}} ((\text{office} \sqsubseteq^{(\text{full})} \text{building} \sqsubseteq^{(\text{full})} \text{city}) | \text{city}) \sqsubseteq^{(\text{full})} \top_{\text{project}}$ is heterogeneous as project values are allowed to roll-up either to office or directly to city. This behavior is the result of having two types of projects, namely, the internal ones with an office location, and the external ones located in other cities.

DEFINITION IL-HOMOGENEOUS HIERARCHY. A hierarchy H is *homogeneous*, if all roll-up relationships between its categories are full (all members roll-up along identical paths):

$$\text{Homogeneous}(H) \Leftarrow \nexists (C_i, C_j) \in \mathcal{C}^H : C_i \sqsubseteq^{(\text{part})} C_j.$$

DEFINITION IL-HETEROGENEOUS HIERARCHY. A hierarchy H is *heterogeneous*, if it admits multiple mutually exclusive paths (related partial roll-up relationships) between its categories:

$$\text{Heterogeneous}(H) \Leftarrow \exists (C_i, C_j) \in \mathcal{C}^H : C_i \sqsubseteq^{(\text{part})} C_j.$$

4.4.1 Strict vs. Non-Strict Hierarchies

Generally, dimension hierarchies are assumed to be *strict*, i.e., each roll-up relationship is many-to-one and, therefore, each child member has at most one parent member. Conventional OLAP tools are constrained to aggregating along strict hierarchies to ensure summarizability. However, in many applications non-strict hierarchies are common: an employee may be registered with more than one department, a product may belong to several categories, etc. Figure 4.9 shows the fragment of the metamodel corresponding to the strictness-based categorization of dimension hierarchies.

DEFINITION IL-STRICT HIERARCHY. A hierarchy H is *strict*, if it prohibits multi-parent roll-up relationships between its members:

$$\text{Strict}(H) \Leftarrow \forall (C_i \sqsubseteq C_j) \in C^H, \forall e_1 \in C_i, \forall (e_2, e_3) \in C_j : ((e_1 \sqsubseteq e_2 \wedge e_1 \sqsubseteq e_3) \Rightarrow e_2 = e_3).$$

DEFINITION IL-NON-STRICT HIERARCHY. A hierarchy H is *non-strict*, if it admits multi-parent roll-up relationships between its members:

$$\text{Non-strict}(H) \Leftarrow \exists (C_i \sqsubseteq C_j) \in C^H, \exists e_1 \in C_i, \exists (e_2, e_3) \in C_j : (e_1 \sqsubseteq e_2 \wedge e_1 \sqsubseteq e_3 \wedge e_2 \neq e_3).$$

A non-strict roll-up relationship is expressed by a predicate \sqsubseteq in the formal model and by a bi-directed edge in X -DFM. The stronger arrow head indicates the direction of the roll-up. Notice that a non-strict roll-up is not just a general many-to-many relationship, but a hierarchical (“part-of”) one. In other words, it is

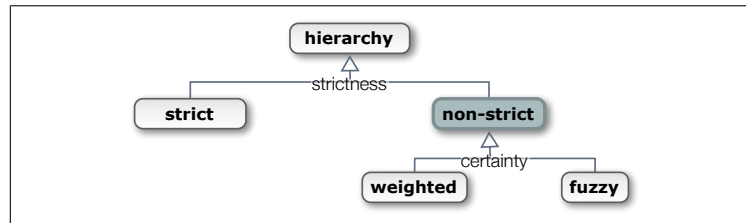


Figure 4.9: Categorization of hierarchy types with respect to strictness

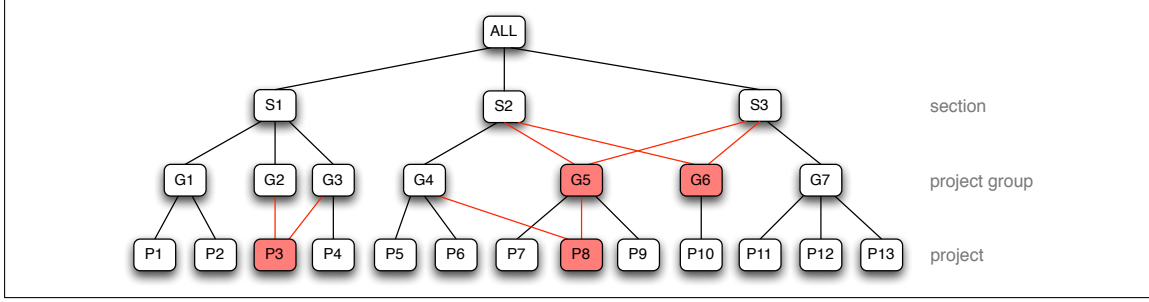


Figure 4.10: A sample instance of a non-strict hierarchy

predominantly a many-to-one mapping with a relatively small portion of multi-parent occurrences (otherwise, it would be a misuse of the term “hierarchy”). An example of a non-strict hierarchy in Figure 4.4 is project $\sqsubseteq^{(\text{full})}$ project group $\sqsubseteq^{(\text{full})}$ section, where a project member may be associated with multiple project group members, and project group members, in their turn, may belong to more than one section value. Figure 4.10 shows a sample instance of this non-strict hierarchy with multi-parent project members ($P3$, $P8$), multi-parent project group members ($G5$, $G6$), as well as multi-parent roll-up edges of those members highlighted with red color.

Non-strict mappings are generally non-summarizable. However, there exist two augmented non-strict hierarchy types that provide correct summarization. Those are weighted and fuzzy hierarchies, which use weights and rules, respectively, to associate the elements of two hierarchy levels.

A *weighted* hierarchy is obtained by specifying each child element’s degree, or probability, of belonging to each of its parent elements. In X -DFM, the weight associated with a non-strict roll-up edge is shown as the child category’s property attribute with grey background color. The relation between project group and section in Figure 4.4 is an example of such mapping, supplied with an obligatory “degree-of-belonging” attribute degree.

In the formal notation, a weighted non-strict roll-up relationship is expressed by a predicate $\sqsubseteq^{(\text{weight})}$ at the scheme level (e.g., project group $\sqsubseteq^{(\text{weight})}$ section) and by super-scripting the \subseteq predicate with the respective weight’s value at the instance level (e.g., $G5 \subseteq^{(0.5)} S2$). Thereby, a strict roll-up relationship $e_1 \subseteq e_2$ is equivalent to a weighted roll-up relationship $e_1 \subseteq^{(1)} e_2$.

DEFINITION IL-WEIGHTED HIERARCHY. A hierarchy H is *weighted*, if the roll-up relationships of its members are supplied with weight values, which express the child member’s degree of participation in that relationship:

$$\text{Weighted}(H) \Leftarrow \forall C_j \in H, \forall e_1, e_2 \in \cup_j C_j, e_1 \subseteq e_2 : (\exists w, 0 < w \leq 1 : e_1 \subseteq^{(w)} e_2).$$

We introduce a function $\text{Weight}(e_1, e_2)$, which returns the weight value w of the relationship $e_1 \subseteq^{(w)} e_2$. The following property holds for a weighted hierarchy and guarantees its summarizability:

$$\forall e_j \in H : \sum_{e_k, e_j \subseteq e_k} \text{Weight}(e_j, e_k) = 1.0.$$

This property states that for any member value e_j , the total weight of all its roll-up relationships sums up to 1.0, and, therefore, the sum of the weighted aggregates at each hierarchy level equals 100%. Figure 4.11 shows the instance of the non-strict mapping between project group and section from Figure 4.10, turned into a weighted hierarchy by supplying the edges with weights (left) and resolving the resulting hierarchy into a non-overlapping tree by splitting multi-parent nodes into multiple single-parent nodes (right).

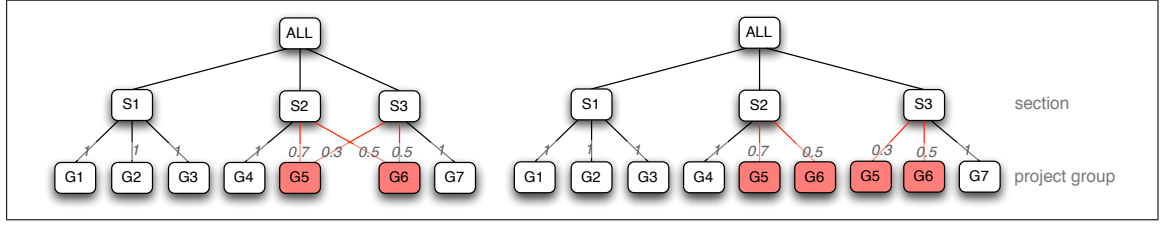


Figure 4.11: Normalizing non-strictness via a weighted hierarchy

Fuzzy hierarchies are a special type of non-strict mappings, in which child elements are assigned to parent elements dynamically using some rules, so that the actual partial order in the hierarchy may vary in time [91]. However, at any single point in time, the mapping is strict and, therefore, summarizable. Consider an example of a fuzzy category *expensiveness* in Figure 4.4: price values are assigned to those in *expensiveness* based on complex rules, e.g., by analyzing the overall price distribution for the products purchased so far.

In the formal notation, a fuzzy roll-up relationship between a pair of categories is expressed by a predicate \sqsubseteq (e.g., $\text{price} \sqsubseteq \text{expensiveness}$). The corresponding X -DFM construct is a roll-up edge with a double arrow. Fuzzy hierarchies for OLAP represent a rather new and immature research field with mostly theoretical contributions. Existing OLAP systems do not support fuzzy hierarchies and no prototype implementations are available yet. Therefore, at this stage we restrain ourselves to providing the constructs for their conceptual modeling with no further discussion of this hierarchy type.

4.4.2 Types of Homogeneous Hierarchies

Homogeneous hierarchies should be tested for symmetry and completeness, as can be seen in the respective fragment of the hierarchy classification, depicted in Figure 4.12.

A hierarchy is *symmetric*, or *onto*, if all its levels are mandatory, i.e., if each non-bottom member has at least one descendant element at the bottom level. Otherwise, the hierarchy is *asymmetric* (*non-onto*).

DEFINITION IL-ONTO HIERARCHY. A hierarchy H is *onto*, if it disallows childless members in non-bottom categories: $\text{Onto}(H) \Leftarrow (\text{Homogeneous}(H) \wedge \forall C_i \in C^H, \text{Type}(C_i) \neq \perp_{\mathcal{H}}, \forall e_1 \in C_i : (\exists e_2, \text{Type}(e_2) = \perp_{\mathcal{H}} : e_2 \sqsubseteq^* e_1))$.

DEFINITION IL-NON-ONTO HIERARCHY. A hierarchy H is *non-onto*, if it admits childless members in non-bottom categories: $\text{Non-onto}(H) \Leftarrow (\text{Homogeneous}(H) \wedge \exists C_i \in C^H, \text{Type}(C_i) \neq \perp_{\mathcal{H}}, \exists e_1 \in C_i : (\nexists e_2, \text{Type}(e_2) = \perp_{\mathcal{H}} : e_2 \sqsubseteq^* e_1))$.

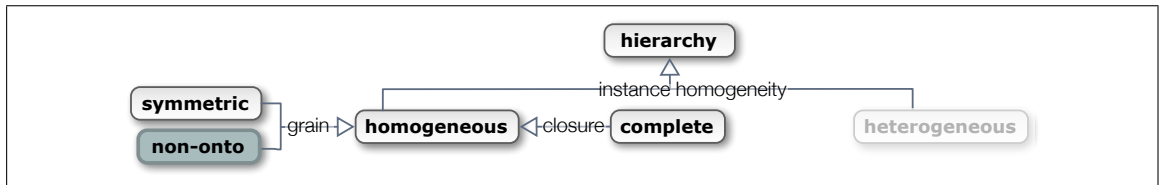
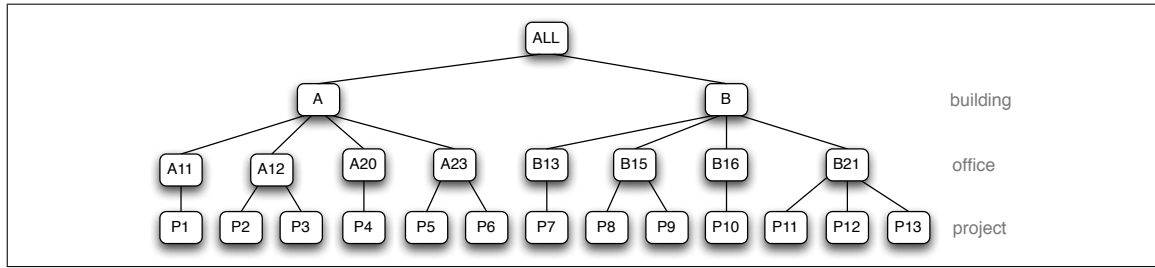
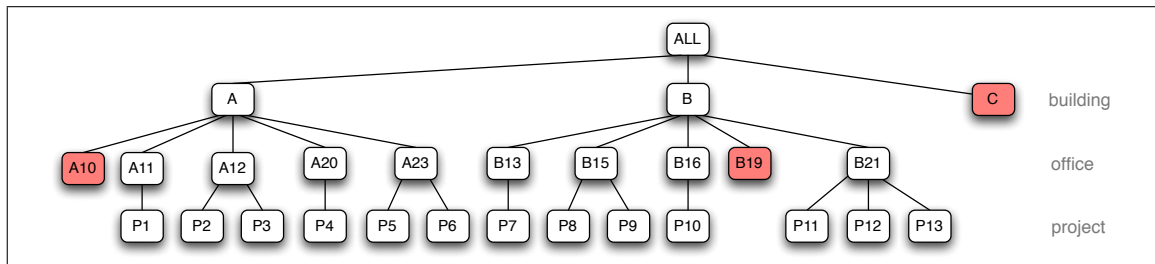


Figure 4.12: Categorization of homogeneous hierarchy types



(a) Project locations as an onto hierarchy



(b) Project locations as a non-onto hierarchy

Figure 4.13: Examples of a symmetric and an asymmetric hierarchy

The hierarchy $\text{project} \sqsubseteq^{(\text{part})} \text{office} \sqsubseteq^{(\text{full})} \text{building}$ is symmetric, as shown in Figure 4.13a. A non-onto variant of this hierarchy can be obtained by including members of office and building categories, which do not serve as project locations. Figure 4.13b shows an example of the resulting asymmetric hierarchy instance, with childless non-bottom members highlighted with red color.

Asymmetry in a hierarchy may have two reasons:

- ◆ **Asymmetry due to dimension conformance.** In a unified multidimensional space, each category type is represented in a non-redundant fashion and is shared by all dimension categories of that type. Therefore, the extension of a category type contains the union of all member values from all conform categories. Each single category, however, is likely to span only a subset of the entire extension. The hierarchy of project locations, depicted in Figure 4.13b, gives an example of this kind of asymmetry: the bottom category project makes use of the existing category types office and building. In such a case, existence of office and/or building members with no child members in project does not lead to non-summarizable behavior because there are no fact entries associated with those members.
- ◆ **Inherent asymmetry** is a result of variable granularity in the hierarchical domain itself. A classical example of inherently asymmetric hierarchies are organisational structures, in which some units are decomposed into more structural levels than others, e.g., some departments are subdivided into units and others are not. This type of non-onto hierarchy is non-summarizable as there may exist fact entries that map to childless non-bottom category members.

Another property of homogeneous hierarchies, orthogonal to symmetry, is that of *completeness*. The term “*complete hierarchy*” is introduced in [104] to denote a hierarchy, in which all child-level members belong to one member of a parent category and that parent member consists of those child members only. For example,

the mapping between month and year is complete because all months together form a year. Awareness of the hierarchy's completeness can be incorporated into a logical schema in form of integrity constraints, such as non-extendibility of existing subtrees. By default, hierarchies are assumed to be *non-complete*.

4.4.3 Types of Heterogeneous Hierarchies

Heterogeneous hierarchies occur whenever members of the same category roll-up along different paths. The term “frozen” dimension is introduced in [61] to denote minimal homogeneous dimension instances representing different structures implicitly combined into a heterogeneous dimension. Typically, a heterogeneous hierarchy is the result of including subtypes that can be represented by a generalization/specialization relationship. In the context of multidimensional modeling, specialization is an opportunity to specify optional attributes, categories and even hierarchies and, hence, is an important mechanism for semantically correct organization of optional hierarchies.

As can be seen in the fragment of the metamodel, depicted in Figure 4.14, heterogeneous hierarchies are subdivided into *non-covering* and *generalized*, based on the criterion of scheme homogeneity.

NON-COVERING HIERARCHY

A *non-covering* hierarchy is obtained by allowing roll-up relationships of a category to be partial, so that its members may skip levels. Non-covering hierarchies have a homogeneous hierarchy scheme, i.e., each category has at most one parent category, however, the members are allowed to skip the parent level and roll up directly to an upper level. Therefore, in the hierarchy instance, each member has a single parent member, however, the path length from the bottom category to the root varies from one member to another.

DEFINITION IL-NON-COVERING HIERARCHY. A hierarchy H is *non-covering*, if it contains an exclusive partial roll-up relationship to a set of categories, which form a hierarchy: $Non-covering(H) \Leftarrow (Heterogeneous(H) \wedge \exists C_i, C_j, C_k \in C^H : (C_i \sqsubseteq^{part} (C_j | C_k) \wedge (C_j \sqsubseteq C_k \vee C_j \sqsubseteq^* C_k)))$.

An example of a non-covering hierarchy scheme is $project \sqsubseteq^{part} (office \sqsubseteq^{full} building \sqsubseteq^{full} city | city) \sqsubseteq^{full} T_{project}$, due to the mutually exclusive partial roll-up relationships of project to office and to city, where $office \sqsubseteq^* city$. A sample instance of this hierarchy is shown in Figure 4.15, revealing the cause of heterogeneity:

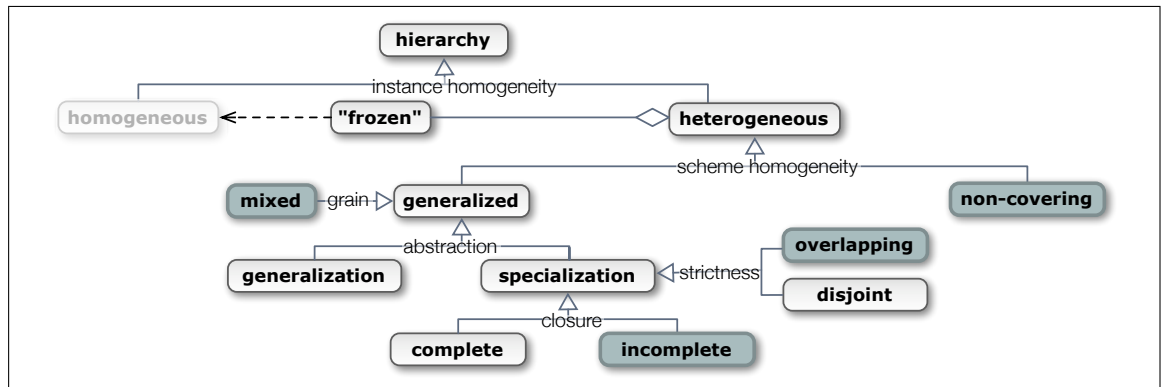


Figure 4.14: Categorization of heterogeneous hierarchy types

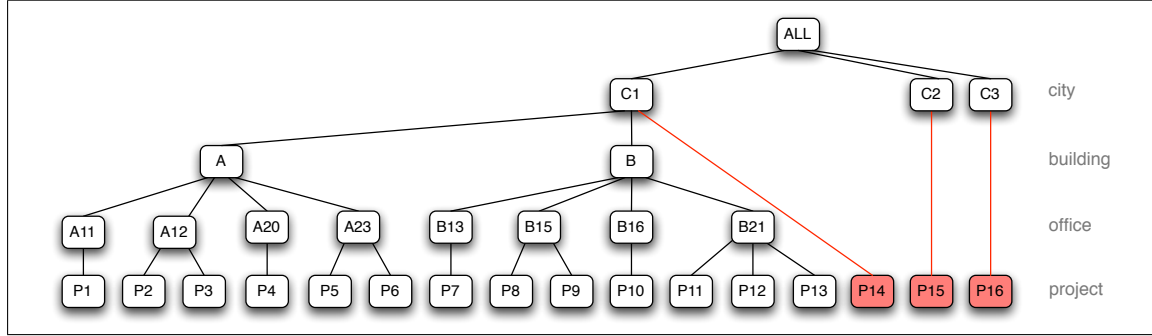


Figure 4.15: Project locations as a non-covering hierarchy

apparently, there exist two types of projects, internal projects located in city *C1* have an office assignment, whereas for external projects only the city is specified.

In the classification of Malinowski and Zimányi [109], non-covering hierarchies are treated as a special case of generalized hierarchies. In our classification, however, the former is not considered generalized as it does not make explicit use of a generalization/specialization relationship. As we proceed with the definition of generalized hierarchies, we will provide further insights on why we distinguish between heterogeneous hierarchies of type non-covering and generalized.

GENERALIZED HIERARCHY

Conventional multidimensional models do not support the object-oriented feature of inheritance. Instead, all subclass categories of a heterogeneous class are represented as optional aggregation paths. Disadvantages of this approach are unavailability of subtypes as subdimensions of their own and, consequently, impossibility to conveniently navigate to a specific subtype or compare measure values aggregated by subtype.

A *generalized* hierarchy contains categories that can be represented by a generalization relationship. At the scheme level, the dimension graph has multiple exclusive paths converging at some categories. Categories, at which the alternative paths split and join are called *splitting* and *joining* levels, respectively. A dimension category is *generalized*, if its individual members differ considerably, e.g., have different properties and/or roll up along different paths. Unlike non-covering hierarchies, where members simply skip hierarchy levels in the same path, generalization allows to unify instances with different hierarchy schemes. As an example, let us consider staff hierarchy in purchaser dimension scheme in Figure 4.4. Category staff is specialized into subclass categories teaching staff and admin. staff, each with its own hierarchy scheme.

In the formalization, we introduce predicates \rightarrow and \rightarrow^* to express direct and transitive specialization relationship, respectively, between a pair of categories or category types: $C_i \rightarrow C_j$ states that C_i is a specialization of C_j and also that C_j is a generalization of C_i . Specialization of category type into multiple subclass category types is expressed as a set of multiple related specializations: $\{C_i, \dots, C_n\} \rightarrow C_j$. Technically, specialization relationship is a degeneration of partial roll-up (a subset of the superclass' members belongs to a subclass) to a one-to-one cardinality. Similarly, multiple related disjoint specializations correspond to a set of exclusive partial roll-up relationships.

DEFINITION IL-GENERALIZED HIERARCHY. A hierarchy is *generalized*, if it includes categories with a generalization/specialization relationship between them:
 $Generalized(H) \Leftarrow (Heterogeneous(H) \wedge \exists (C_i, C_j) \in C^H : C_i \rightarrow C_j).$

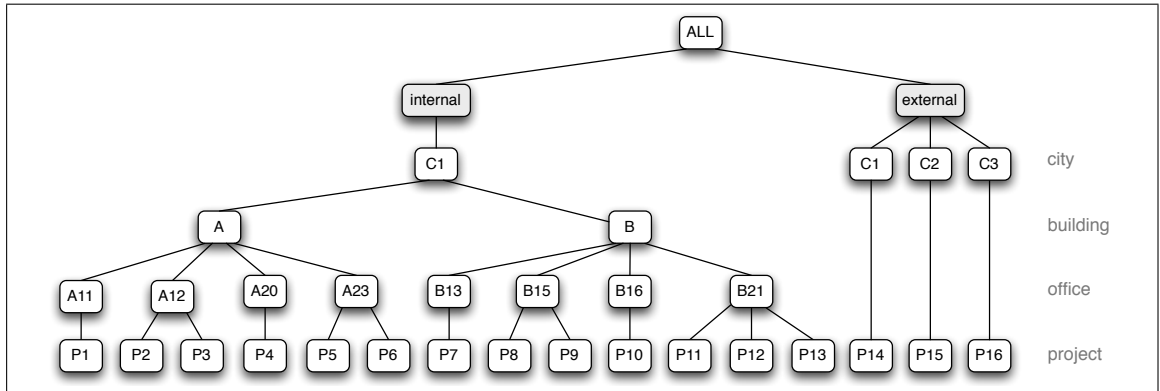


Figure 4.16: A non-covering hierarchy transformed into a generalized hierarchy

With the above definition, it becomes evident that non-covering hierarchies do not represent a subclass of generalized ones. However, the former type can be reshaped into the latter one by arranging diverse roll-up behaviors into subclasses. The sample non-covering hierarchy of project locations, depicted in Figure 4.15, can be turned into a generalized one by specializing project category into subtypes *internal* and *external*, with the resulting instance shown in Figure 4.16 (subtype nodes are filled with grey color). The resulting generalized hierarchy consists of homogeneous subtrees of its specializations. Whether non-covering hierarchies should be transformed into generalized or not depends on the expected query patterns: a homogeneous scheme of a non-covering hierarchy is useful for treating all members of a generalized category as the same class, however, the hierarchy is non-summarizable; a heterogeneous scheme of a generalized hierarchy separates different behaviors within a category into subtype hierarchies, thus treating them as different classes and yielding a summarizable mapping. Normalization of non-covering mappings is presented in Chapter 7.

X-DFM provides two alternatives for modeling specialization relationships (see Table 3.4): *i*) a set of related partial roll-up edges enables modeling of splitting levels, and *ii*) a generalization/specialization edge (or edge set) and abstract categories can be used to model both splitting and joining levels and abstract superclass category types, respectively. In Figure 4.4, a specialization edge set was used to model the sets of subtype relationships of the categories *purchaser* and *staff*.

The generalization/specialization construct in X-DFM may appear somewhat counter-intuitive as it does not reveal the direction of the roll-up relationship. Naturally, a superclass represents a higher hierarchy level with respect to its subclasses, as shown in Figure 4.17 at the example of the inheritance hierarchies in *purchaser* dimension. Two alternatives of subtyping the generalized category are presented: the left-side tree is a structural hierarchy (unit vs. staff) and the right-hand tree is a functional one (admin. purchaser vs. teaching purchaser). It is not always obvious how the inheritance relationships in a dimension should be modeled. In order to be useful, the inheritance hierarchy should reflect the desired navigation structure for the analysis, i.e., provide meaningful aggregation levels. Multiple inheritance hierarchies in the same dimension may be supported similarly to multiple alternative hierarchies.

Unlike in inheritance hierarchies, superclass relationships in roll-up hierarchies appear “upside-down”, i.e., with a generalized category below its specializations, as in *purchaser* hierarchy (Figure 4.4). This is due to the fact that generalized categories in dimensions are used to “homogenize” diverse classes into a single bottom category so that the latter can be used as a dimension in a fact scheme. Considering bi-directionality (i.e., one-to-one relationship) of generalization, we suggest that the respective X-DFM construct should be resolved into roll-up edges and propose the following approach to modeling generalized hierarchies:

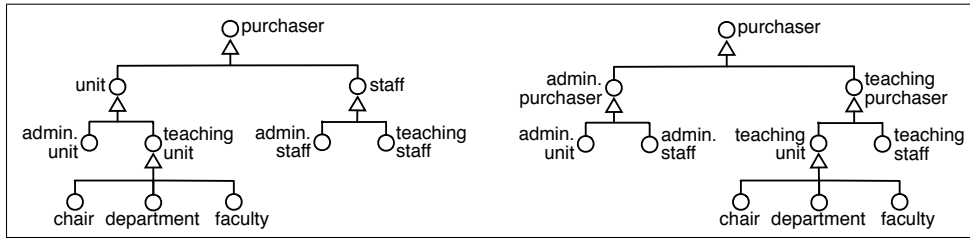


Figure 4.17: Examples of inheritance hierarchies in purchaser dimension

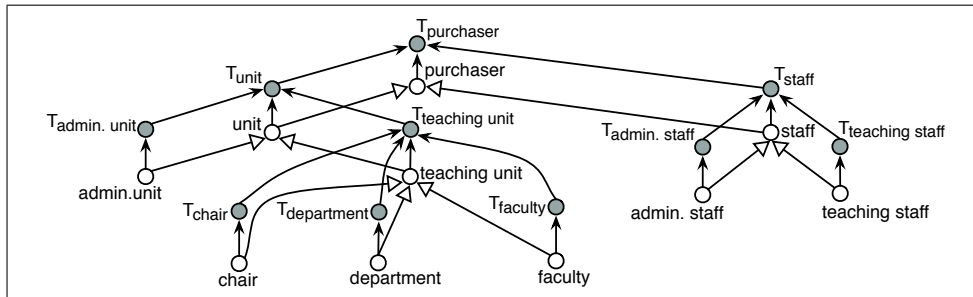


Figure 4.18: An inheritance hierarchy with local root categories added

1. A “pure” inheritance hierarchy, i.e., without roll-up relationships, is constructed via a stepwise decomposition of the generalized category into subclasses according to the sets of mutual properties. Multiple inheritance hierarchies can be specified for the same dimension, as was shown in Figure 4.17.
2. Each subclass category type \mathcal{C} is assembled into a hierarchy of its own by adding an abstract local root node $T_{\mathcal{C}}$ on top. Figure 4.18 shows the resulting graph for the left-hand side inheritance hierarchy from Figure 4.17. Shared-target style specialization relationships are replaced by a distinct specialization edge for each subclass to improve the visibility of the scheme. The most general superclass category on top of the hierarchy is augmented by the abstract root node.
- 3.
4. Proceeding top-down, each non-abstract category type \mathcal{C} is augmented with aggregation hierarchies valid for that category, placing the hierarchy scheme between \mathcal{C} and its local root $T_{\mathcal{C}}$. In the resulting scheme, subclass-specific hierarchies appear connected to the respective subclass categories, whereas common hierarchies are attached to the respective generalized category. Figure 4.19 shows the results of augmenting the dimension scheme from Figure 4.18 with aggregation paths. At the highest generalization level purchaser, the members are aggregable by location. Superclass category unit has no aggregation paths of its own, whereas admin. staff and teaching staff have their specific hierarchies as well as common ones, associated with their generalization category staff.
5. The hierarchy scheme obtained so far has multiple categories at the bottom. However, to be usable as a dimension, the scheme must originate at a single bottom category. Actually, the scheme in Figure 4.19 already contains a category qualifying as the bottom level, namely purchaser. This category has no incoming roll-up relationships and contains the union of values of all purchaser subtypes, which corresponds to the grain of purchaser dimension. The bottom category is made evident in the scheme

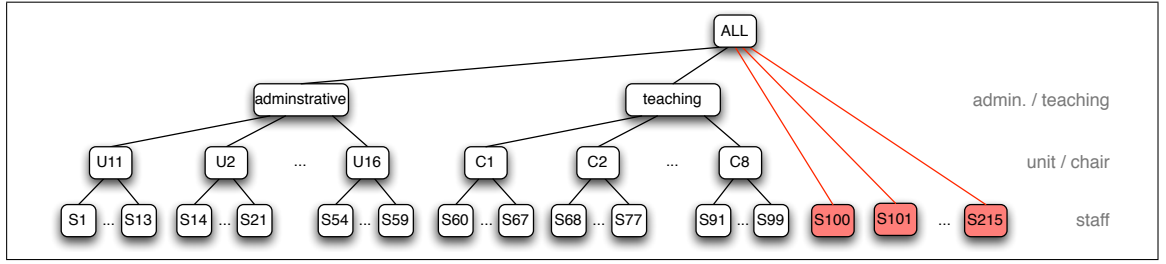
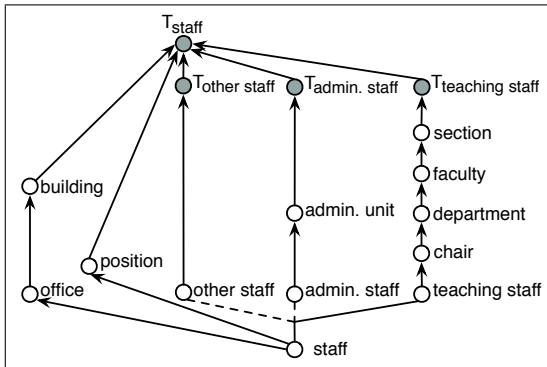


Figure 4.22: Staff hierarchy with incomplete specialization

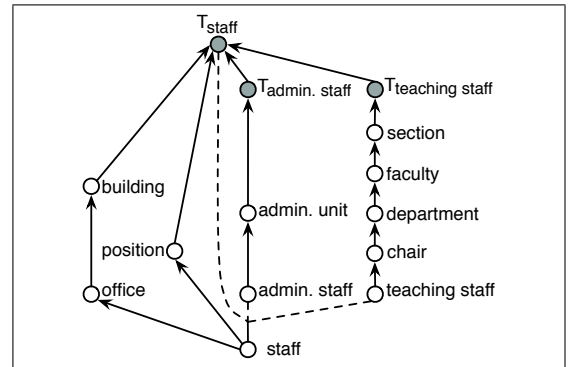
DEFINITION IL-INCOMPLETE SPECIALIZATION. A set of direct specialization relationships of a generalized category C_i is *incomplete*, if C_i contains members not belonging to any subclass:
 $IncompleteSpecialization(C_i) \Leftarrow \exists e_m \in C_i : (\forall C_j, C_j \rightarrow C_i : (\nexists e_n \in C_j : e_m \rightarrow e_n))$.

As an example of an incomplete specialization, consider staff hierarchy. In the context of fact scheme PURCHASE, depicted in Figure 4.4, subdivision into admin. staff and teaching staff is sufficient as only those staff categories may function as purchasers. In the university-wide staff hierarchy, however, there may exist further staff types. Figure 4.22 shows a sample instance of the resulting incomplete specialization hierarchy, in which the bottom level staff contains members (marked with red background color) covered by neither admin. staff nor teaching staff.

Incompleteness of a specialization can be expressed in X-DFM in a straightforward fashion by augmenting the set of partial roll-up edges of the specialization relationship with an additional edge, which shows the aggregation path for those members not involved in the specialization. A more elegant option, however, would be to normalize the specialization into a complete one by either adding the missing subclasses or using just a single additional subclass (e.g., others) to cover those members not participating in the specialization. Figure 4.23 shows two variants of modeling incomplete specialization at the example of staff dimension: the scheme in (a) preserves the original hierarchy by adding the respective partial roll-up edge, whereas in (b) the incompleteness is normalized by adding a missing specialization class.



(a) Preserving incomplete specialization



(b) Normalization into a complete specialization

Figure 4.23: Handling incomplete specialization hierarchies in X-DFM

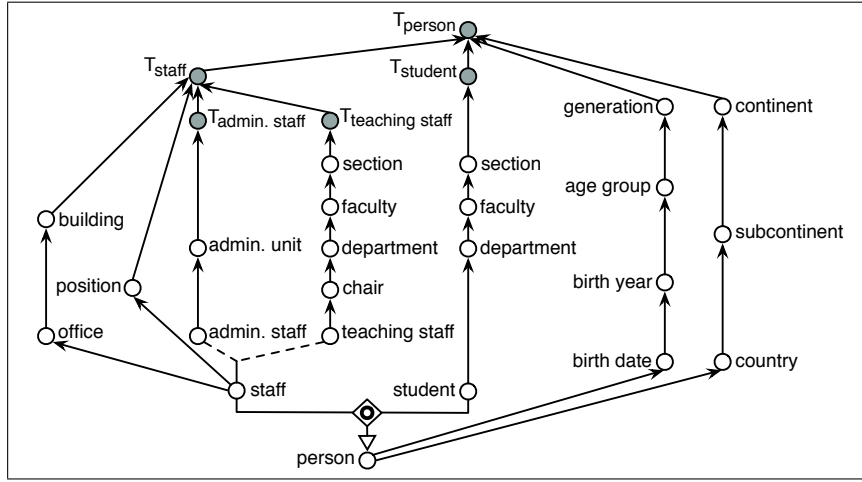


Figure 4.24: Person hierarchy with overlapping specialization

Another property of specialization is *disjointness*, which guarantees that each member of a generalized category belongs to at most one of its direct subclasses, yielding non-overlapping subclass instances.

DEFINITION IL-DISJOINT SPECIALIZATION. A set of direct specialization relationships of a generalized category C_i is *disjoint*, if each of C_i 's members belongs to at most one subclass:

$DisjointSpecialization(C_i) \Leftarrow \forall C_j, C_j \rightarrow C_i, \forall C_k, C_k \rightarrow C_i : (\nexists e_m \in C_i, \nexists e_n \in C_j, \nexists e_p \in C_k : e_n \rightarrow e_m \wedge e_p \rightarrow e_m).$

DEFINITION IL-OVERLAPPING SPECIALIZATION. A set of direct specialization relationships of a generalized category C_i is *overlapping*, if C_i contains members belonging to more than one subclass:

$OverlappingSpecialization(C_i) \Leftarrow \exists C_j, C_j \rightarrow C_i, \exists C_k, C_k \rightarrow C_i, \exists e_m \in C_i, \exists e_n \in C_j, \exists e_p \in C_k : (e_m \rightarrow e_n \wedge e_m \rightarrow e_p).$

As an example of an *overlapping*, or *non-exclusive*, specialization, let us consider modeling person dimension in the university context. Persons can be subdivided into two major categories, namely student and staff. While most persons fall into one of the two categories, there may exist cases of students employed as staff members. Figure 4.24 shows the scheme of the resulting generalized hierarchy. The most general class person at the bottom rolls up along age and origin hierarchies and specializes itself into staff and student. Due to its non-exclusive paths, overlapping specialization may not be replaced by a set of exclusive partial roll-ups. Therefore, it is shown using the overlapping specialization edge set construct in X-DFM.

To restore summarizability, an overlapping specialization must be transformed into a disjoint one. Various strategies are conceivable depending on the analysis requirements. For example, if the student status is more significant for the members belonging to both student and staff, specialization of those members into staff may simply be removed from the hierarchy instance. In many cases, however, it is desirable to keep the original relationships. A normalization technique for overlapping specialization is proposed in Section 7.2.2.

A peculiar type of generalized hierarchies is a *mixed*, or *ragged*, hierarchy, marked as non-summarizable in the hierarchy categorization shown in Figure 4.14. Mixed granularity occurs due to allowing the same

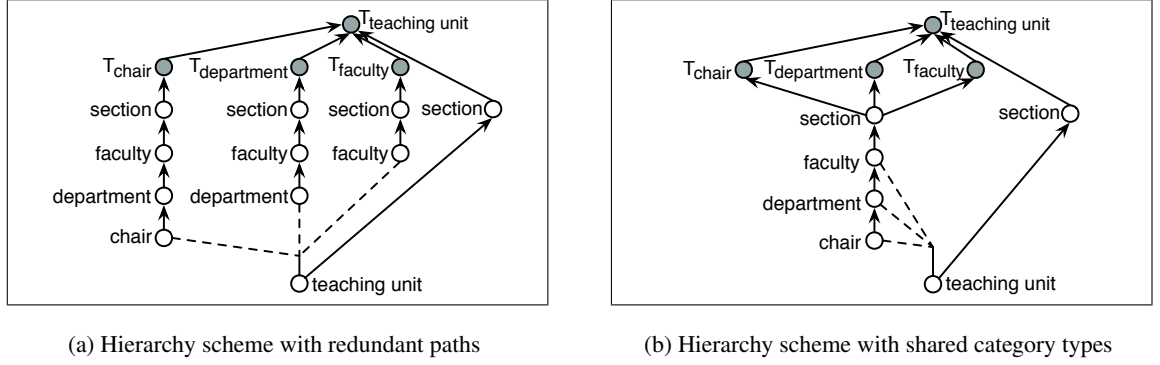


Figure 4.25: Revealing mixed grain by eliminating redundant scheme fragments

member values to represent the bottom grain, on the one hand, and serve as parents for values of other categories, on the other hand. As a result, the bottom level is a generalized category, whose direct specializations have hierarchical relationships with one another.

DEFINITION IL-MIXED-GRAIN HIERARCHY. A generalized category C_i is *mixed*, if there exists a direct or transitive roll-up relationship between C_i 's direct specializations:
 $Mixed(C_i) \Leftarrow \exists C_j, C_j \rightarrow C_i, \exists C_k, C_k \rightarrow C_i : (C_j \sqsubseteq C_k \vee C_j \sqsubseteq^* C_k).$

Mixed grain phenomenon can be observed in the teaching unit hierarchy in purchaser dimension depicted in Figure 4.21: bottom category teaching unit consists of subtypes chair, department, and faculty, whereas chair rolls-up to department and department rolls-up to faculty, i.e., the three subtypes build an aggregation hierarchy of their own. The double role (i.e., bottom and non-bottom level) of categories department and faculty in purchaser is to be understood as follows: departments and faculties may act as purchasers in their right but also consist of small units, which also act as purchasers in their own right. In the dimension scheme, the mixed grain pattern can be revealed by merging conform categories into shared category types, as shown in Figure 4.25 at the example of the generalized hierarchy teaching unit: (a) is the original scheme fragment in purchaser dimension and (b) is its equivalent in a unified multidimensional space, i.e., with each set of conform categories represented as one shared category type.

Aggregating measures along a mixed-grain hierarchy is not trivial. As an example, consider a simple query “What is the total amount spent on purchases by faculty X?”. This formulation appears ambiguous as it can be interpreted in at least three ways:

1. the total amount spent by all subdivisions of faculty X,
2. the total amount spent by faculty X itself, i.e., not including the subdivisions,
3. the total amount spent by faculty X itself and all its subdivisions.

Naturally, all three variants represent valid queries and, therefore, are expected to be supported by the system. The dimension scheme in Figure 4.21, however, does not provide adequate aggregation levels for supporting the last query, i.e., for summing up the expenditures of the faculty itself with those of the faculty's departments and chairs. In Section 7.2.2 we describe a transformation technique for re-modeling mixed grain into a summarizable generalization hierarchy by means of abstraction levels.

In Chapter 7 we propose a relational mapping of generalized schemes and demonstrate how those schemes can be represented in the navigation hierarchy of frontend tools in Chapter 8.

4.5 Classification of Multiple Hierarchies

Multiple hierarchies represent alternative ways of aggregating within a dimension, expressed as diverging non-exclusive roll-up paths of the same category. Figure 4.26 shows the part of the metamodel referring to the types of multiple hierarchies.

A classical example of multiple hierarchies is time dimension, in which date or time values may be grouped into weekdays, weeks, months, etc. Multiple hierarchies in time dimension of fact scheme PURCHASE are shown in Figure 4.30a. Multiple hierarchies are subdivided into *alternative* and *parallel* according to whether their constituent hierarchies are based on the same or on different analysis criteria, respectively. Analysis criterion refers to the underlying hierarchical property such as location, age, origin, etc.

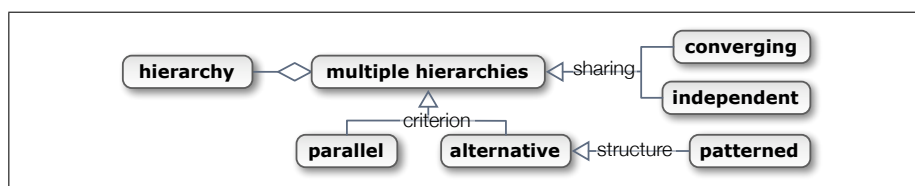


Figure 4.26: Categorization of multiple hierarchies

PARALLEL HIERARCHIES

Parallel hierarchies differ from the alternative ones by accounting for different analysis criteria. A pair of roll-up relationships originating at the same category are parallel to one another, if their target categories have no interrelation whatsoever. Figure 4.30c shows parallel hierarchies in time dimension: date values roll up to weekday, on the one hand, and to weeks or months, on the other hand. Since the values of weekday are in no way associated with those of weeks or months, both hierarchies can be explored in parallel.

Let us investigate the properties of parallel hierarchies by considering two sample hierarchy instances, shown in Figure 4.27: (a) groups date values by weekday and (b) groups the same values by weeks and, subsequently, by years. When drilling down to PURCHASE facts along the time axis, categories of both hierarchies may be combined into a new aggregation hierarchy, as shown in Figure 4.28: the hierarchy in (a) is obtained by decomposing the measure's total value by weekday, weeks, and, finally, by date, whereas the hierarchy in (b) decomposes the data first by weeks and then by weekday and date.

Apparently, parallel hierarchies in a dimension are similar to various dimensions in a fact scheme: both can be combined in any order and at any grain for aggregating the facts. The only validity constraint upon the resulting decomposition hierarchy in both cases is that the categories within each dimension should be added in the increasing order of their grain. For instance, category years could be added to decomposition hierarchy shown in Figure 4.28b only prior to its child category weeks.

DEPENDENT HIERARCHIES

Multiple hierarchies are *converging*, or *dependent*, if their paths join at some upper level. Hierarchies with no shared categories except the bottom one are called *independent*. Convergence occurs in both alternative and parallel types of multiple hierarchies, however, its semantics differs between these two types.

In case of alternative hierarchies, the convergence is *true*, i.e., both hierarchies yield the same aggregate values at any of their shared levels. As an example, consider the alternative hierarchy paths *i*) date \sqsubseteq weeks \sqsubseteq years and *ii*) date \sqsubseteq months \sqsubseteq quarters \sqsubseteq semi-annuals \sqsubseteq years in time dimension. Both paths truly converge

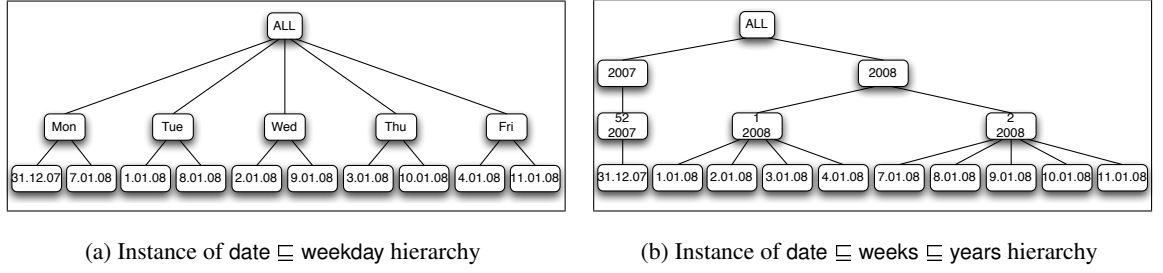


Figure 4.27: Sample instances of parallel hierarchies

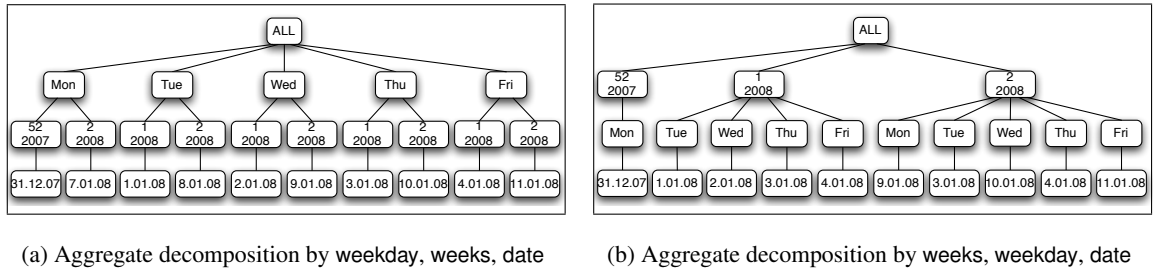


Figure 4.28: Examples of combining parallel hierarchies into new aggregation hierarchies

in years as both produce the same aggregate values when rolled up to years. In other words, the total measure value computed for any years member can be decomposed into weeks subtotals or, alternatively, into months subtotals.

In parallel hierarchies, the convergence is *fictitious* as it corresponds to a mere category type sharing in a unified space. Due to differing analysis criteria, a shared category type represents different categories, expressed as different roles assigned to each of the incoming roll-up relationships. As an example, let us suppose that project dimension in fact scheme PURCHASE has two parallel hierarchies: *i*) a supervision hierarchy project $\sqsubseteq^{(full)}$ manager $\sqsubseteq^{(full)}$ city $\sqsubseteq^{(full)}$ country, in which the last two categories refer to the city and the country of the manager's origin, and *ii*) a location hierarchy project $\sqsubseteq^{(part)}$ office $\sqsubseteq^{(full)}$ city $\sqsubseteq^{(full)}$ country, in which the same two categories refer to the project location. Figure 4.29 shows sample instances of the above hierarchies revealing the different roles of city and country categories in each hierarchy. As a result, the respective instances of those categories may contain different member sets and even the same member value may span a different subset of child elements. Therefore, parallel hierarchies produce different aggregate values at the levels of their convergence. Due to their semantic non-relatedness, shared category types may be used as different grouping criteria in a drill-down hierarchy, e.g., the total PURCHASE amount can be decomposed by city as a project location and, subsequently, by the manager's home city, or vice versa.

ALTERNATIVE HIERARCHIES

Multiple alternative hierarchies are non-exclusive roll-up paths originating from the same category and accounting for the same analysis criterion. The paths are non-exclusive in a sense that the same child entity participates in each of the alternative paths. However, when it comes to aggregating the data, it would be

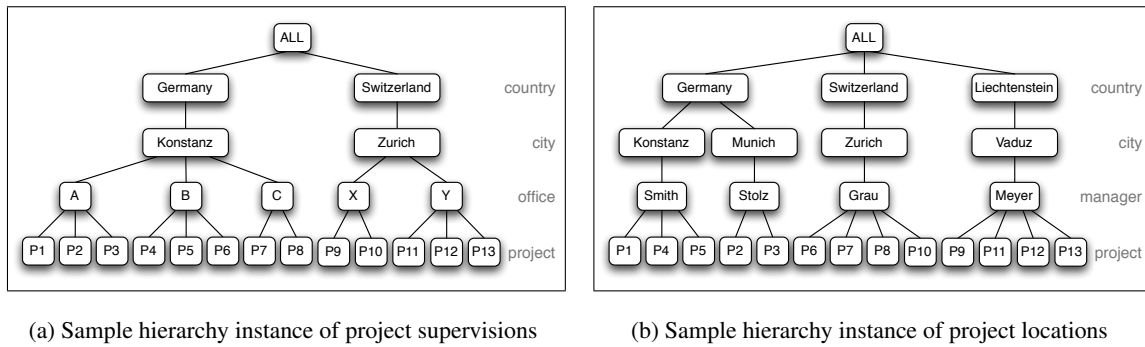


Figure 4.29: Parallel hierarchies with fictitious convergence in city and country

semantically incorrect to combine categories from the alternative paths as grouping criteria. The reason for this incompatibility is an implied many-to-many relationship between the bottom categories of the alternative paths. Back to the example of time dimension, Figure 4.30b shows two major alternative hierarchies originating as roll-up relationships of date to weeks and to months, respectively. Since there exists a many-to-many mapping between those two target categories (a month consists of multiple weeks and a week may be split between two months), the resulting paths are incompatible.

Patterned hierarchies are a special case of multiple alternatives in which a certain regularity, or pattern, reoccurs in the roll-up behavior of its members. Temporal hierarchies are a classical example of containing a structural pattern. Treating time as an ordinary dimension would yield a simplified scheme shown in Figure 3.2. However, recognition of a certain roll-up pattern, such as that each year consists of the 1st and the 2nd semi-annual, of the same 4 quarters, the same 12 months, and so on, makes it possible to generate further valid hierarchies by explicitly representing the type of the patterned category as its parent category. For instance, members of quarters roll-up to quarter (“Quarter 1 1997” and “Quarter 1 1998” are child elements of “Quarter 1”), as illustrated in Figure 4.31. The enriched dimension scheme of time supports additional aggregation paths, as can be seen in Figure 4.30a.

Jones and Song [74] proposed a comprehensive framework for identifying frequently occurring dimensional design patterns, such as temporal, action, location, object, stakeholder, qualifier, and combination.

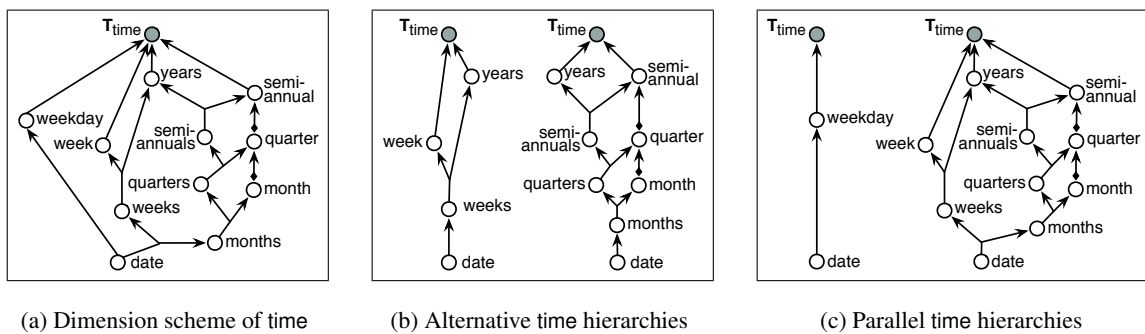


Figure 4.30: Examples of multiple hierarchies

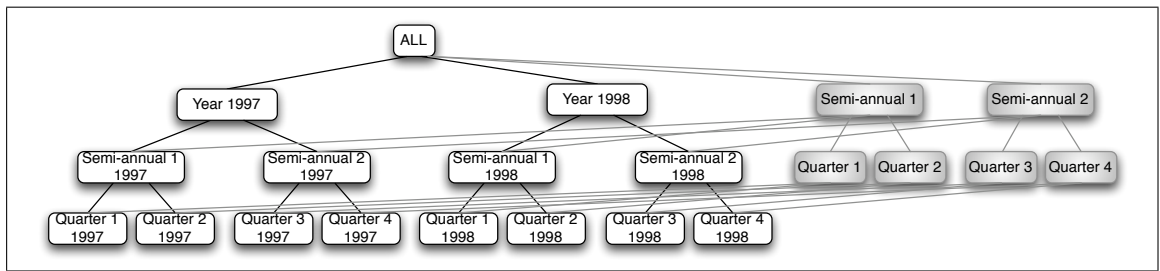


Figure 4.31: Adding semi-annual and quarter (right) as aggregation levels in time hierarchy (left)

In this chapter, we focused on extending the scope of the conceptual multidimensional model with respect to dimension and hierarchy modeling. The goal of the extensions is to cover the kinds of hierarchical behaviors not supported by conventional models. At this stage, we only provided the formalization of various dimension and hierarchy types as well as the guidelines for their conceptual design with no consideration of implementation issues. Methods for enforcing summarizability and implementing the proposed framework in relational OLAP systems are presented in Chapter 7.

Chapter 5

Measures, Facts, and Galaxies in the Multidimensional Data Model

THIS CHAPTER CARRIES ON the extension of the multidimensional data model by handling complexity in OLAP facts and relationships. Building upon the concept of the unified multidimensional space, we provide classifications of fact and measure types, multi-fact schemes, and dimension sharing patterns. Consideration of dynamic multidimensional properties, such as drilling across, deriving new measures, and rearranging cube schemes raises the phenomenon of fading duality of fact and dimension roles. A case study from the field of Surgical Workflow Analysis is used to motivate the proposed categorization and to demonstrate the benefits of the obtained extended model.

Contents

5.1	Surgical Workflow Analysis as a Motivating Case Study	90
5.1.1	Requirements of Surgical Workflow Analysis	90
5.1.2	Structuring Surgical Workflows	92
5.2	Categorization of Facts and Measures	94
5.2.1	Measurable Facts and Measure Types	95
5.2.2	Non-Measurable Facts	98
5.3	Types of Multi-Fact Schemes	99
5.3.1	Fact Degeneration	100
5.3.2	Fact Roll-up	101
5.3.3	Fact Generalization	101
5.3.4	“Fading” Duality of Fact and Dimension Roles	103
5.4	Classification of Dimension Sharing Patterns	105
5.4.1	Dimension Sharing Modes in <i>X</i> -DFM	105
5.4.2	Levels and Types of Dimension Sharing	106

5.1 Surgical Workflow Analysis as a Motivating Case Study

Healthware domain is a rather known supplier of data warehousing challenges as testified by a series of research works: Pedersen et al. [145] used patient diagnosis data as an application scenario for their extended multidimensional data model; Golfarelli et al. [47] demonstrated the methodology of obtaining multidimensional schemes from existing E/R schemes at the example of hospital admission data; Song et al. [166] use patient diagnosing and billing case study to demonstrate various strategies of handling many-to-many relationships between facts and dimensions.

Concepts and proposals presented in this work have been inspired by practical challenges encountered during the design phase of the ongoing project on developing a BI platform for the domain of *Surgical Workflow Analysis*, henceforth abbreviated as SWA. The project is hosted by the Innovation Center Computer Assisted Surgery (ICCAS)¹ and involves domestic and international collaborators from multiple scientific disciplines, such as medicine, medical engineering, databases and data warehousing, web technologies, scientific visualization, etc. Major directions of their projects are surgical workflow formalization [129], semantics [13], analysis [130], standardization [12], and visualization [128].

The medical informatics term *Surgical Workflows* refers to a methodology for intelligent acquisition and consolidation of process descriptions from surgical interventions for the purpose of their clinical and technical analysis [129]. This type of analysis is crucial for the development of surgical assist systems for the operating room of the future. Besides, it provides a framework for evaluating innovative devices and surgery strategies. Process execution data is obtained manually and semi-automatically by monitoring and recording the course of a surgical intervention. Manual acquisition is carried out either in the real-time mode, i.e., by observing the surgical intervention live in the operating room, or retrospectively, typically, from a video recording.

Surgical processes clearly fall into a category of knowledge intensive processes: even though each surgery type has a pre-defined execution scheme, individual executions are highly diverse because the actual course of each intervention is largely determined by the expertise of the surgeon expressed in terms of his/her actions, reactions, instructions to other participants, etc. Therefore, one of the long-term goals of SWA is extraction and interpretation of surgical “know-how”. In order to achieve this goal, a thorough understanding of the process context needs to be gained. This is done by capturing the relevant aspects of the domain in a formal model.

Challenges of SWA as a non-conventional data warehousing application have been tackled in a series of our own works. In [114] we showed how the requirement to warehouse the original process execution data, i.e., without pre-aggregation to a set of statically defined measures of analysis, results in the necessity to extend the foundations of the multidimensional data model. Implications of propagating the extensions introduced at the conceptual level to the backend and the presentation layer of the data warehouse system are presented in [115]. The overall process of designing and implementing a data warehouse for accumulating surgical workflow data is described in [127]. In [120] we used surgical workflow modeling as an example of handling complex data in the extended multidimensional model. Finally, a book chapter on conceptual data warehouse design for Business Process Intelligence [113] summarizes and extends previously published findings putting them into a comprehensive methodological framework.

5.1.1 Requirements of Surgical Workflow Analysis

Surgeons, medical researchers, and engineers are interested in obtaining a well-defined formal recording scheme of a surgical process that would lay a foundation for a systematic accumulation of the obtained process descriptions in a centralized data warehouse to enable its comprehensive analysis and exploration.

¹Innovation Center Computer Assisted Surgery (ICCAS) is located in the Hospital at the University of Leipzig, Leipzig, Germany. Website: <http://www.iccas.de>

Whatever abstraction approach is adopted, there is a need for an unambiguous description of concepts that characterize a surgical process in a way adequate for modeling a wide range of workflow types and different surgical disciplines.

Applications of SWA are manifold: support for the preoperative planning by retrieving similar precedent cases, clinical documentation, postoperative exploration of surgical data, formalization of the surgical know-how, analysis of the optimization potential with respect to the instruments and systems involved, evaluation of the ergonomic conditions, verification of medical hypotheses, gaining input for designing surgical assist systems and workflow automation.

Obviously, such high diversity of potential applications results in the diversity of expected query types. We distinguish the following major categories of analytical queries:

1. *Quantitative* queries are concerned with performance indicators and other measurements occurrences, frequencies, duration, or availability of various events or objects.
2. *Qualitative* queries aim at discovering relationships, patterns, trends, and other kind of additional knowledge from the data.
3. *Ergonomic* queries evaluate the design of the workspace, ergonomic limitations, positions and directions of involved participants and objects.
4. *Cognitive* queries attempt to assess such “fuzzy” issues as usefulness, relevance, satisfaction, etc.

Considering the expected kinds of queries, the multidimensional database technology seems a promising solution as it allows analysts to view the data from different perspectives, to define various metrics, and to aggregate the latter to a desired level of detail. A simple example should convey the idea of benefiting from the OLAP approach in SWA context. Figure 5.1 shows a fragment of a 3-dimensional data cube, storing instrument usage statistics (number of instruments as the cube’s measure) determined by dimensions Surgeon, Treated Structure, and Date. Besides the original cube storing the data at the finest granularity, Figure 5.1 also displays the results of two roll-up operations, which summarize the measure across all treated structures and, subsequently, across all dates, thus providing different abstractions of instrument usage numbers.

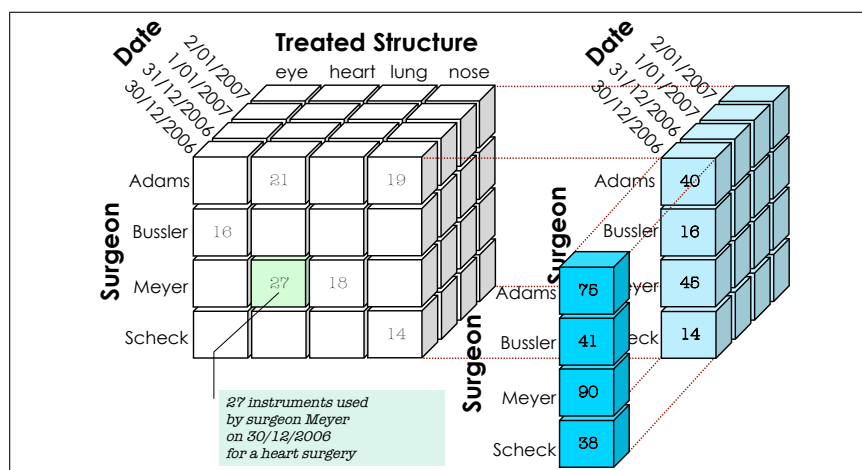


Figure 5.1: A sample 3-dimensional cube (fragment) storing surgical instrument usage statistics (left) and its aggregated views (right)

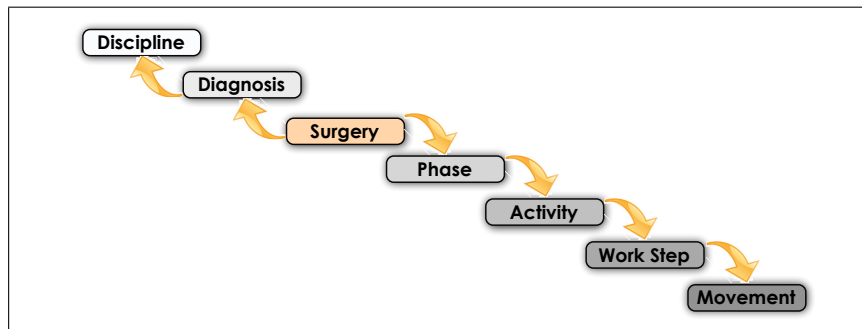


Figure 5.2: Vertical (de-)composition of a surgical process

5.1.2 Structuring Surgical Workflows

Surgical workflow is an abstraction of a surgical intervention obtained by capturing the characteristics of the original process that are relevant for the analysis. A common approach to structuring a process is to decompose it vertically, i.e., along the timeline, into logical units, such as subprocesses, stages, and work steps. Figure 5.2 shows a possible vertical decomposition scheme of a surgery: a surgical process consists of phases, which, in their turn, consist of activities, the latter being a series of work steps, each performing a certain action. Technically, an action may be executed by multiple participants and/or using multiple instruments. To account for this observation, we refine the granularity to the instrument usage level, denoted *movement*. Each movement refers to a part of the work step performed by a single actuator (i.e., a body part of a participant) on a single treated structure of a patient using a single surgical instrument. In the upward direction, surgical instances can be grouped into classes by diagnosis or therapy, which, in their turn, are associated with particular surgical disciplines. Discipline and diagnosis are the determining factors in the classification of surgery types. The above decomposition is called *logical*, or *task-based*, as it relies on the reasoning of a human expert in recognizing process elements.

An alternative decomposition practice is a *state-based* one, aimed at automated data acquisition. This approach uses the concepts *system*, *state*, and *event* to capture the state evolution of involved systems and the events that trigger state transitions. The concept of a *system* is very generic and may refer to a participant or his/her body part, a patient or a treated structure, an instrument or a device, etc. For instance, if surgeon's eyes are considered a system, then their gaze direction can be then modeled as states, while surgeon's directives to other participants may be captured as events.

The two data acquisition practices can be used as complementary ones to benefit from both the human and the systemic perspective. We introduce a superordinate concept of *component*, which is synonymous to the term *flow object* in the Business Process Modeling Notation [137], to enable uniform treatment of logical (activities and work steps) and technical (states and events) units of a process with regard to their common properties. Thereby, the analyst is able to retrieve a unified timeline for the whole course of a surgery.

In the vertical decomposition, we identify two major granularity levels of the acquired data:

- ◆ *Workflow level* refers to the characteristics of a surgical intervention as a whole, such as patient, location, date, and surgeon. This data is normally imported from clinical information systems. Workflow-level data is useful for high-level analysis, such as hospital utilisation or patient history.
- ◆ *Intra-workflow level* refers to the properties of process components (events, activities), such as instruments used or structures treated. This fine-grained data is acquired from running surgical interventions and is used for analyzing workflow execution within as well as across surgery instances.

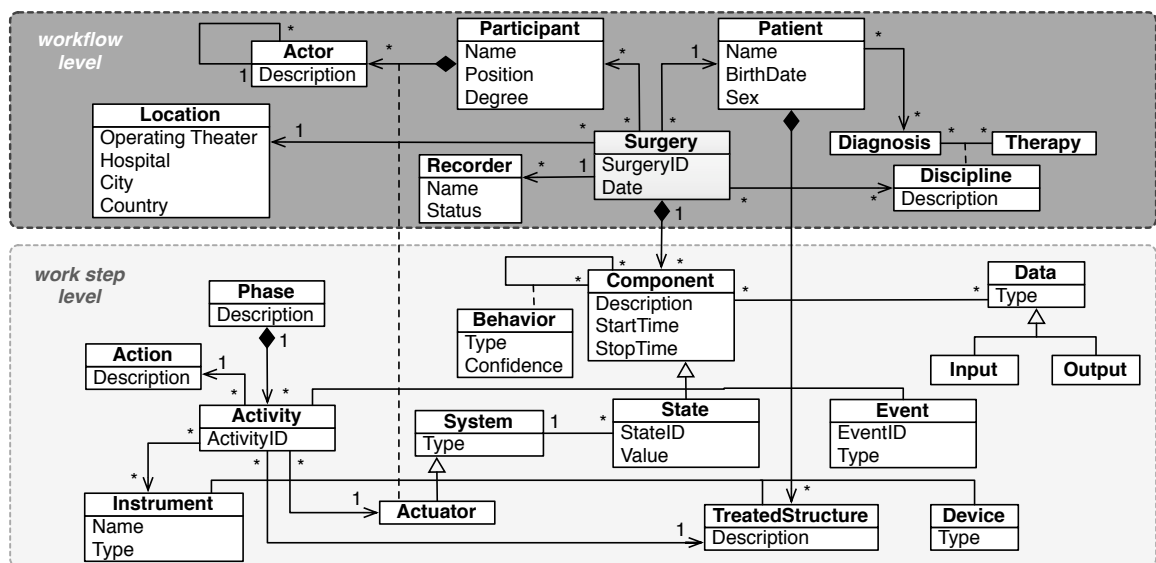


Figure 5.3: Recording scheme of a surgical process model as a UML class diagram

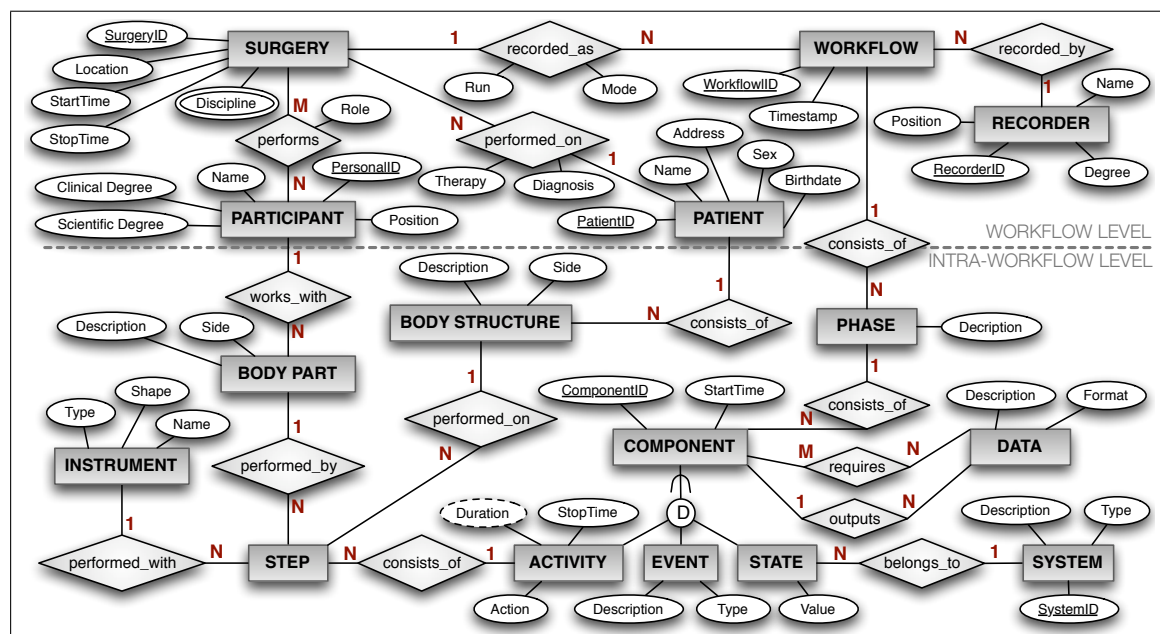


Figure 5.4: Recording scheme of a surgical process model as an E/R diagram

Figure 5.3 presents the initial approximation of the surgical workflow recording scheme, originally proposed by the collaborators from the ICCAS in [129] and refined in our joined follow-up works [114, 115]. Notice how the graphical presentation reveals the two-level structure of the recording scheme.

Figure 5.4 shows a revised model of the same application, however, expressed in the E/R (Entity/Relationship) modeling notation. This E/R diagram represents a more recent model that evolved as a result of multiple refinements. For example, this model distinguishes between a surgery itself and a surgical workflow as its abstraction, thus accounting for the possibility to produce multiple recordings of the same instance. Conceptual models shown in Figures 5.3 and 5.4 will be refined in the upcoming sections.

5.2 Categorization of Facts and Measures

In this section we undertake a categorization of fact and measure types similarly to the categorization of dimension and hierarchy types presented in the previous chapter. Multidimensional schemes obtained in the process of designing a data warehouse for accumulating surgical workflow data provide illustrative examples for concepts and constructs defined throughout this chapter. As an introductory example, consider the multidimensional fragment from the surgical workflow scenario depicted in Figure 5.5. This scheme was constructed by invoking the LL layer of *X*-DFM under the constraint of the unified multidimensional space. Therefore, sets of conform categories are shown as shared category types. For instance, start time and end time dimension schemes appear as a single scheme, apart from the top categories as those are exempted from sharing by definition.

Figure 5.5 contains a pair of interrelated fact schemes. Fact scheme SURGERY captures surgical interventions as fact entries with no measures and with a degenerated dimension SurgeryID as the fact identifier. A many-to-many relationship between SURGERY and its dimensions discipline, diagnosis, and therapy is extracted into a degenerate fact SURGERY-DISCIPLINE. Some categories have optional attributes, such as degree of diagnosis. Dimensions in SURGERY contain two examples of derived categories: age is derived from birth year and is used as an additional aggregation level in dimension patient, whereas delay is computed from the bottom categories start and end (defined as end – start) and thus represents a derived dimension of SURGERY.

The remainder of this chapter is dedicated to capturing advanced semantics related to various types of fact schemes, measure aggregability, types of multi-fact schemes and dimension sharing patterns. The presented formalization applies to the LL model as it premises the highest level of semantics.

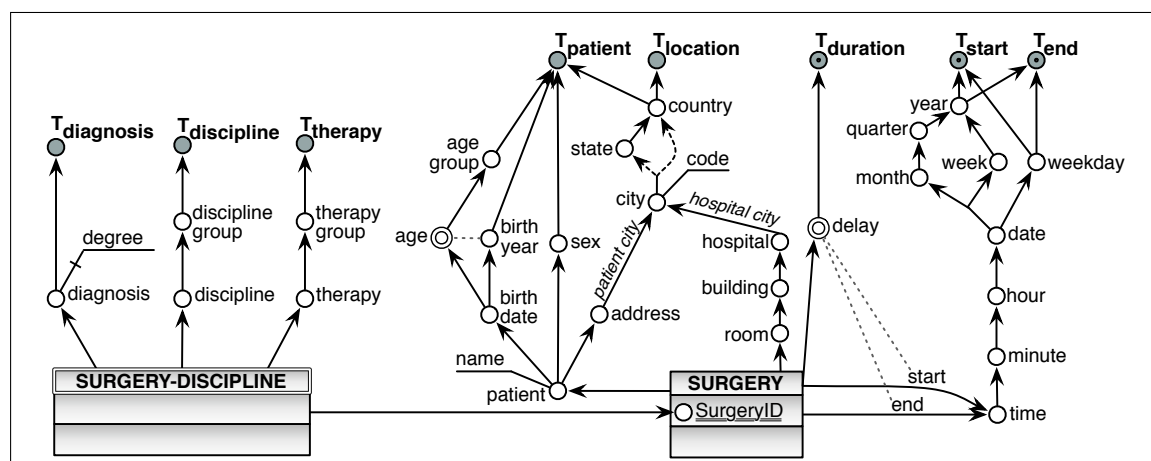


Figure 5.5: Example of multidimensional modeling of surgery data using *X*-DFM

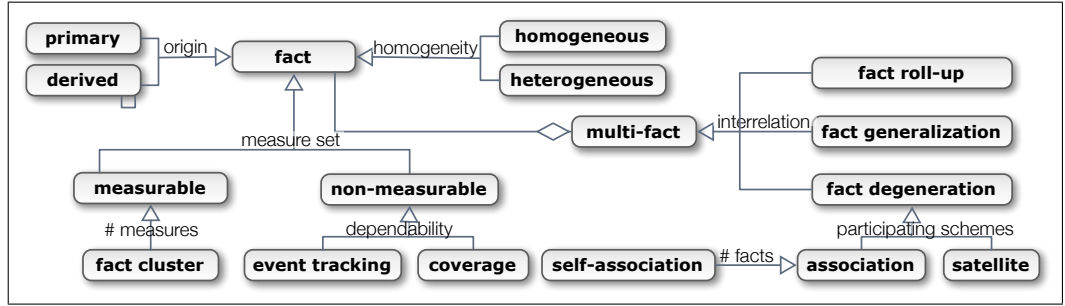


Figure 5.6: Categorization of fact and multi-fact schemes

Figure 5.6 gives an overview of fact types in the multidimensional model in form of a metamodel, with fact types in the left-hand side and multi-fact types in the right-hand side of the categorization. The metamodel uses the same notation as in that of dimension and hierarchy types (see Figure 4.7) presented in the previous chapter. With the LL definition of a fact presented in Section 3.4.2, we do not only distinguish between the terms “fact” and “measure”, but we can also classify fact schemes into *measurable* and *non-measurable* according to the size of the fact’s measure set. Measurable fact schemes and measure types are classified in Section 5.2.1, followed by the consideration of non-measurable fact schemes in Section 5.2.2. Multi-fact schemes and inter-factual relationships are inspected in Section 5.3.

5.2.1 Measurable Facts and Measure Types

Classical designation of facts is to contain relevant measures for analysis some business process. Normally, facts are modeled by specifying the measures of interest and the context (dimensions) for their analysis. Consequently, facts schemes are expected to have a non-empty set of measures.

DEFINITION LL-MEASURABLE FACT. A fact scheme \mathcal{F} is *measurable*, if it has a non-empty set of measures: $Measurable(\mathcal{F}) \Leftarrow (\mathcal{M}^{\mathcal{F}} \neq \emptyset)$.

Recall that according to the UL and the IL definitions of a fact each measure is mapped to a fact of its own, whereas a set of facts (measures) with identical dimensional characteristics form a *fact cluster*. Obviously, in the LL model, fact cluster is a just subtype of a measurable fact (see Figure 5.6), in which the measure set contains more than one measure.

DEFINITION LL-FACT CLUSTER. A fact scheme \mathcal{F} represents a *fact cluster*, if its set of measures has more that one element: $Cluster(\mathcal{F}) \Leftarrow |\mathcal{M}^{\mathcal{F}}| > 1$.

In the introductory Section 2.2.1 we introduced three fundamental types of facts according to Kimball [81], namely *i*) transactional, *ii*) periodic snapshots, and *ii*) accumulating snapshots. Apparently, this classification adopts the UL definition of a fact since the fact type is identified through its measure type. The enumerated fact characteristics actually describe the aggregation semantics of single measure attributes within that fact. According to our LL definition of a measurable fact, however, a fact scheme may contain multiple measures with different aggregation semantics. Therefore, Kimball’s classification of fact types should be re-declared as that of measure types in our framework. Figure 5.7 shows a metamodel of our proposed categorization of measure types, with meta-classes as nodes, specialization relationships between them as edges, and the underlying discrimination criteria as edge labels.

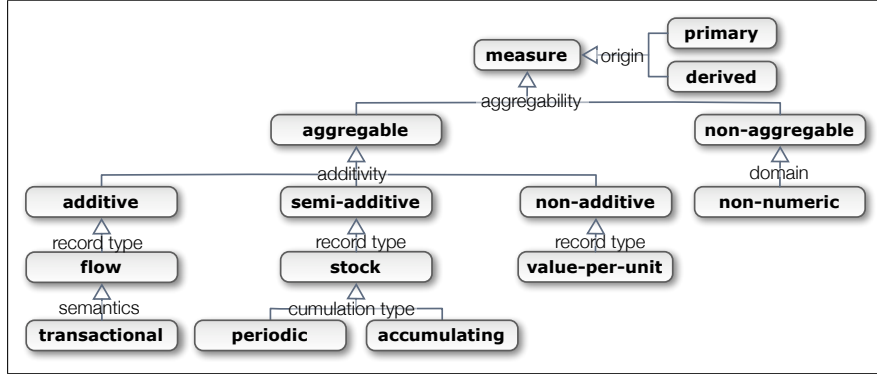


Figure 5.7: Categorization of measure types

In the first place, measures are subdivided into *primary* and *derived* according to the way their values are acquired: the values of a derived measure attribute are computed from the values of another measure or a set measures according to some derivation formula.

Another discrimination criterion, orthogonal to the measure's acquisition strategy, is *aggregability*. The aggregation semantics of a measure in a fact scheme is formalized using the concept of an *aggregation statement*, as proposed in [47]:

DEFINITION LL-AGGREGATION STATEMENT. An *aggregation statement* is a triple $\mathcal{G}(\mathcal{M}, \mathcal{D}, \Omega)$, where $\mathcal{M} \in \mathcal{M}^{\mathcal{F}}$ is a measure and $\mathcal{D} \in \mathcal{D}^{\mathcal{F}}$ is a dimension in \mathcal{F} and $\Omega \in \{\text{SUM}(), \text{AVG}(), \text{COUNT}(), \text{MIN}(), \text{MAX}(), \text{AND}(), \text{OR}(), \text{RANK}(), \text{EXISTS}(), \dots\}$ is an aggregate function.

A complete set of aggregation statements for all measures in fact scheme \mathcal{F} is denoted $\mathcal{G}^{\mathcal{F}}$. An aggregation statement $\mathcal{G}(\mathcal{M}, \mathcal{D}, \Omega) \in \mathcal{G}^{\mathcal{F}}$ declares that measure \mathcal{M} can be aggregated along dimension \mathcal{D} by applying aggregation operator Ω . The complete set of supported aggregate functions is system-dependent, however, it is sufficient to consider the standard set of distributive and algebraic functions $\text{Aggr}^{(\text{standard})} = \{\text{SUM}(), \text{AVG}(), \text{COUNT}(), \text{MIN}(), \text{MAX}()\}$ to determine the aggregability of a measure, since the latter is given by the existence of at least one aggregation statement for the respective measure.

DEFINITION LL-AGGREGABILITY. A measure \mathcal{M} is *aggregable*, if it has a non-empty set of aggregation statements: $\text{Aggregable}(\mathcal{M}) \Leftarrow (\exists \mathcal{D} \in \mathcal{D}^{\mathcal{F}}, \exists \Omega \in \text{Aggr}^{(\text{standard})} : \exists \mathcal{G}(\mathcal{M}, \mathcal{D}, \Omega) \in \mathcal{G}^{\mathcal{F}})$.
A measure \mathcal{M} is *aggregable* along a dimension \mathcal{D} , if there exists at least one aggregation statement with respect to \mathcal{M} and \mathcal{D} : $\text{Aggregable}(\mathcal{M}, \mathcal{D}) \Leftarrow (\exists \Omega \in \text{Aggr}^{(\text{standard})} : \exists \mathcal{G}(\mathcal{M}, \mathcal{D}, \Omega) \in \mathcal{G}^{\mathcal{F}})$.

DEFINITION LL-NON-AGGREGABILITY. A measure \mathcal{M} is *non-aggregable*, if its set of aggregation statements is empty: $\text{Non-aggregable}(\mathcal{M}) \Leftarrow (\forall \mathcal{D} \in \mathcal{D}^{\mathcal{F}}, \forall \Omega \in \text{Aggr}^{(\text{standard})} : \nexists \mathcal{G}(\mathcal{M}, \mathcal{D}, \Omega) \in \mathcal{G}^{\mathcal{F}})$.
A measure \mathcal{M} is *non-aggregable* along a dimension \mathcal{D} , if there exists no aggregation statement with respect to \mathcal{M} and \mathcal{D} : $\text{Non-aggregable}(\mathcal{M}, \mathcal{D}) \Leftarrow (\forall \Omega \in \text{Aggr}^{(\text{standard})} : \nexists \mathcal{G}(\mathcal{M}, \mathcal{D}, \Omega) \in \mathcal{G}^{\mathcal{F}})$.

Non-aggregable measures are rather unusual, even paradox, as they may not be analyzed using classical OLAP operators. Such measures are expected to represent some non-numeric characteristics in a fact, which are analyzed with no aggregation or using other approaches than OLAP (e.g., data mining algorithms).

Aggregable measures are further classified according to their *additivity*, defined as applicability of the $\text{SUM}()$ operator, i.e., by the ability to total the measure's values. Thereby, measures are subtyped into *fully additive*, *semi-additive*, and *non-additive*.

DEFINITION LL-FULL ADDITIVITY. A measure \mathcal{M}_j is *additive*, if it is aggregable using $\text{SUM}()$ along any dimension: $\text{Additive}(\mathcal{M}_j) \Leftarrow (\forall \mathcal{D}_i \in \mathcal{D}^{\mathcal{F}} : \exists \mathcal{G}(\mathcal{M}_j, \mathcal{D}_i, \text{SUM}()) \in \mathcal{G}^{\mathcal{F}})$.

DEFINITION LL-SEMI-ADDITIVITY. A measure \mathcal{M}_j is *semi-additive*, if it is aggregable using $\text{SUM}()$ along a subset, but not the whole set, of dimensions in the fact scheme:
 $\text{Semi-additive}(\mathcal{M}_j) \Leftarrow (\exists \mathcal{D}_i, \mathcal{D}_k \in \mathcal{D}^{\mathcal{F}} : \exists \mathcal{G}(\mathcal{M}_j, \mathcal{D}_i, \text{SUM}()) \in \mathcal{G}^{\mathcal{F}} \wedge \nexists \mathcal{G}(\mathcal{M}_j, \mathcal{D}_k, \text{SUM}()) \in \mathcal{G}^{\mathcal{F}})$.

DEFINITION LL-NON-ADDITIVITY. A measure \mathcal{M}_j is *non-additive*, if it is non-aggregable using $\text{SUM}()$: $\text{Non-additive}(\mathcal{M}_j) \Leftarrow (\forall \mathcal{D}_i \in \mathcal{D}^{\mathcal{F}} : \nexists \mathcal{G}(\mathcal{M}_j, \mathcal{D}_i, \text{SUM}()) \in \mathcal{G}^{\mathcal{F}})$.

Another additivity-based classification of measures, also referred to as “summary properties”, is known from the area of statistical databases. With respect to their summation, these properties are subtyped into *flow*, *stock*, and *value-per-unit*, elaborated in [98]. There is a strong correspondence between the above classification and Kimball's fundamental fact types:

1. A property of type *flow* records a change or a cumulative effect of a measure over a period of time. Most measures are of this type, which is fully additive. Kimball's notion of *transactional facts* falls into this category.
2. A property of type *stock* records a state or a level of a measure at specific points in time and, therefore, its values can be thought of as snapshots of the current state. Semi-additive behavior is typical for measures of type *stock* as their values are not summable with respect to temporal characteristics, but are additive along any other dimensions. Kimball subdivides snapshot measures into *periodic* and *accumulating*:
 - (a) *Periodic snapshots* represent regular (e.g., daily or monthly) measurements of status.
 - (b) *Accumulating snapshots* already include the accumulation of the measurement with respect to some starting point in time.
3. A property of type *value-per-unit* records the value of a measure at specific points in time in relation to some unit (e.g., “interest rate per repayment”). Measures of this type are non-additive as their values may be considered only in the context of their unit.

As an illustrative example of various measure types, let us consider a sample fact scheme in Figure 5.8. Both fact schemes capture hospitalization records, however, as different types of measurements. Fact scheme HOSPITALIZATION in Figure 5.8a stores each new hospitalization case as a fact record with a measure `bill_amount`. The grain of the record corresponds to the actual hospitalization transaction and `bill_amount` is an additive measure of type transactional. Fact scheme HOSPITALIZATIONS in Figure 5.8a stores the state of hospitalization records as two measurements: *i*) `records_by_day` is a periodic snapshot of running hospitalization cases registered on a daily basis, whereas *ii*) `accum_records` is an accumulating snapshot registering the total number of hospitalization cases to date. Notice that the measure `records_by_day` is still aggregable along `date` with the exception of $\text{SUM}()$ function, as its values have the same granularity and the same context and are thus comparable with one another. The measure `accum_records`, however, is non-aggregable along `date` at all as its values already include the cumulative effect over time.

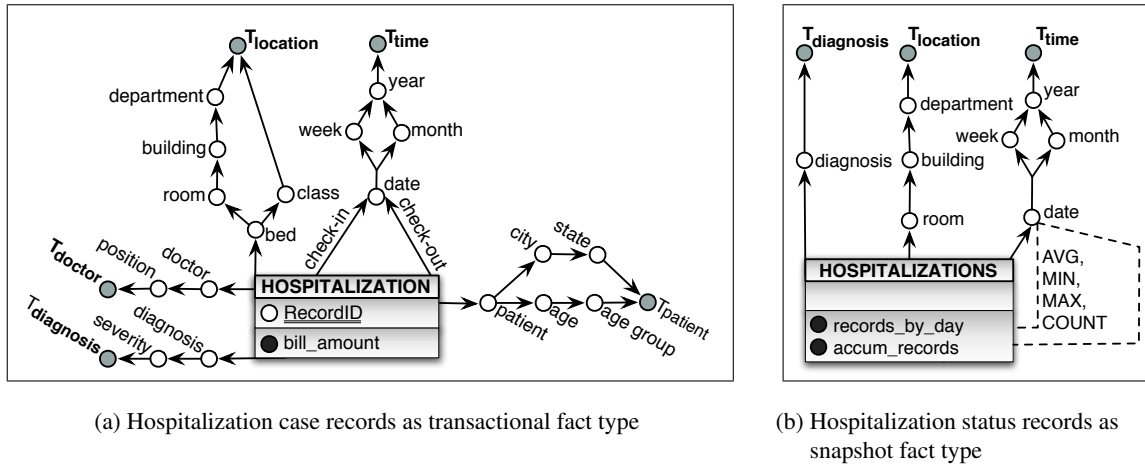


Figure 5.8: Examples of modeling hospitalization records as transactions, periodic and accumulating snapshots

5.2.2 Non-Measurable Facts

Technically, a fact type is given by a many-to-many relationship between a set of attributes. According to one of Kimball's laws, any many-to-many relationship is a fact by definition [81]. Some scenarios require storing many-to-many mappings in which no attribute qualifies as a measure. Typical cases include recording of some events, where an event is given by a combination of simultaneously occurring dimensional characteristics. Such scenarios result in so called *factless fact tables* – a term introduced by Kimball [81]. However, *fact table* is a logical design construct as it implies a table structure. We denote the conceptual equivalent of a factless fact table as a *non-measurable* fact type.

DEFINITION LL-NON-MEASURABLE FACT. A fact scheme \mathcal{F} is *non-measurable*, if its set of measures is empty: $\text{Non-measurable}(\mathcal{F}) \Leftarrow (\mathcal{M}^{\mathcal{F}} = \emptyset)$.

Non-measurable fact schemes are crucial for capturing facts of type *event-tracking* and *coverage* [81]:

1. *Event-tracking* fact record occurrence of events, defined as robust sets of many-to-many relationships between multiple dimensions. This fact type is *primary* in a sense that it is not derivable from or dependent on other facts.
2. *Coverage* facts are used to track events that were eligible, but did not happen. This fact type is *secondary* as it is always semantically related to some other fact.

Figure 5.9 provides examples of non-measurable fact schemes (for simplicity, only the bottom dimension categories are shown). Fact scheme SURGERY in Figure 5.9a is an event tracking fact type as its records correspond to real events of type surgery, characterized by a set of dimensions with no measures. An example of a useful coverage fact in this scenario could be a record of all patient diagnoses, with and without surgical treatment. A patient may have multiple diagnoses, which are prone to changes in time. In SURGERY, diagnosis dimension captures only the primary diagnosis associated with the respective surgical intervention. A complete patient diagnosis history is managed in a coverage fact DIAGNOSIS, depicted in Figure 5.9b.

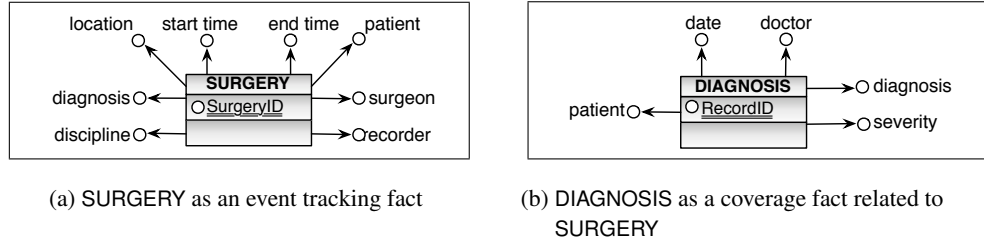


Figure 5.9: Examples of non-measurable fact schemes

FACT IDENTIFIER

Whenever the fact's grain corresponds to actual events, there may exist a dimensional attribute with identifier properties, i.e., whose values are unique for each fact entry. For example, each SURGERY instance has a unique SurgeryID. Kimball uses the concept of a *degenerated dimension* [81] to handle such *id*-like attributes, while DFM handles them as *non-dimension attributes* of a fact. In our model, a *fact identifier* attribute is a special case of a *degenerated dimension*, defined as a dimension represented by a single data field. An example of a degenerated dimension in Figure 4.4 is InvoiceNumber in fact scheme PURCHASE. Note that InvoiceNumber does not fulfill the role of a fact identifier in PURCHASE since multiple purchasing records may appear in the same invoice.

DEFINITION LL-DEGENERATED DIMENSION. A dimension \mathcal{D} is *degenerated*, if it has a single category \mathcal{C} consisting of a single attribute $\mathcal{A}^{\mathcal{C}}$: $Degenerated(\mathcal{D}) \Leftarrow (\mathcal{C}^{\mathcal{D}} = \{\mathcal{C}, \top_{\mathcal{D}}\} \wedge \mathcal{C} = \{\mathcal{A}^{\mathcal{C}}, \emptyset\})$.

DEFINITION LL-FACT IDENTIFIER. A degenerated dimension \mathcal{D}_i is a *fact identifier* of fact scheme \mathcal{F} , if the values of \mathcal{D}_i in \mathcal{F} uniquely identify the fact entries, i.e., \mathcal{D}_i functionally determines the entire set of \mathcal{F} 's dimensions: $FactIdentifier(\mathcal{D}_i, \mathcal{F}) \Leftarrow (Degenerated(\mathcal{D}_i) \wedge \mathcal{D}_i \rightarrow \mathcal{D}^{\mathcal{F}})$.

Since a degenerated dimension is represented by a single category consisting solely of the dimension level attribute, the functions $Degenerated()$ and $FactIdentifier()$ may be invoked on a dimension, a category, or an attribute. Existence of a fact identifier is common for, but not limited to, event tracking fact schemes. In case of a measurable fact scheme, the fact identifier also functionally determines its set of measures (implied by the transitive functional dependency $\mathcal{D}_i \rightarrow \mathcal{D}^{\mathcal{F}} \wedge \mathcal{D}^{\mathcal{F}} \rightarrow \mathcal{M}^{\mathcal{F}}$). As fact identifier examples, consider the attributes SurgeryID and RecordID in non-measurable fact schemes SURGERY and DIAGNOSIS, respectively, in Figure 5.9. Since a degenerated dimension is only valid in the context of its fact, X -DFM places it into the designated area inside the respective fact's node. Degenerated attributes of type fact identifier are shown by double-underlining the attribute's name. Recognition of fact identifier properties lays the foundation for modeling multi-fact schemes discussed in the following section.

5.3 Types of Multi-Fact Schemes

Multi-fact schemes emerge as sets of related fact schemes in the unified multidimensional space. In this section we investigate various patterns of semantic factual interrelationships (see Figure 5.6).

5.3.1 Fact Degeneration

There may exist a many-to-many mapping of a fact with some of its dimensional characteristics or even with another fact. Giovinozzo proposes a concept of a *degenerate fact*, defined as a measure recorded in the intersection table of a many-to-many relationship between a pair of facts or a fact and a dimension [45].

We have been able to identify the following three types of fact degeneration:

- ◆ A *satellite* fact scheme \mathcal{F}_n extracts a many-to-many relationship between a fact scheme \mathcal{F}_m and a dimension scheme \mathcal{D}_i along with the corresponding measure characteristics of this relationship as a separate fact. Thereby, \mathcal{F}_m acts as a dimension in \mathcal{F}_n . The term *satellite* reflects the accompanying nature of this degenerate fact with respect to its base fact.
- ◆ An *association* fact scheme \mathcal{F}_n extracts a many-to-many relationship between a set (typically, just a pair) of fact schemes $\{\mathcal{F}_j, j = 1, \dots, k\}$ along with the corresponding measure characteristics of this relationship as a separate fact.
- ◆ A *self-association* fact scheme \mathcal{F}_n extracts a recursive relationship within a fact scheme \mathcal{F}_m , converting the latter into two distinct dimensions of \mathcal{F}_n .

To put the above descriptions into formal definitions, we declare a function $Degeneration(\mathcal{F}_n, \{\mathcal{F}_j, j = 1, \dots, k\})$, which returns **true**, if \mathcal{F}_n is a degeneration (i.e., a satellite, an association, or a self-association) of \mathcal{F}_j . If called with just one argument, e.g., $Degeneration(\mathcal{F}_n)$, the function returns true if \mathcal{F}_n is a degeneration of any other fact scheme(s). There may be no syntactic definition of fact degeneration as it is a purely semantic property and as such, has to be explicitly specified by the data warehouse designer.

DEFINITION LL-SATELLITE FACT. A degenerate fact scheme \mathcal{F}_n is a *satellite* of fact scheme \mathcal{F}_m , if \mathcal{F}_n contains \mathcal{F}_m as a dimension scheme, with the fact identifier of \mathcal{F}_m as that dimension's bottom level in \mathcal{F}_n : $Satellite(\mathcal{F}_n, \mathcal{F}_m) \Leftarrow (Degeneration(\mathcal{F}_n, \{\mathcal{F}_m\}) \wedge \exists \mathcal{D}_i \in \mathcal{F}_n, \exists \mathcal{C}_j \in \mathcal{F}_m : \perp_{\mathcal{D}_i} = \mathcal{C}_j \wedge FactIdentifier(\mathcal{C}_j, \mathcal{F}_m))$.

Consider a many-to-many relationship between SURGERY and PARTICIPANT in the E/R diagram (Figure 5.4) of our case study. An attempt to map this relationship to a multidimensional scheme would yield a satellite fact SURGERY-PARTICIPANT, shown in Figure 5.10a. Extraction of this relationship into a separate fact scheme enables handling of that relationship's further attributes, such as *fee* stored as a measure in SURGERY-PARTICIPANT.

DEFINITION LL-ASSOCIATION FACT. A degenerate fact scheme \mathcal{F}_n is an *association* of a set of fact schemes $\{\mathcal{F}_k, k = 1, \dots, p\}$, if \mathcal{F}_n contains each \mathcal{F}_k as a dimension scheme, with the fact identifier of \mathcal{F}_k as that dimension's bottom level in \mathcal{F}_n : $Association(\mathcal{F}_n, \{\mathcal{F}_k, k = 1, \dots, p\}) \Leftarrow (Degeneration(\mathcal{F}_n, \{\mathcal{F}_k\}) \wedge \forall \mathcal{F}_k : (\exists \mathcal{D}_i \in \mathcal{F}_n, \exists \mathcal{C}_j \in \mathcal{F}_k : \perp_{\mathcal{D}_i} = \mathcal{C}_j \wedge FactIdentifier(\mathcal{C}_j, \mathcal{F}_k)))$.

DEFINITION LL-SELF-ASSOCIATION FACT. A degenerate fact scheme \mathcal{F}_n is a *self-association* of fact scheme \mathcal{F}_m , if \mathcal{F}_m plays the role of multiple dimension schemes in \mathcal{F}_n , with the fact identifier of \mathcal{F}_m as the respective dimension's bottom level in \mathcal{F}_n : $Self-Association(\mathcal{F}_n, \mathcal{F}_m) \Leftarrow Degeneration(\mathcal{F}_n, \{\mathcal{F}_m\}) \wedge \exists \mathcal{D}_i, \mathcal{D}_j \in \mathcal{F}_n, \exists \mathcal{C}_k \in \mathcal{F}_m : \perp_{\mathcal{D}_i} = \perp_{\mathcal{D}_j} = \mathcal{C}_k \wedge FactIdentifier(\mathcal{C}_k, \mathcal{F}_m)$.

As an example of an association fact, consider modeling of a trigger relationship between the facts EVENT and ACTIVITY (e.g., event X triggered activity Y). This many-to-many relationship is extracted into an association fact scheme EVENT-ACTIVITY, shown in Figure 5.10b. As expected, the schemes of EVENT

and ACTIVITY act as the corresponding dimensions of the resulting association fact, and attribute confidence could be added as a measure of the captured trigger relationship. Similarly, a self-association of EVENT can be used to capture trigger relationships between events. The resulting fact scheme EVENT-EVENT is also shown in Figure 5.10b.

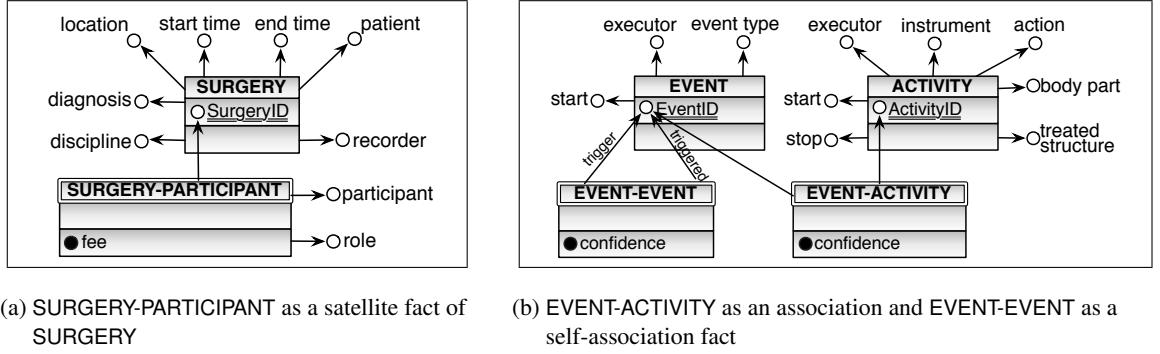


Figure 5.10: Examples of degenerate fact schemes

5.3.2 Fact Roll-up

So far we considered roll-up relationships only between facts and dimensions and between dimension categories. However, in multi-fact schemes facts may also be involved into a similar kind of roll-up relationships, i.e., be in a many-to-one relationship with each other. This behavior was already encountered in the previous section, where we observed that a degenerate fact scheme rolls-up to each of its base fact schemes by converting the latter into dimension hierarchies. In this section we investigate interfactual roll-up relationships that occur between non-degenerate fact schemes. Intuitively, a pair of fact schemes forms a roll-up, or a hierarchy, if those schemes represent different granularity of the same process, event, or object.

DEFINITION LL-FACT ROLL-UP. A pair of non-degenerate fact schemes \mathcal{F}_m and \mathcal{F}_n form a *fact hierarchy*, or a *fact roll-up*, denoted $\mathcal{F}_m \sqsubseteq^* \mathcal{F}_n$, if \mathcal{F}_m has a dimension containing the fact identifier of \mathcal{F}_n as its category at any level of the hierarchy:

$$\mathcal{F}_m \sqsubseteq^* \mathcal{F}_n \Leftarrow \neg(\text{Degeneration}(\mathcal{F}_m) \vee \text{Degeneration}(\mathcal{F}_n)) \wedge (\exists \mathcal{D}_i \in \mathcal{F}_m, \exists \mathcal{C}_k \in \mathcal{D}_i : \text{FactIdentifier}(\mathcal{C}_k, \mathcal{F}_n)).$$

A fact roll-up is *direct*, denoted $\mathcal{F}_m \sqsubseteq \mathcal{F}_n$, if the fact identifier of \mathcal{F}_n serves as the bottom category in \mathcal{F}_m , and is *transitive* otherwise, denoted $\mathcal{F}_m \sqsubseteq^* \mathcal{F}_n$.

In our scenario, hierarchical relationships exist between the event-tracking fact schemes that model the surgical process itself and its vertical decomposition into phases, activities, work steps, etc. For example, there is a transitive fact roll-up of ACTIVITY to SURGERY, as depicted in Figure 5.11a: category phase of ACTIVITY rolls-up to SurgeryID, which is a fact identifier of SURGERY.

5.3.3 Fact Generalization

An object-oriented concept of *inheritance* is helpful for dealing with heterogeneity in fact records. Conventional multidimensional models disallow heterogeneous fact instances by enforcing decomposition of a

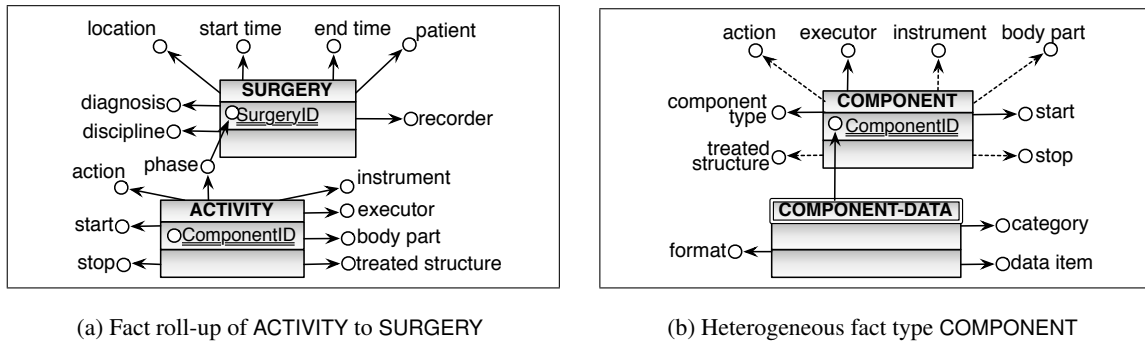


Figure 5.11: Examples of hierarchical relationships between fact schemes

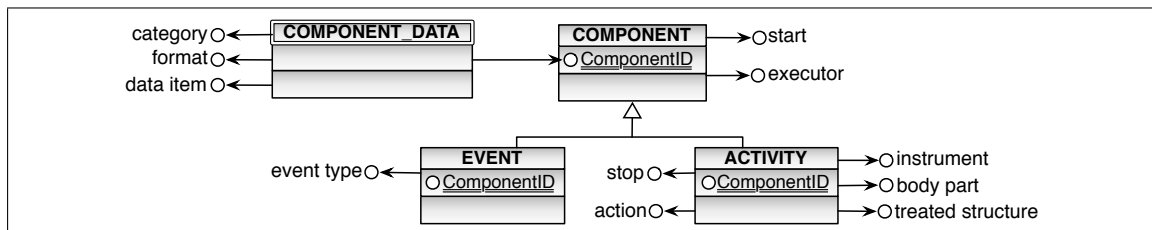
heterogeneous cube into a set of homogeneous subcubes. One obvious disadvantage of this normalization is that the entire set of fact entries can no longer be analyzed as the same class with respect to their common dimensions as there exists no such OLAP operator as cube union.

In many applications, it might be of benefit to provide support to heterogeneous schemes. Let us consider the example of modeling surgical workflows: each workflow is recorded as a sequence of different types of components, such as activities and events. All component subclasses have a subset of common properties, e.g., start time and executor, as well as type-specific properties, such as event category, instrument used, and treated structure, characterizing an activity.

A so-called *fact generalization* is obtained, when a set of heterogeneous fact types, projected to their common characteristics, is extracted into a superclass fact type. Subclass fact schemes, denoted *fact specializations*, inherit all properties of their superclass. Theoretically, there is no limitation on the depth of fact inheritance hierarchies.

In our example, **EVENT** and **ACTIVITY** are made subclasses of class **COMPONENT**, as shown in Figure 5.12. The superclass contains a set of dimensions shared by all subclasses. Moreover, fact generalization enables non-redundant modeling of fact degeneration, applicable to all subclasses, by elevating that relationship to the superclass level. In our example, **COMPONENT-DATA** is modeled as a satellite of the generalized fact scheme **COMPONENT**.

Fact generalization offers an elegant solution to handling heterogeneity. If inheritance between fact schemes is not supported, the respective multi-fact structures are resolved into a set of isolated fact schemes, which can be of type *homogeneous* or *heterogeneous*. A fact scheme is homogeneous, if it disallows partial roll-up relationships between the fact and any of its dimensions, and is heterogeneous otherwise.

Figure 5.12: Fact generalization of classes **EVENT** and **ACTIVITY** as a superclass **COMPONENT**

DEFINITION LL-HOMOGENEOUS FACT. A fact scheme \mathcal{F} is *homogeneous*, if all its fact-dimensional roll-up relationships are full: $\text{Homogeneous}(\mathcal{F}) \Leftarrow (\forall \mathcal{D} \in \mathcal{D}^{\mathcal{F}} : \mathcal{F} \sqsubseteq^{(\text{full})} \perp_{\mathcal{D}})$.

DEFINITION LL-HETEROGENEOUS FACT. A fact scheme \mathcal{F} is *heterogeneous*, if it contains partial fact-dimensional roll-up relationships: $\text{Heterogeneous}(\mathcal{F}) \Leftarrow (\exists \mathcal{D} \in \mathcal{D}^{\mathcal{F}} : \mathcal{F} \sqsubseteq^{(\text{part})} \perp_{\mathcal{D}})$.

Heterogeneous fact types result from storing non-uniformly structured facts as the same type, i.e., avoiding specialization. Figure 5.11b shows a variant of COMPONENT modeled as a heterogeneous fact scheme storing all characteristics of both subclasses EVENT and ACTIVITY. Subclass specific dimensions have to be modeled as optional roll-up relationships (dashed-line edge).

All fact types considered so far are called *primary*, or *base*, as they store fine-grained data that cannot be derived from other already available facts. It is a common practice in data warehousing to build additional multidimensional data views on top of the existing facts – a process known as “cube/subcube computation”. Results of such computations are typically materialized to boost the performance and to reveal additional information hidden in the data (e.g., to compute a new measure or derive an additional dimension category). All types of facts computed from the primary data are called *secondary*, or *derived*. The latter can be further categorized according to the way they were obtained. We limit ourselves to enumerating a few frequent fact derivation methods:

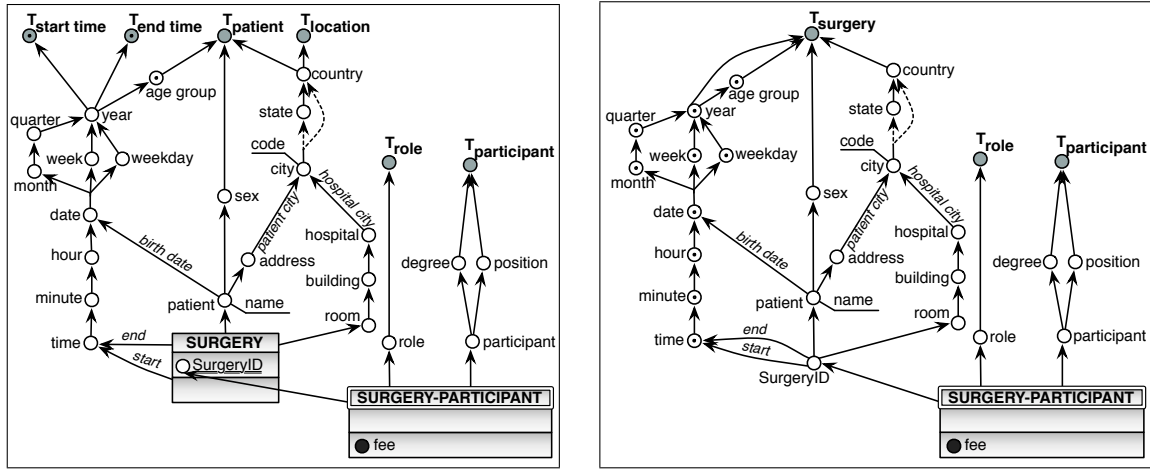
- ◆ *Aggregation* fact type is obtained by aggregating its base fact type to a coarser granularity.
- ◆ *Drill-across* fact type obtains new measures by combining measures from multiple related fact types.
- ◆ *Partition* fact type contains a subset of fact entries from its base fact type.
- ◆ *Transformation* fact type is drawn by defining a new measure from a dimension category (*push*) or converting a measure into a dimension (*pull*).

5.3.4 “Fading” Duality of Fact and Dimension Roles

Throughout this section we encountered multiple examples of fact schemes acting as dimensions in other fact schemes. That might seem paradox, but it has its legitimacy. Structurally, both facts and dimensions are given by a graph of roll-up relationships between their categories. The difference is that the aggregation graph of a dimension depends on its proper semantics, while the aggregation graph of a fact depends on the aggregation hierarchies of its analysis dimensions [3]. Fact and dimension roles are fixed only in the context of isolated fact schemes. In a multi-fact environment, however, these roles are determined by the focus of a given analytical task, which may vary from one query to another. For example, a query focusing on a measure of an association fact treats the base fact schemes of this association as the dimensions of the former. Altogether, multidimensionality implies that what is considered a fact in one task could be considered a dimension by another one, and vice versa.

The first interchangeability case is concerned with a fact scheme acting as dimension of another fact scheme. Fact scheme \mathcal{F} can be treated as a dimension in fact scheme \mathcal{F}' while querying the measure(s) of the latter scheme, when \mathcal{F}' rolls-up to the fact identifier dimension of \mathcal{F} . This relationship may be encountered in satellite facts and fact roll-ups.

One implication of this interchangeability is that it results in different conceptual schemes for the same data fragment, depending on the focus of the analysis. Figure 5.13 illustrates the example of two conceptual views of the satellite fact relationship between SURGERY-PARTICIPANT and SURGERY. A focus-independent view of both fact schemes is shown in Figure 5.13a and a perspective focused on SURGERY-PARTICIPANT and its valid aggregation paths is given in Figure 5.13b. Thereby, fact scheme SURGERY is



(a) Focus-independent view of a satellite fact scheme

(b) A base fact as a dimension of its satellite fact

Figure 5.13: Fact SURGERY as a dimension in its satellite fact SURGERY-PARTICIPANT

transformed into a dimension surgery, in which all dimensions of the original fact scheme turn into parallel hierarchies, diverging from the bottom category SurgeryID. The validity of regarding the fact identifier of SURGERY as a bottom category in surgery is given by the fact that the latter has the same grain as SURGERY fact entries, and thus, has a many-to-one, i.e., a roll-up, relationship to all other dimensions.

Another kind of interchangeability is related to treating dimensions as measures and vice versa at query time. Support of advanced OLAP operators, such as PUSH for converting a dimension category into an ad hoc measure and PULL for converting a measure into an ad hoc dimension, as well as DRILL-ACROSS for combining measures from multiple related fact schemes to obtain new facts, is a challenge not handled by conventional conceptual models. The output of those operators is a new conceptual multidimensional scheme derived from an existing scheme or a set of schemes.

Our solution for supporting scheme-transforming operators at the conceptual level is straightforward, namely, to enable explicit modeling of the resulting fact schemes. Figure 5.14 exemplifies this idea by showing the conceptual consequences of “pushing” a dimension category hospital in fact scheme SURGERY (with the original fact scheme depicted in Figure 5.15b) into a measure attribute to support query measures such as COUNT(hospital) or COUNT(DISTINCT hospital). The “pushed” category hospital itself as well as all categories below it are removed from the output dimension scheme of location as their granularity levels are no longer available. Dashed lines connecting the measure attribute hospital with all dimensions indicate non-additivity of the former, i.e., inapplicability of any aggregate function except mere enumeration with COUNT.

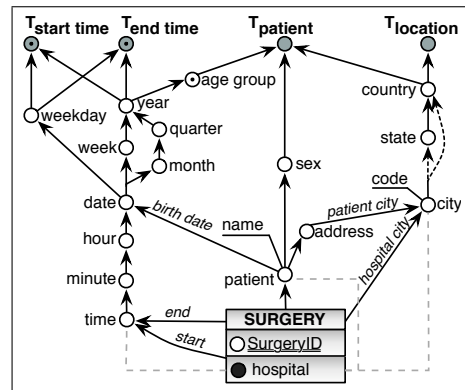


Figure 5.14: Example of a PUSH operation

5.4 Classification of Dimension Sharing Patterns

Dimension sharing is an advanced concept of dimensional modeling aimed at accurate capturing of multi-dimensional semantics. A set of cube's dimensions represents the multidimensional space of the respective fact. Intuitively, a common multidimensional space of a set of facts contains all dimensions occurring in those facts. The formal framework of the unified multidimensional space in terms of compatible and conform categories as well as of related dimension and fact schemes was provided in Section 3.4.2. In this section we proceed by investigating various patterns of dimension sharing and propose guidelines for conforming dimension schemes in *X*-DFM. We continue using the surgical workflow scenario presented in the beginning of this chapter, as it possesses the necessary complexity and variety of factual and dimensional interrelationships.

As formalized in the the IL modeling framework in Section 3.4, semantically related categories are subdivided into compatible and conform, where conformance denotes full compatibility, i.e., identity of both the value domains and the hierarchy schemes. Figure 5.15a shows fact scheme SURGERY, in which start and end categories in start time and end time dimensions, respectively, are conform, whereas date in start time and birth date in patient are compatible (the same value domain), but not conform (different roll-up paths).

5.4.1 Dimension Sharing Modes in *X*-DFM

With respect to dimension sharing, *X*-DFM can be used in three modes: *i*) non-shared, *ii*) partially shared, and *iii*) fully shared mode.

In the *non-shared mode*, categories are not examined for compatibility, i.e., each category is presented by a distinct node, as in a scheme shown in Figure 5.15a. Thereby, a category can be a shared target of multiple roll-up edges only in the case of true convergence, occurring in non-covering and multiple alternative hierarchies, with country and year as the respective examples of true convergence.

In the *partially shared mode*, only conform categories qualify to be represented as shared category types. As a result, the aggregation paths of conform categories are merged into a common path. This mode was applied in the scheme shown in Figure 5.5, where compatible yet non-conform categories birth date and date

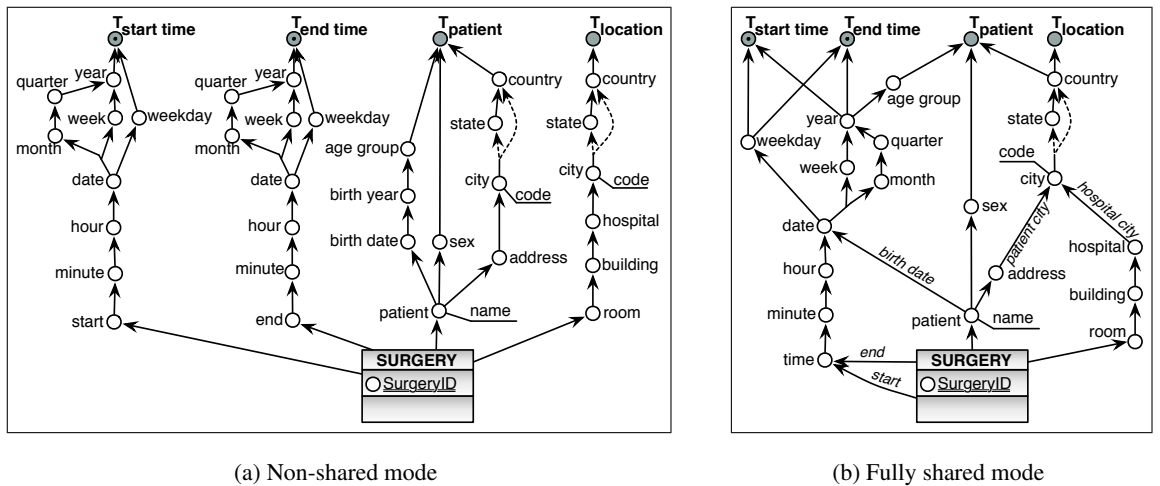


Figure 5.15: Fact scheme SURGERY modeled using different dimension sharing modes in *X*-DFM

along with their aggregation paths are exempted from merging. The partially shared mode produces “clean”, easy to interpret conceptual schemes and is a preferred option for general modeling purposes.

In the *fully shared* mode, all compatible categories are required to be represented as shared category types to reflect the semantics of the unified multidimensional space. Full sharing in *X*-DFM is achieved as follows:

- ◆ Proceeding from the bottom level upwards, each set of conform categories is merged into a single shared category type node. Subsequently, the same is done for the remaining compatible categories.
- ◆ Each shared node is named after its category type.
- ◆ The actual names of single categories behind the shared category type are shown as labels of the respective incoming roll-up edges.
- ◆ Edge labels are obligatory in the existence of multiple unrelated incoming roll-up edges of a node and may be omitted otherwise. In the latter case, the category name is identical to its category type name.
- ◆ To resolve ambiguities, fully qualified edge labels can be used (or displayed on demand). Such labels follow the naming convention *<fact-name>.<dimension-name>.<category-name>*.

Figure 5.15 depicts the concept of modeling shared dimensions at the example of fact scheme SURGERY: (a) shows the initial state of the model, in which each category is represented by a distinct node in the scheme and (b) presents the same fragment, modified by applying the above rules of category sharing in the unified multidimensional space. Dimensions start time and end time now appear merged as their hierarchy schemes are identical. The two bottom categories are merged into one node of type time, whereas the categories’ actual names start and end are shown as edge labels. Dimensions patient and location also appear related as their paths converge in the shared category type city.

In case of conform categories, the entire roll-up graphs rooted at those categories can be merged in a single step. Merging of compatible categories with deviating aggregation behavior is less trivial. Let us consider the example of merging birth date and date. Originally, birth date was modeled with the only parent category birth year of type year. Category date also rolls-up to year, however via two alternative hierarchy paths. At this stage, the designer has to decide whether these roll-up relationships should also be made valid for birth date. In that case, birth year is simply merged with year, as shown in Figure 5.15b. Category age group, however, which is a parent of year in dimension patient, does not appear semantically feasible as an additional aggregation level in dimensions start time and stop time and, therefore, is not added to their dimension schemes.

5.4.2 Levels and Types of Dimension Sharing

With respect to the degree of convergence, dimension sharing can be *full* (i.e., *conformance*) or *partial*. Partial sharing is further subtyped into *overlap* and *inclusion*, with *fact-as-dimension* and *convergence* as special cases of inclusion and overlap, respectively. Any of those sharing levels may occur between dimensions within the same fact or in different facts. Figure 5.16 shows the metamodel of dimension sharing discussed in the remainder of this section.

Conformance is the highest degree of convergence, corresponding to identical dimension schemes. In that case, multiple dimensions are represented by a single scheme, with its bottom level referenced by multiple incoming fact-dimensional roll-up relationships. Different roles of the same dimension are expressed by the labels of the respective fact-dimensional roll-up edges as well as by the name of the top category. As an example, consider the use of time hierarchy as start and stop dimensions in SURGERY as well as in ACTIVITY, with the respective scheme fragment shown in Figure 5.17. Besides, the same hierarchy serves as time dimension in WORKFLOW.

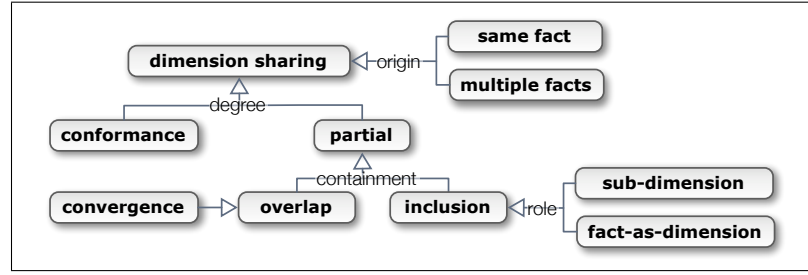


Figure 5.16: Categorization of dimension sharing patterns

DEFINITION IL-CONFORM DIMENSIONS. A pair of related dimensions D_i and D_j are *conform*, if their bottom categories are conform: $Conform(D_i, D_j) \Leftarrow (Related(D_i, D_j) \wedge (\exists C_m \in D_i, Type(C_m) = \perp_{D_i}, \exists C_n \in D_j, Type(C_n) = \perp_{D_j} : Conform(C_m, C_n)))$.

Partial sharing occurs between dimensions with non-conform bottom categories, when both dimension schemes contain hierarchy paths converging at some level. Partial sharing is of type *inclusion*, if some category in D_i fully rolls-up to the bottom level of D_j , i.e., when two dimensions represent different grain of the same hierarchy. In our scenario, this is the case with dimensions patient in SURGERY and treated structure in WORKSTEP, shown in Figure 5.18: the bottom category of patient serves as an upper aggregation level in treated structure. As a result, WORKSTEP facts, if grouped by treated structure, can be further aggregated along the entire dimension scheme of patient.

DEFINITION IL-INCLUDED DIMENSION. A dimension D_i is *included* into a dimension D_j , if scheme D_i is a subgraph in D_j : $Included(D_i, D_j) \Leftarrow (Related(D_i, D_j) \wedge \mathcal{C}^{D_i} \setminus \{T_{D_i}\} \subset \mathcal{C}^{D_j} \setminus \{T_{D_j}\})$.

Whereas the typical case of inclusion is that between dimension schemes, there exists a special case of a fact scheme acting as a dimension in another fact scheme, encountered in satellite and hierarchical factual interrelationships discussed in Section 5.3.

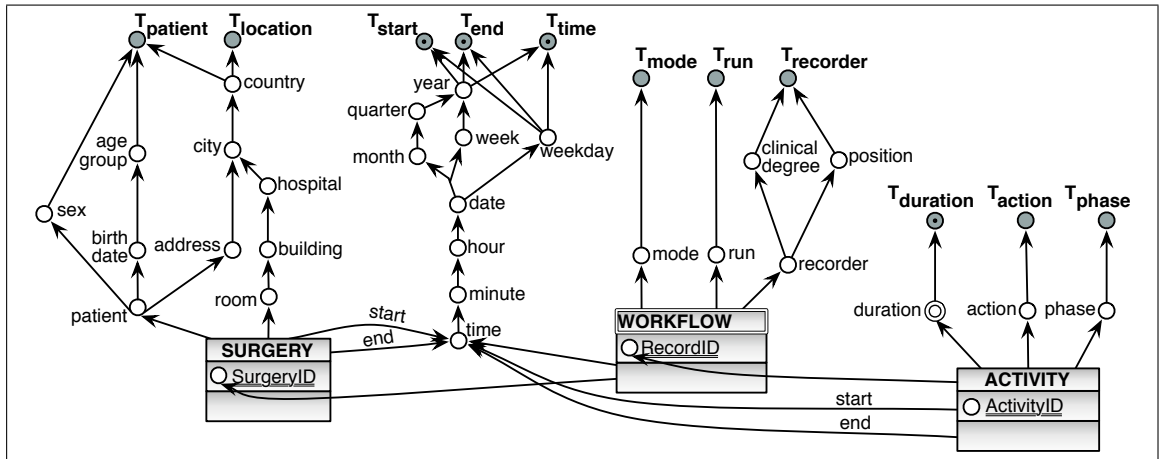


Figure 5.17: Examples of dimension conformance within and across fact schemes

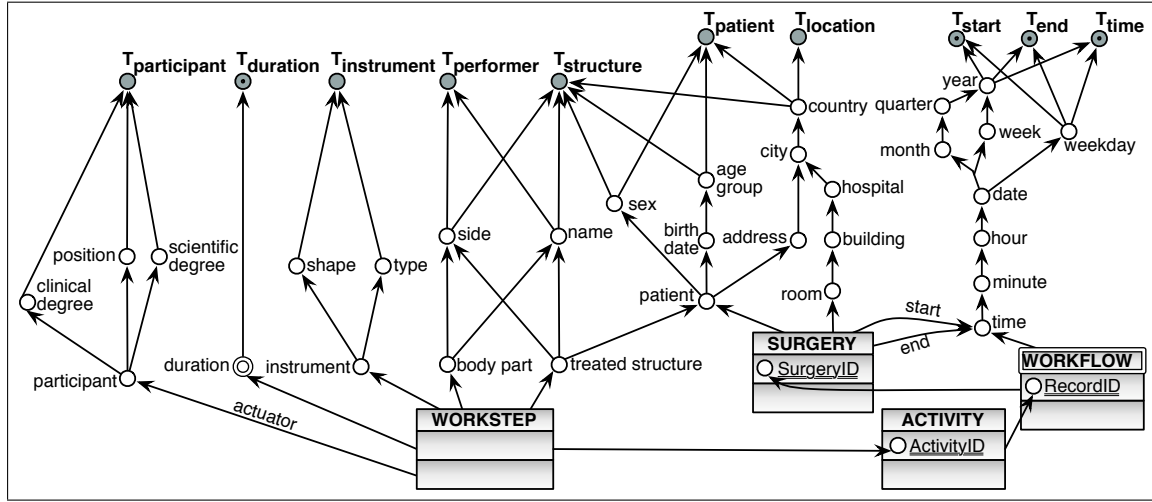


Figure 5.18: Examples of dimension inclusion within and across fact schemes

DEFINITION IL-FACT-AS-DIMENSION. A fact schema \mathcal{F} is included as a dimension in fact schema \mathcal{F}' , if \mathcal{F}' rolls-up to the fact identifier of \mathcal{F} :

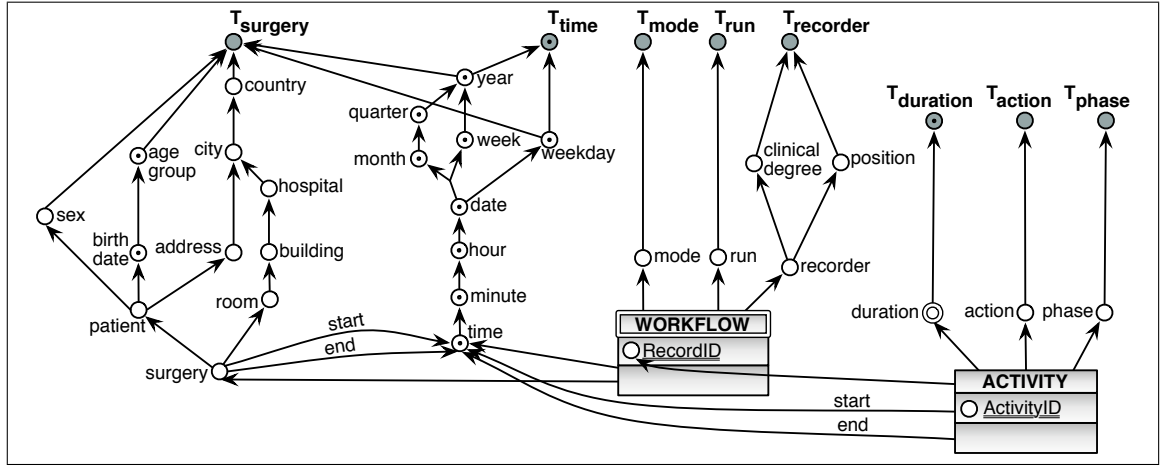
$$\text{Included}(\mathcal{F}, \mathcal{F}') \Leftarrow \text{Related}(\mathcal{F}, \mathcal{F}') \wedge \exists \mathcal{D}_i \in \mathcal{F}, \exists \mathcal{D}_j \in \mathcal{F}' : \text{FactIdentifier}(\perp_{\mathcal{D}_i}, \mathcal{F}) \wedge \perp_{\mathcal{D}_i} = \perp_{\mathcal{D}_j}.$$

To be treated as a dimension, the affected fact scheme \mathcal{F} has to be converted into a dimension scheme \mathcal{D}_j of \mathcal{F}' . This conversion of the conceptual scheme evolves in the following steps:

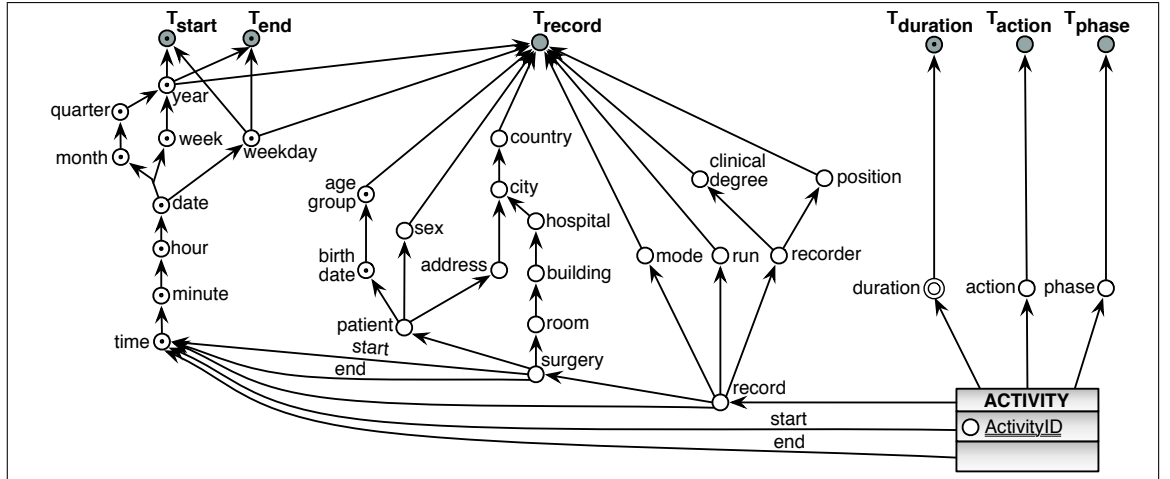
1. Fact identifier of \mathcal{F} is converted into a bottom level in \mathcal{D}_j .
2. Fact node \mathcal{F} itself is replaced by the bottom-level node created in the previous step.
3. The set of \mathcal{F} 's dimensions is added as a set of parallel hierarchies outgoing from $\perp_{\mathcal{D}_j}$.
4. Top node $\top_{\mathcal{D}_j}$ is added to the scheme.
5. Top categories of \mathcal{F} 's dimensions are removed. In case of a totally ordered top category, the total order mark should be placed into each category of the respective dimension's scheme.
6. All hierarchies in \mathcal{D}_j are made to roll-up to the new top-level $\top_{\mathcal{D}_j}$.

As an example, consider a series of fact roll-ups between ACTIVITY, RECORD and SURGERY in Figure 5.17. From within the satellite fact scheme RECORD, its base fact SURGERY plays a role of a dimension surgery and the former, in its turn, plays a role of a dimension in ACTIVITY. Figure 5.19 shows the step-wise process of creating a focus-dependent fact scheme of ACTIVITY. In case of a recursive “fact-as-dimension” inclusion, such as the one between ACTIVITY, RECORD and SURGERY, the transformation is done in a top-down fashion, i.e., starting from the fact scheme with no outgoing interfactual roll-up relationship. Therefore, SURGERY is transformed into a dimension of RECORD first, with the resulting scheme depicted in Figure 5.19a, followed by the transformation of the obtained focus-dependent fact scheme RECORD into a dimension of ACTIVITY, as shown in Figure 5.19b. As expected, in the obtained focus-dependent scheme, the number of top categories corresponds to the number of fact-dimensional roll-up relationships, i.e., the number of dimensions in the respective fact scheme.

The last pattern of dimension sharing to be considered is *overlap*. As implied by the name, dimensions are said to overlap if their schemes overlap, i.e., have hierarchies converging at some level, non-bottom for either of the dimensions.



(a) Transforming fact scheme SURGERY into a dimension surgery in WORKFLOW



(b) Transforming fact scheme WORKFLOW into a dimension record in ACTIVITY

Figure 5.19: A recursive “fact-as-dimension” inclusion in a series of fact roll-ups

DEFINITION IL-OVERLAPPING DIMENSIONS. A pair of dimensions D_i and D_j *overlap*, if they are related at a level, non-bottom for either of them:

$$Overlap(D_i, D_j) \Leftarrow Related(D_i, D_j) \wedge \neg(Included(D_i, D_j) \vee Included(D_j, D_i)).$$

Dimensions patient and location of SURGERY in Figure 5.18 overlap as they contain hierarchies that converge in city. Another example of overlap is given by dimensions performer and structure of WORKSTEP as both share non-bottom categories side and name.

Overlapping dimensions may belong to the same or to different fact schemes. The latter case builds the foundation for performing a drill-across operation over a conceptual scheme.

A special case of overlap is *convergence*, given when both dimension schemes fully converge at some level (or a set of levels). Convergence results in identical sets of roll-up destinations below the top category.

DEFINITION IL-CONVERGING DIMENSIONS. A pair of overlapping dimensions D_i and D_j *converge*, if their hierarchies roll-up to the same set of categories at some level:
 $Converge(D_i, D_j) \Leftarrow Overlap(D_i, D_j) \wedge \forall \mathcal{C}_m, \mathcal{C}_n \sqsubset \top_{\mathcal{D}_i} : (\exists \mathcal{C}_n, \mathcal{C}_n \sqsubset \top_{\mathcal{D}_j} : Conform(\mathcal{C}_m, \mathcal{C}_n))$.

None of the above examples of overlapping dimensions fulfills the convergence criterion: patient dimension has hierarchies that roll-up to sex and age group – categories not present in the scheme of location – and structure includes patient hierarchy, not present in performer.

This chapter concludes the description of the extended multidimensional data model by considering various types of facts, measures, and relationships between facts and dimensions in multi-fact environments. While presenting the conceptual framework in Chapters 3 through 5, we deliberately did not provide any guidelines for its implementation in an OLAP system. These issues are elaborated in the following two chapters.

Chapter 6

Data Warehouse Design for Non-Conventional Applications

THE CONTRIBUTION OF THIS CHAPTER IS to propose a methodological framework for designing non-conventional data warehouse applications. We focus on the challenges of operational data warehouses, in which the data flows are required to be stored without pre-aggregation to a set of measures. Surgical Workflow Analysis, introduced in the previous chapter, serves as a motivating application for the presented framework. We demonstrate that the classical data warehouse design steps are not feasible for modeling operational data warehouses and propose an alternative solution based on cardinality-driven transformation of operational models into multidimensional schemes. We expect the proposed approach to be applicable to a variety of data warehouse scenarios dealing with ad hoc analysis of operational data.

Contents

6.1	Challenges of Conceptual Data Warehouse Design	111
6.1.1	Standard Stages of Conceptual Data Warehouse Design	112
6.1.2	Limitations of Conventional Design Methodologies	112
6.2	Acquisition of Multidimensional Schemes from the E/R Schemes	113
6.2.1	Verification and Refinement of the E/R Scheme	114
6.2.2	Identifying Facts and Dimensions	118
6.3	Evaluation of the Proposed Framework	123
6.3.1	Usage Scenario 1: Discectomy Surgery	124
6.3.2	Usage Scenario 2: Functional Endoscopic Sinus Surgery	127

6.1 Challenges of Conceptual Data Warehouse Design

The original data behind multidimensional cubes typically comes from operational and other data sources of various applications. The data is made “analyzable” with the OLAP technology by remodeling it into (preferably) measurable facts with relevant dimensional characteristics. The resulting data set is made subject-oriented by getting rid of application-specific data models. Whatever data cannot be remodeled into cubes of facts and measures, remains unavailable for OLAP.

Considering that the overall characteristics of data transformation in a data warehouse are to get away from application domain orientation towards subject orientation, one could interpret the task of obtaining multidimensional models from domain-specific models as semantic reduction. For example, dealing with business processes or workflows implies that semantically rich business process models, which structure data into complex control flows of diverse elements and relationships, must be mapped to cubes of measurable facts.

6.1.1 Standard Stages of Conceptual Data Warehouse Design

Convergence of domain models into the multidimensional data model takes place primarily at the conceptual level. Therefore, conceptual design phase is the central issue of this chapter. In the context of the multidimensional data model, the output of this phase is a set of *fact schemes* and the prevailing techniques are based on graphical notations, such as the E/R model, the UML, and their variants, understandable by both designers and target users. Back to the SWA scenario, the E/R diagram in Figure 5.4 may be taken as a model of the pre-existing system, whereas the expected types of queries and applications enumerated in Section 5.1.1 correspond to the output of the requirement specification phase.

The conceptual design of a data warehouse evolves in modeling the structure of facts and their associated dimensions. Once major fact types have been defined, aggregation hierarchies are imposed upon dimensions to enable additional granularity levels. Hüsemann et al. [64] structure the conceptual data warehouse design process into the following consecutive phases:

1. Context definition of *measures*,
2. *Dimensional hierarchy* design,
3. Definition of *summarizability constraints*.

A lot of research has been done on facilitating the conceptual modeling phase as highlighted in Section 2.2.4. The main idea is to enable automatic acquisition of star schemes from the available conceptual, logical, or physical models of the underlying data sources. Since most of the existing business information management systems are relational, especially many efforts were put into deriving multidimensional models from the relational ones, the latter expressed as E/R or the UML class diagrams at the conceptual level. Outstanding contributions in this field were made by [18, 42, 47, 146, 175]. Some of the approaches, such as the ones proposed by [18] and [42], are based on “encoding” multidimensional semantics into the original E/R constructs, while others provide extended variants of the E/R model. Prominent examples of the latter class are the *starER* [175] and the *Multidimensional Entity Relationship (ME/R) Model* [159]. Yet another group of works provides mapping of the E/R schemes to ad hoc multidimensional models. The DFM approach [47], which is the predecessor of our proposed *X-DFM* model, is an example of such a methodology.

A property common to all the approaches mentioned above is *measure-centrism* as they presume that measures of interest and their desired granularity are known at the design stage. Therefore, measure specification is defined as the initial step for restructuring a relational scheme into facts and dimensions. In our experience, measure-centric data warehouse design is not universal enough to cover those application domains, in which the set of potentially useful measures cannot be determined in advance or evolves in time.

6.1.2 Limitations of Conventional Design Methodologies

Versatility of feasible application areas and analysis tasks imposes multiple challenges on the conventional data warehouse design methodology. Back to the kinds of queries in the context of SWA, the same data field may serve as a measure, i.e., input of an aggregate function, in one query or as a dimension, i.e., a grouping criterion for aggregation, in another query. As an example, let us consider entity types SURGERY

and PATIENT in Figure 5.4. In order to decide whether those entity types should be mapped to facts or dimensions one has to consider the types of queries referring to those elements. However, some scenarios, such as hospital utilization assessment, may define number of surgeries as a measure with hospital as one of its dimensions, whereas other scenarios, such as surgical discipline analysis, may be interested in the number of hospitals offering surgical support in a specified discipline. This example shows the necessity for symmetric treatment of measure and dimension roles. Similar examples can be specified for virtually any other entity type in the given case study. To support all kinds of expected queries, the detailed data, i.e., without pre-aggregation to any of the expected measures of interest, must be available in the data warehouse.

Apparently, the classical approach to designing multidimensional schemes based on the three previously mentioned phases is not adequate for supporting the above scenario. Kimball proposes a slightly different approach to structuring the conceptual design process, which appears less measure-centric and, thus, more suitable for meeting the requirements of comprehensive analysis. According to [81], the design process undergoes the stages of:

1. choosing a business process,
2. choosing the grain of the process,
3. identifying the dimensional characteristics,
4. defining the measured facts.

One major advantage of the latter approach is its ability to abstract the data model from the expected measures of analysis. This abstraction is realized by proposing to reason in terms of the business process itself and its grain and by putting measure definition into the last stage of the design. At this final step, the transformation of the “raw” process data into cubes of specified measures takes place. It is by “pushing” the measure definition from the initial step, as in [64], to a final step, as in [81], that the support of operational BI scenarios can be achieved.

Quantitative queries represent just a fraction of SWA. Some analysis tasks go beyond mere aggregation and may address more complex issues, such as pattern recognition, relevance assessment, and process discovery. These tasks require that the original process data, i.e., with no pre-aggregation to a specified set of measures, is available in the data warehouse.

In the remainder of this chapter we present an approach to data warehouse re-engineering, which automates the acquisition of a multidimensional model from the available models of input applications. The main difference between our methodology and previously proposed approaches is that we shift the focus from fact modeling to the modeling of the unified multidimensional space, in which measurable and non-measurable facts can be identified both statically, i.e., at the design phase, and dynamically, i.e., as ad hoc cubes.

6.2 Acquisition of Multidimensional Schemes from the E/R Schemes

In the preceding Chapter 5 we formalized the properties of fact and multi-fact schemes in the extended multidimensional model. The presented formalisms were illustrated using relevant multidimensional fragments from the field of SWA. However, we did not elaborate on how those fragments had actually been obtained. The algorithm for acquiring multidimensional models from non-multidimensional conceptual models of the underlying data sources is the subject of the current chapter.

In the previous section we explained why the classical data warehouse design approach, based on identifying the measures of interest and their dimensional context, is not adequate for some usage scenarios, such as process and workflow analysis. Our alternative design approach is to acquire multidimensional data views for the analysis from pre-existing conceptual models available as E/R or UML class diagrams.

The E/R model structures data in terms of *entity types* and their *attributes* as well as *relationship types* between entity types and the cardinality of each entity type's participation in a given relationship. UML class notation uses the concepts of a *class*, *property*, *relationship*, and *multiplicity* to express the same concepts as entity type, attribute, relationship type, and cardinality, respectively. Due to this straightforward correspondence between both notations, it is sufficient to provide a mapping for either of them. We opt for the E/R model as the input notation and consider the E/R diagram depicted in Figure 5.4 to be the starting point of the data warehouse design for our usage scenario. The transformation task consists in mapping semantic constructs of the E/R model to those of the multidimensional data model.

Established methodologies, which proceed by determining the facts and subsequently refining their dimensional context, proved inapplicable to our usage scenario due to their fundamental assumption of knowing the measures of interests at design time. Dealing with predominantly “factless” data of type “event tracking” necessitates a different procedure for identifying facts and measures. Our approach to identifying candidate fact entities in an E/R scheme is based on analyzing the set of each entity type's relationships with other entity types by inspecting the cardinalities and the structural constraints of those relationships. From the definitional framework of the multidimensional model provided in the previous three chapters, the following cardinality information with respect to the fact scheme structure can be deduced:

- ◆ A fact scheme is given by a set of dimension categories that have an n -ary relationship to each other or where a distinguished category, representing the grain of the fact, has a binary relationship with each other category in the set.
- ◆ In measurable schemes, each measure attribute has an $n : 1$ relationship with any of its dimensions.
- ◆ Non-measurable schemes correspond to an entity type that represents some event, along with the set of entity types, related to the former via a $1 : n$ relationship.

With respect to dimension hierarchies, the cardinality constraints are straightforward:

- ◆ Each category corresponds to an entity type and a set of its single-valued attributes.
- ◆ A homogeneous dimension hierarchy is given by a lattice of categories, in which each category is connected to at most one parent category via an $n : 1$ relationship.
- ◆ Heterogeneous hierarchy contains categories involved in a generalization relationship, with the subclass as a parent category of the subclasses.

The above observations provide valuable insights for automatic recognition of fact and dimension candidates in an E/R scheme, subject to the condition that the input scheme accurately and fully maps all required attributes, relationships, and dependencies between attributes.

6.2.1 Verification and Refinement of the E/R Scheme

In most cases, pre-existing conceptual models are tailored towards specific application needs and are thus focused on the properties and relationships relevant in that application's context. Besides, the level of detail, accuracy, and completeness of the model may not be adequate for meeting the requirements of the analysis. Therefore, the actual transformation of the E/R scheme into a multidimensional one is preceded by the transformation of the E/R scheme itself. This transformation evolves in two phases, namely, *i*) pruning and enriching the data set and *ii*) refining the relationships between the elements.

The data set is pruned to eliminate parts of the model irrelevant for the analysis. For instance, private data of the patients, such as name, address, and birth date, may have to be removed to comply with data privacy regulations. Subsequently, the model is enriched to include further data sources available. Most

of the enhancements are concerned with enabling additional granularity levels. For example, a geographic database may be added to be able to aggregate address entries by zip code, city, region, and so on.

The aim of the refinement phase is to have an accurate mapping of all relationships between entity types and attributes in the scheme. There is a fundamental difference in the way the E/R model and the multidimensional data model handle relationships: the former admits relationships only between entity types, whereas the latter specifies relationships between attributes. In the E/R model, each attribute is associated with a single entity or relationship type implying a one-to-many relationship in the general case, a one-to-one relationship in case of an identifier property, and a many-to-one or many-to-many relationship in case of a multivalued attribute. Thereby, it is impossible to specify dependencies between attributes. A legitimate way to overcome this penalty is to re-arrange attributes into additional entities and explicitly specify the relationships between the newly defined entities.

The only constructs of the multidimensional model that fully corresponds to that of an *attribute* in the E/R notation, are *dimension level* attribute, *property* attribute, and *measure* as each of them is related to one element in the scheme. Other constructs, such as facts, dimensions, and dimension categories participate in relationships and, therefore, have to be represented by entity types. As for relationship types, it is insufficient to have the cardinalities specified as a simple ratio ($1:1$, $1:n$, or $m:n$) as this notation does not reveal whether the relationship is with respect to any of participating entity types. Therefore, transformation of cardinality ratios into structural constraints (in (\min, \max) notation) is a crucial requirement of E/R scheme refinement. The above considerations of the multidimensional modeling constraints with respect to attributes and relationships is fundamental for formulating the ultimate goal of approximating an E/R scheme to a multidimensional one.

DEFINITION ER-SCHEME ACCURACY. An E/R scheme is *accurate*, if the following conditions hold:

1. the structural constraints are fully specified for each relationship type R and each entity type E participating in R ,
2. all generalization/specialization relationships are made explicit,
3. for each attribute \mathcal{A}_i in the scheme holds:
 - ◆ \mathcal{A}_i is simple (i.e., non-composite),
 - ◆ \mathcal{A}_i is single-valued,
 - ◆ \mathcal{A}_i either belongs to the key or functionally depends on the key,
 - ◆ \mathcal{A}_i is not related (i.e., has no functional dependency) to any other attribute apart from the key of its entity type.

The accurate state defined above is achieved by means of the following transformation procedure:

1. Identify implicitly composite attributes (i.e., consisting of multiple data fields) and replace them by explicit composite attributes.
2. Similarly, re-shape explicit composite attributes into entity types consisting of simple attributes.
3. Each multivalued attribute is reshaped into an entity type, related to the attribute's original entity type.
4. Identify dependencies and relations between attributes, not explicit in the scheme. Each attribute, involved in such relations, is transformed into an entity type and the relationship between newly created entity types is specified.
5. Identify implied generalization/specialization relationships and make them explicit in the scheme.
6. Redundant fragments of the scheme are merged into shared fragments.

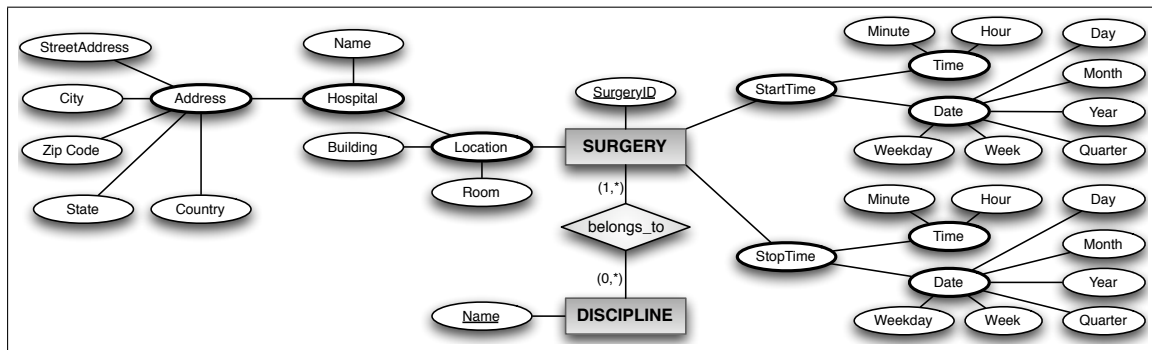


Figure 6.1: Examples of presenting complex attributes as composite ones and re-modeling multivalued attributes into related entity types

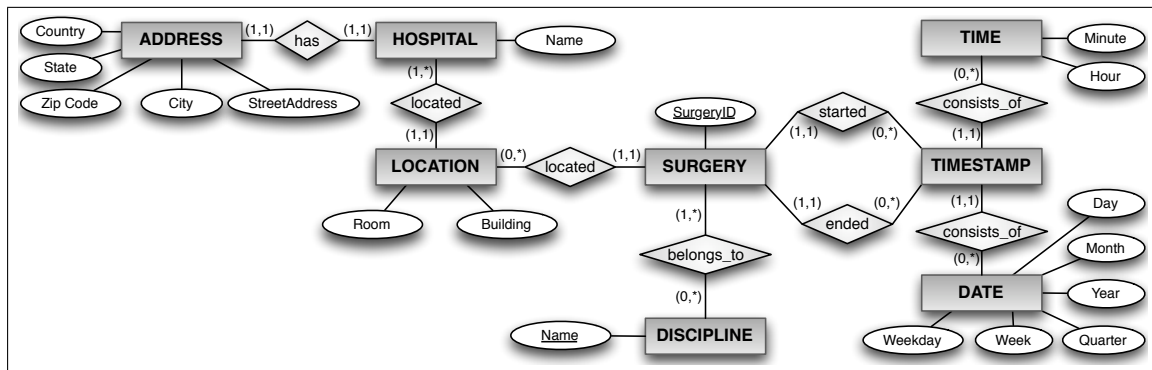


Figure 6.2: Transforming composite attributes into related entity types

7. Elements of the scheme that became obsolete are eliminated.

The above sequence of steps is chosen as to complete the transformation of the scheme in a single iteration. As an example of refining the E/R scheme according to the above procedure, let us consider the case of SURGERY attributes in Figure 5.4.

In the first step, attribute Location has been identified as implicitly composite, as its values are full addresses of respective operating theatres specified as the room, the building, the name of the hospital and its full address. The address values, in their turn, are also decomposable into multiple fields. Similarly, attributes of type date and time should be decomposed into their constituent fields. Figure 6.1 shows the results of re-structuring implicitly composite attributes Location, StartTime, and StopTime.

In the second step, composite attributes are transformed into related entity types. Figure 6.2 shows the results of translating composite attributes Location, StartTime, and StopTime into a set of entity types and aggregation relationships between them. Notice that both temporal attributes could be represented by the same entity type TIMESTAMP due to their identical structure. As a result, these two attributes are replaced by two respective relationships between SURGERY and TIMESTAMP.

Multivalued attributes are handled in the third step. Each multivalued attribute is transformed into an entity type linked to the hosting entity type of that attribute via a $1:n$ or an $m:n$ relationship. As an example,

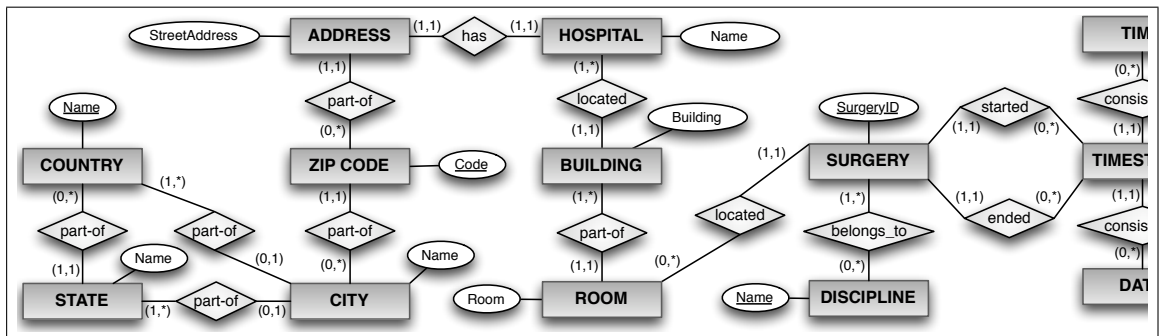


Figure 6.3: Transforming attributes into entity types to reveal implied roll-up relationships between them

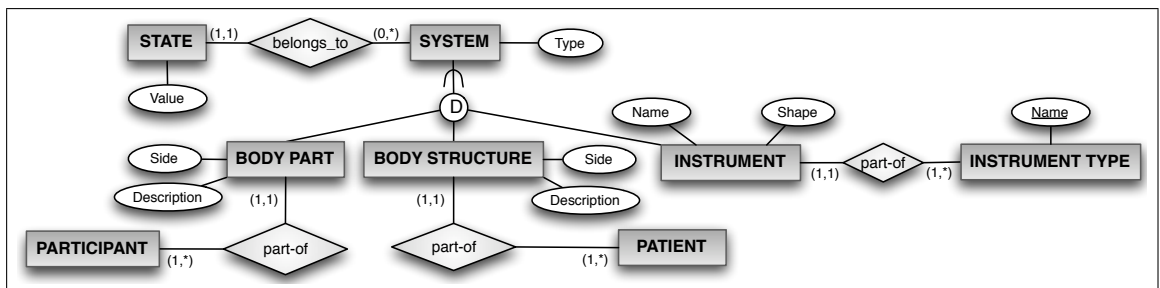


Figure 6.4: Adding specialization to the heterogeneous entity type SYSTEM

consider the result of transforming Discipline attribute into an entity type, depicted in Figure 6.1.

The fourth step of identifying “hidden” relationships between attributes is primarily concerned with revealing candidate roll-up, or “part-of”, relationships. Explicit modeling of those relationships facilitates recognition of dimension hierarchies at a later stage. Back to our example, aggregation relationships exist between Room and Building, between Building and Hospital, between Hospital and City, and so on. Figure 6.3 shows the results of revealing the hierarchical structure behind the attributes of surgery location.

In the next step, the scheme is verified with respect to implied generalization/specialization relationships. Our original model (see Figure 5.4) already contains a generalization of heterogeneous process components, such as ACTIVITY, EVENT, and STATE into a superclass COMPONENT. However, the scheme can be further refined by adding a specialization relationship to the entity type SYSTEM. In our scenario, the notion of a “system” is heterogeneous and may refer to an instrument, a body part of a participant, or a treated structure of a patient. Figure 6.4 shows the affected part of the scheme.

The last two transformation steps finalize the refined scheme by identifying redundant fragments, merging them, and removing obsolete elements. Redundant fragments emerge in the course of transforming attributes into entity types. For instance, decomposition of the Address attribute in PATIENT will yield the same scheme as the one produced by transforming the Address attribute in HOSPITAL. This redundancy is eliminated by relating all entity types that have an address property, with the same entity type ADDRESS. Some elements become obsolete at different stages of refinement. For example, entity type LOCATION (see Figure 6.2) gets dissolved into ROOM and BUILDING along with a “part-of” relationship between them (see Figure 6.3). In the final step, the scheme is verified to ensure that it contains no obsolete elements.

6.2.2 Identifying Facts and Dimensions

Once the transformation of the E/R scheme is complete, a cardinality-based transformation into a multidimensional scheme can be applied. Essentially, the task consists in determining for each entity type whether it maps to a fact, a bottom-level or an upper level dimension category.

As facts build the focus of a multidimensional scheme, the first step is concerned with identifying candidate facts in the scheme. Remember that, technically, a fact structure is a collection of properties, which have many-to-many relationship to each other and a one-to-many relationship to the fact's grain. Therefore, there exist just three structures in terms of the E/R model, which satisfy this cardinality constraint:

- ◆ An entity type that has $n : 1$ relationships with multiple other entity types,
- ◆ An n -ary relationship between a set of entity types,
- ◆ An $m : n$ relationship between a pair of entity types.

For the sake of simplicity, the first two cases can be unified as any n -ary relationship can be converted into an entity type by replacing each branch with a binary relationship towards the respective participating entity type. Besides, the concept of an entity type is generally superior to that of a relationship as the former may participate in other relationships. The third case is typical for a fact degeneration, i.e., an $m : n$ relationship between a fact and a dimension, but may also occur in a non-strict dimension hierarchy.

IDENTIFYING FACTS

Generally, a fact is given by an entity type E_f involved into multiple $n : 1$ relationships with other entity types (whereas existence of $1 : n$, $m : n$ or $1 : 1$ relationships between E_f and other entity types is not prohibited). E_f corresponds to the fact's grain, whereas the set of the related entity types along with the attributes of E_f define the fact's dimensional context. To investigate the properties of E_f as a candidate fact scheme, all relationships of E_f are arranged into the following mutually disjoint sets:

- ◆ $\mathcal{E}^{<rec>}(E_f)$ is a set of recursive (i.e., connecting the entity type to itself) relationships of E_f ,
- ◆ $\mathcal{E}^{<n:1>}(E_f)$ is a set of E_f 's candidate dimensions, i.e., a set of its non-key attributes and entity types with which E_f has an $n : 1$ relationship,
- ◆ $\mathcal{E}^{<super>}(E_f)$ is a set of superclasses, i.e., direct generalizations, of E_f ,
- ◆ $\mathcal{E}^{<sub>}(E_f)$ is a set of subclasses, i.e., direct specializations, of E_f ,
- ◆ $\mathcal{E}^{<1:1>}(E_f)$ is a set of E_f 's identifier dimensions, i.e., a set of entity types and attributes with which E_f has a $1 : 1$ relationship,
- ◆ $\mathcal{E}^{<1:n>}(E_f)$ is a set of E_f 's candidate sub-facts, i.e., entity types with which E_f has a $1 : n$ relationship,
- ◆ $\mathcal{E}^{<m:n>}(E_f)$ is a set of E_f 's candidate degenerate facts, i.e., a set of entity types with which E_f has an $m : n$ relationship.

Convergence of the E/R scheme into a multidimensional one evolves in a bottom-up fashion, starting with the entity types that qualify as terminal facts, i.e., the elements of the finest grain, and proceeding to the entities of coarser grain.

DEFINITION ER-TERMINAL FACT CANDIDATE. Entity type E_f is a *terminal fact* candidate, if it is not involved into any decomposition or specialization relationship, i.e., $\mathcal{E}^{<1:n>}(E_f) = \mathcal{E}^{<sub>}(E_f) = \emptyset$.

A $1 : n$ relationship between E_f and some other entity type E_k indicates a composition or an aggregation relationship and, thus, existence of a fact roll-up pattern (E_k rolls-up to E_f). A specialization relationship of

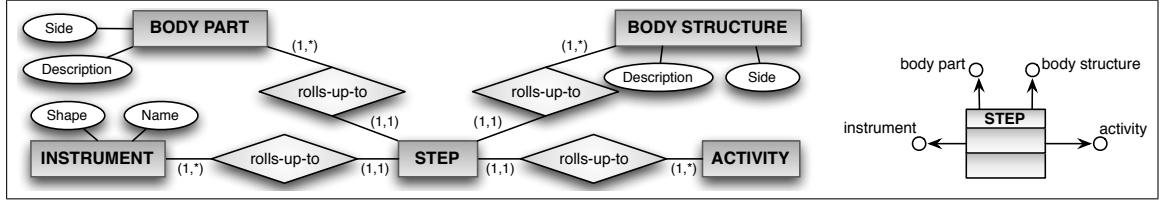


Figure 6.5: Transforming entity type STEP (left) into a fact scheme (right)

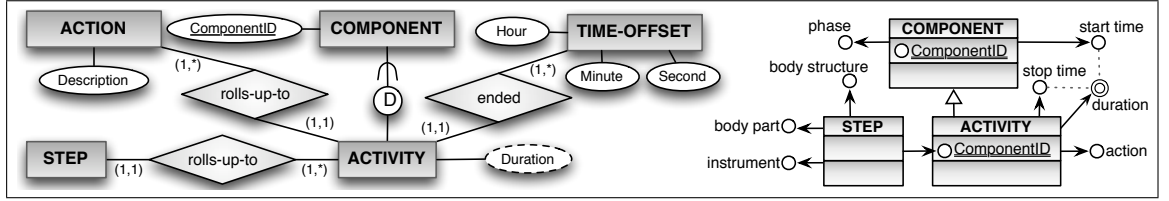


Figure 6.6: Transforming entity type ACTIVITY (left) into a fact scheme (right)

E_f implies that subclasses of E_f represent more specialized facts than E_f as they inherit all characteristics of E_f and may have further characteristics of their own.

In our surgical workflow model, three entity types qualify as terminal facts, namely, STEP, EVENT, and STATE. Figure 6.5 shows the part of the E/R diagram referring to STEP and its relationships types as well as its mapping to a 4-dimensional fact scheme. For consistency, $n:1$ relationship with full participation, i.e., with $(1,1)$ and $(1,*)$ as its structural constraints, are all renamed as “rolls-up-to”. The transformation appears straightforward as the only non-empty set of related categories $\mathcal{E}^{<n:1>}(\text{STEP}) = \{\text{INSTRUMENT}, \text{BODY PART}, \text{BODY STRUCTURE}, \text{ACTIVITY}\}$ maps seamlessly to a set of the fact’s dimensions.

As an example of a more complex fact candidate at a coarser granularity level, let us consider the entity type ACTIVITY, depicted in Figure 6.6, with its non-empty sets $\mathcal{E}^{<n:1>}(\text{ACTIVITY}) = \{\text{TIME-OFFSET}, \text{ACTION}\}$, $\mathcal{E}^{<super>}(\text{ACTIVITY}) = \{\text{COMPONENT}\}$, and $\mathcal{E}^{<1:n>}(\text{ACTIVITY}) = \{\text{STEP}\}$. As STEP has already been mapped to a fact scheme, the $1:n$ relationship is interpreted as fact roll-up. COMPONENT as a superclass of ACTIVITY is also represented as a fact, yielding a fact generalization pattern.

Finally, consider an example of identifying and modeling degenerate facts. Once an entity type E_f has been converted to a fact, its degenerate facts correspond to E_f ’s relationships in $\mathcal{E}^{<m:n>}(E_f)$ (satellite facts and fact associations) and $\mathcal{E}^{>rec>}(E_f)$ (fact self-associations). Figure 6.7 (left) shows a fragment of the E/R diagram modeling a generalized entity type COMPONENT and its relationships. COMPONENT’s $m:n$ relationship with DATA and a recursive relationship triggers are converted to a satellite fact COMPONENT-DATA and a self-association COMPONENT-TRIGGER, respectively, as depicted in Figure 6.7 (right).

Having considered various examples of identifying parts of the E/R scheme that qualify to be converted into facts, we are ready to provide an algorithmic description of acquiring fact schemes from accurate E/R schemes. Algorithm 1 is invoked on each “terminal” entity type E_f , outputting a set of fact schemes, obtained by recursively applying itself to each entity type identified as a candidate fact. Sets $\mathcal{E}^{(<sub>)>}(E_f)$ and $\mathcal{E}^{(<1:n>)}(E_f)$ used for identifying “terminal” entity types become obsolete inside the algorithm as it proceeds in the bottom-up fashion.

In the first step, Algorithm 1 creates an empty fact type and converts the attributes of the underlying entity into measures and degenerated dimensions, as shown in the subroutine Algorithm 2.

Algorithm 1: ConvertToFact**Data:** Entity type E_f , Set of previously identified fact schemes \mathcal{F} **Result:** Updated set of fact schemes \mathcal{F} **begin** $\mathcal{F} \leftarrow \text{ConvertAttributes}(E_f, \mathcal{F});$ $\mathcal{E}^{<rec>} \leftarrow \emptyset;$ $\mathcal{E}^{<super>} \leftarrow \emptyset;$ $\mathcal{E}^{<1:1>} \leftarrow \emptyset;$ $\mathcal{E}^{<n:1>} \leftarrow \emptyset;$ $\mathcal{E}^{<m:n>} \leftarrow \emptyset;$ $Rel \leftarrow \text{getRelationships}(E_f);$ **foreach** $E_f \diamond E_i \in Rel$ **do** **if** $E_f = E_i$ **then** $\text{append}(E_f \diamond E_i, \mathcal{E}^{<rec>});$ **else if** $E_i = \text{Generalization}(E_f)$ **then** $\text{append}(E_i, \mathcal{E}^{<super>});$ **else** $c = \text{Cardinality}(E_f \diamond E_i);$ **switch** c **do** **case** $1 : 1$ $\text{append}(E_i, \mathcal{E}^{<1:1>});$ **case** $n : 1$ $\text{append}(E_i, \mathcal{E}^{<n:1>});$ **otherwise** $\text{append}(E_i, \mathcal{E}^{<m:n>});$ **foreach** $E_i \in \mathcal{E}^{<1:1>}$ **do** $\text{addDimension}(E_i, \mathcal{F}, \text{"shadow"});$ **foreach** $E_i \in \mathcal{E}^{<n:1>}$ **do** $\text{addDimension}(E_i, \mathcal{F}, \text{"normal"});$ **if** $\text{qualifiesAsFact}(E_i)$ **then** $\mathcal{F} \leftarrow \text{ConvertToFact}(E_i, \mathcal{F});$ **foreach** $E_i \in \mathcal{E}^{<super>}$ **do** $\text{addDimension}(E_i, \mathcal{F}, \text{"superclass"});$ $\mathcal{F} \leftarrow \text{ConvertToFact}(E_i, \mathcal{F});$ **foreach** $E_f \diamond E_i \in \mathcal{E}^{<rec>}$ **do** $\mathcal{F}_k \leftarrow \text{CreateFactSelfAssociation}(\mathcal{F}, E_f \diamond E_i);$ $\text{append}(\mathcal{F}_k, \mathcal{F});$ **foreach** $E_i \in \mathcal{E}^{<m:n>}$ **do** $\mathcal{F}_k \leftarrow \text{CreateDegenerateFact}(\mathcal{F}, E_i);$ $\text{append}(\mathcal{F}_k, \mathcal{F});$ **foreach** $E_i \in \mathcal{E}^{<1:n>}$ **do** $\text{addDimension}(E_i, \mathcal{F}, \text{"normal"});$ $\text{append}(\mathcal{F}, \mathcal{F});$ **return** $\mathcal{F};$ **end**

Algorithm 2: ConvertAttributes

```

Data: Entity type  $E_f$ 
Result: Fact type  $\mathcal{F}$  corresponding to  $E_f$ 
begin
   $\mathcal{F} \leftarrow \text{createFact}(E_f)$ ;
   $\text{Attr} = \text{getAttributes}(E_f)$ ;
  foreach  $A \in \text{Attr}$  do
    if  $\text{isMeasure}(A)$  then
       $\text{addMeasure}(A, \mathcal{F})$ ;
    else if  $\text{isIdentifier}(A)$  then
       $\text{addDimension}(A, \mathcal{F}, \text{"identifier"})$ ;
    else
       $\text{addDimension}(A, \mathcal{F}, \text{"degenerated"})$ ;
  return  $\mathcal{F}$ ;
end

```

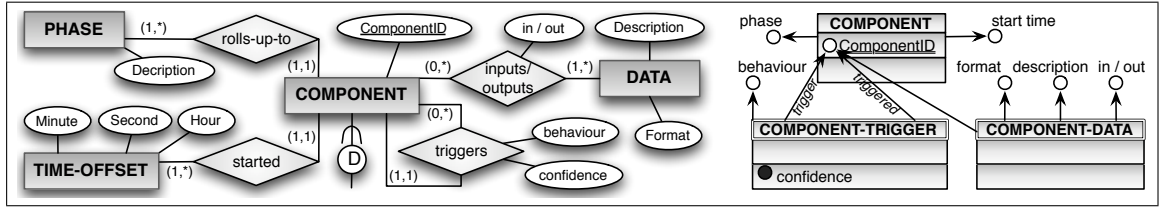


Figure 6.7: Transforming $m:n$ and recursive relationships of **COMPONENT** (left) into degenerate fact schemes (right)

IDENTIFYING DIMENSION HIERARCHIES

Fact schemes produced by Algorithm 1 are incomplete in a sense that dimensions are defined solely in terms of their bottom categories. Therefore, the next step is to model dimension hierarchies. Once the E/R scheme is brought into an accurate state defined in the previous subsection, dimension hierarchies become easily identifiable: each category type corresponds to an entity type and the partial order on the category types is given by the hierarchical, i.e., many-to-one, relationships between categories.

Similarly to the fact conversion procedure, dimension schemes are constructed in a bottom-up fashion by rooting the dimension's graph at the bottom category and recursively adding roll-up relationships until the top level is reached. In the presence of multiple and heterogeneous hierarchies the resulting dimension scheme contains diverging and converging paths.

Roll-up behaviour of an entity type is determined by its relationships. As dimension categories are identified bottom-up, the set of *relevant* relationships is reduced to $1:1$, $n:1$, and $m:n$. Let us consider the process of hierarchy modeling at the example of **phase** dimension in **COMPONENT**. The corresponding part of the E/R diagram (simplified for presentation purposes) is given in Figure 6.8.

Possible roll-up behaviours of a candidate dimension category given by an entity type E_d can be categorized based on the number of its relevant relationships, their structural constraints and interdependencies (detailed definitions of the presented hierarchy types are given in Chapter 4):

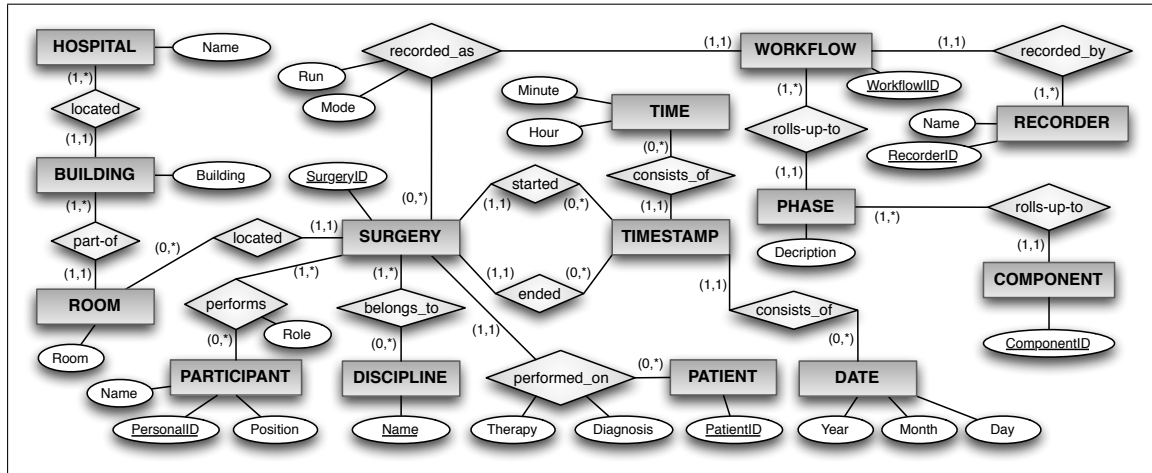


Figure 6.8: Fragment of the E/R scheme relevant for building phase dimension of COMPONENT

- ◆ **Homogeneous (non-)hierarchy** emerges in the existence of at most one relevant relationship:
 - Non-hierarchy** is given, if E_d is not involved into any relevant relationship. As an example, consider a non-hierarchical dimension RECORDER of fact scheme WORKFLOW in Figure 6.8.
 - Simple hierarchy** is given by an $n:1$ relationship between E_d and some other entity type E_i with $(1,1)$ as the structural constraint on E_d 's participation (full roll-up of E_d to E_i). For instance, PHASE and WORKFLOW yield a simple hierarchy.
 - Non-strict hierarchy** is given by an $m:n$ relationship between E_d and some other entity type. Non-strict hierarchies are not supported by the conventional OLAP.
- ◆ **Heterogeneous hierarchy** emerges in the existence of an optional roll-up or a single set of relevant mutually exclusive relationships:
 - Optional hierarchy** is given by an $n:1$ relationship between E_d and some other entity type E_i with $(0,1)$ as the structural constraint on E_d 's participation as this relationship produces a partial roll-up of E_d to E_i .
 - Non-covering hierarchy** results from a set of related partial $n:1$ relationships. The partiality is given by $(0,1)$ as the structural constraint on E_d 's participation in each relationship. Besides, the diverging roll-up paths of E_d ought to converge at a later stage. An example of such partial related roll-ups is the relationship between CITY, STATE, and COUNTRY in Figure 6.3.
 - Specialization hierarchy** emerges from a specialization relationship of E_d into multiple subclass categories. As an example, consider a generalized category SYSTEM in Figure 6.4.
- ◆ **Multiple hierarchies** correspond to multiple relevant relationships that are mutually non-exclusive. Figure 6.9 shows the relationships of the category DATE as an example of multiple hierarchies.
 - Alternative hierarchies** result from multiple roll-up relationships towards mutually related entity types. For instance, the relationships of DATE with CAL_MONTH and with CAL_WEEK are alternative, since the latter two categories have a many-to-many relationship with each other.
 - Parallel hierarchies** correspond to multiple roll-up relationships towards unrelated entity types. For instance, the relationship of DATE with CAL_MONTH is parallel to that of DATE and WEEKDAY.

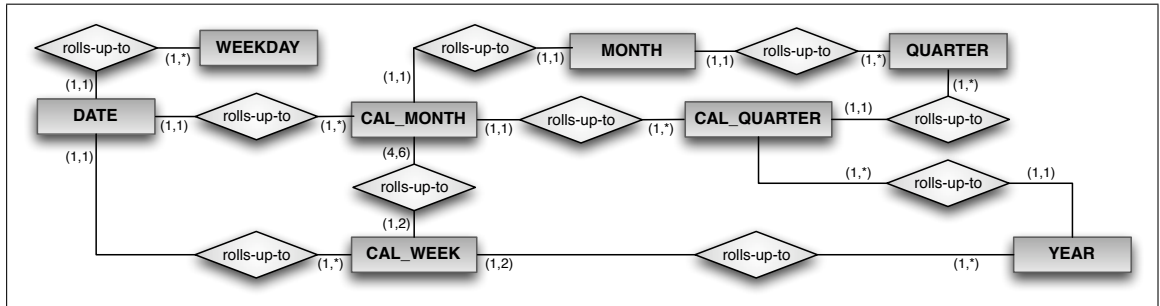


Figure 6.9: Multiple alternative and parallel hierarchies in DATE dimension

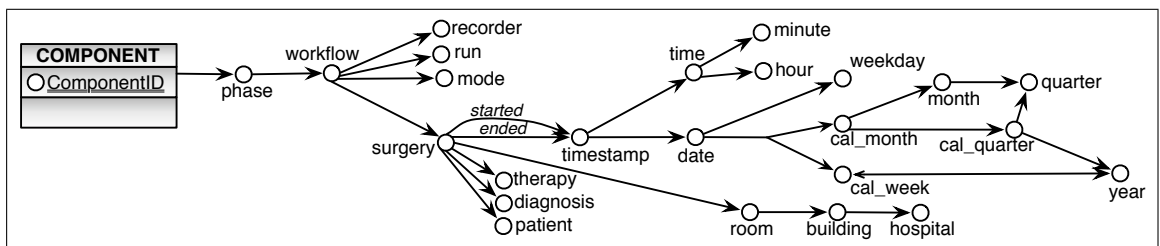


Figure 6.10: The resulting dimension scheme of the PHASE dimension in COMPONENT

Figure 6.10 shows the result of converging the fragment of the E/R model from Figure 6.8 into a dimension. Additionally, the structure of the hierarchical category DATE corresponding to the E/R scheme from Figure 6.9 is shown. Once the construction of the dimension scheme is complete, an abstract top category is added as a root node at which all dimension's hierarchies converge. In case of the unified multidimensional space, redundant elements of dimension schemes have to be eliminated by merging sets of compatible categories as described in Section 5.4 of the previous chapter.

6.3 Evaluation of the Proposed Framework

In this chapter we provided a methodology for obtaining multidimensional schemes from existing conceptual models of operational data sources. We advocate that the proposed cardinality-based modeling approach is more adequate for non-conventional data warehousing applications than the classical measure-centric methodologies. The advantages become manifest especially when dealing with operational or process-oriented data warehouse scenarios, in which non-measurable fact schemes prevail and metrics of interest are too versatile to be fully captured at design time.

We invoked the presented methodology for designing a data warehouse for Surgical Workflow Analysis. Once the conceptual modeling stage is accomplished, it is mapped to a logical model and implemented as a physical one in the selected backend system. Thereupon, cube designer tools can be used to define desired multidimensional views from the available data.

In this concluding section of the chapter we present two usage scenarios, in which the designed data warehouse filled with real data experimentally acquired at ICCAS from a series of running surgical interventions was used to solve typical SWA tasks. Data acquisition is performed using a graphical workflow editor

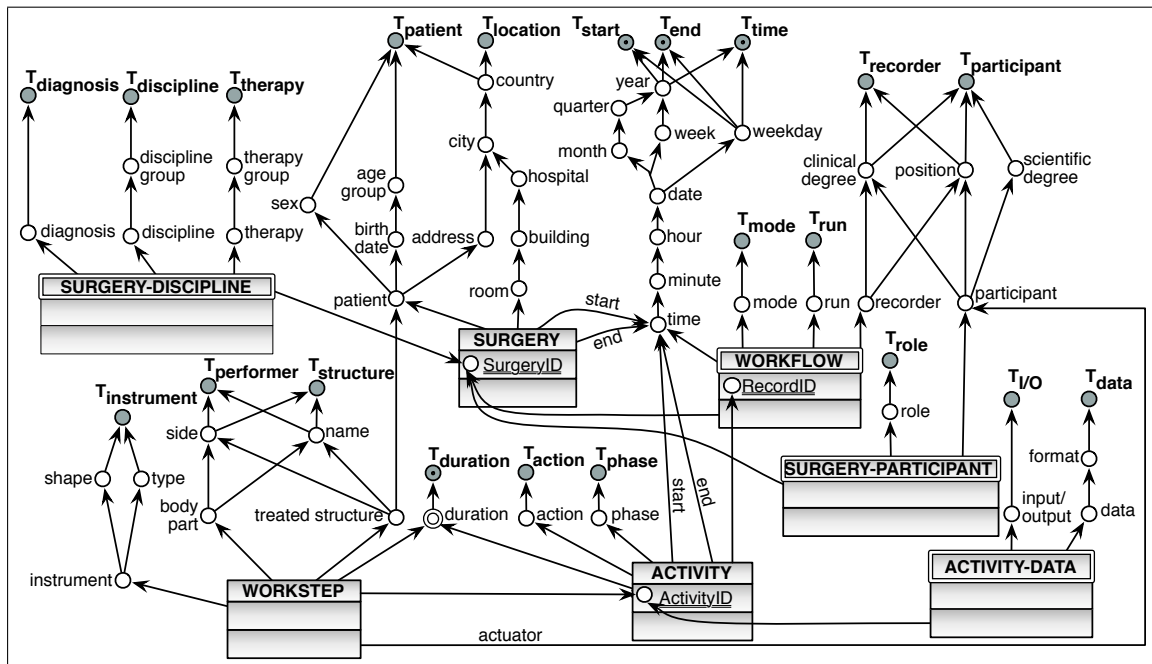


Figure 6.11: Multidimensional scheme of a surgical workflow structure

interface, implemented at ICCAS [126]. A trained recorder generates the workflow data instantaneously by monitoring a running surgical intervention. The originally obtained flow often contains errors, inconsistencies, and gaps due to time pressure, limited visibility or audibility, human or software failures. Therefore, prior to transferring the workflow to the data warehouse, the data is placed into temporary storage for verification, correction, and completion. Two methods of reducing recording errors are applicable, whereas the choice of the method depends on the configuration of the operating environment and the resources available:

1. Multiple recorders are employed to generate multiple versions of the workflow. These versions are combined to produce the final workflow.
2. A video recording of a surgery is produced to be used as a reference for post-processing the workflow.

Figure 6.11 shows the underlying conceptual model of a surgical workflow (some parts of the scheme irrelevant for the considered usage scenarios are omitted in order to unclutter the presentation). The final workflow of each intervention is stored with the characteristic run=0. Only the final workflows are considered for answering end-user queries.

6.3.1 Usage Scenario 1: Discectomy Surgery

The first usage scenario is concerned with instrument usage analysis in surgical interventions of type discectomy, which is an intervention at the spine. The goal of a discectomy is the partial removal of the herniated intervertebral disc. The objective of this sample analysis itself is to estimate the potential benefit of modifying the surgery by introducing an alternative surgical assist system. Typical expert queries in this scenario focus on the use of different conventional surgical instruments that have the same surgical objective.

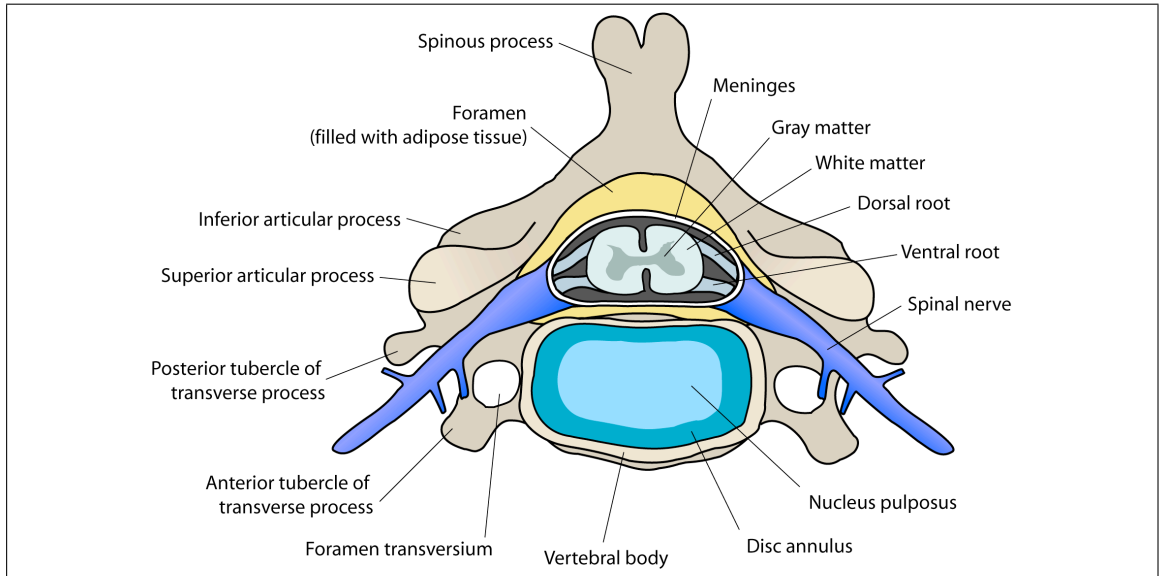


Figure 6.12: Annotated diagram of vertebra as an anatomic structure affected by a discectomy intervention

During a discectomy, parts of the vertebra are removed to assess the underlying intervertebral disc. Figure 6.12, adopted from [187], shows the main elements of vertebra and should give the reader some insight into the affected anatomic structure. Figure 6.13 shows a computer-tomographic image of a rapid prototyping model of the human spine (cross-section): the intervertebral disc (visible as white segments) is hidden from surgical access in the center angle under the bone material. The red-line marks the approximate volume of the vertebra to be removed by the surgeon to gain access to the intervertebral disc in order to remove it.

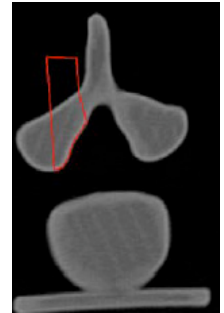


Figure 6.13: Spine cross-section view

To minimize invasiveness at the patient's body, the access area to the spine is spatially restricted. The two steps of ablating vertebra material and removing the disc are performed iteratively, i.e., the surgeon ablates only a small part of the vertebra, subsequently removing as much tissue of the intervertebral disc as he/she can reach, and then decides whether further access is needed. If so, the surgeon ablates the next portion of the vertebra and removes the tissue again, and so on.

Conventional bone ablation at the vertebra is performed using different surgical instruments, such as surgical punch, trephine, or surgical mallet/chisel. Each of the instrument types is available in different sizes and has different properties regarding invasiveness or handedness. Instrument usage patterns in terms of frequency and duration of usage during a discectomy can be obtained by aggregating the corresponding data from the protocols of surgical intervention.

In a visual OLAP tool, the required aggregates can be obtained by performing a series of simple interaction steps. Figure 6.14 contains the results of the first two of the following four sample queries, obtained by dragging the fields of interest into a pivot table interface.

Query I. *For each intervention of type discectomy and each of the specified bone ablating instruments, return the number of those work steps, in which that instrument was used by a surgeon.*

The query is answered by specifying a new measure Occurrence, defined as COUNT(*) (simple counting of qualifying fact entries), in fact table WORKSTEP. The aggregates are then computed as a roll-up of Occurrence by Surgery, Instrument Type and Instrument with selection conditions on Instrument Type ('bone ablating') and on Participant ('surgeon').

Query II. For each intervention of type discectomy and each of the specified bone ablating instruments, return the average duration of a work step, in which that instrument was used by a surgeon.

The query is answered by specifying a new measure Average_duration, defined as AVG(Duration), in fact table WORKSTEP and performing the same roll-up operation as in Query I.

In the backend, the two interactively specified queries, the results of which are shown in the pivot table (Figure 6.14), are answered by executing the following SQL statement:

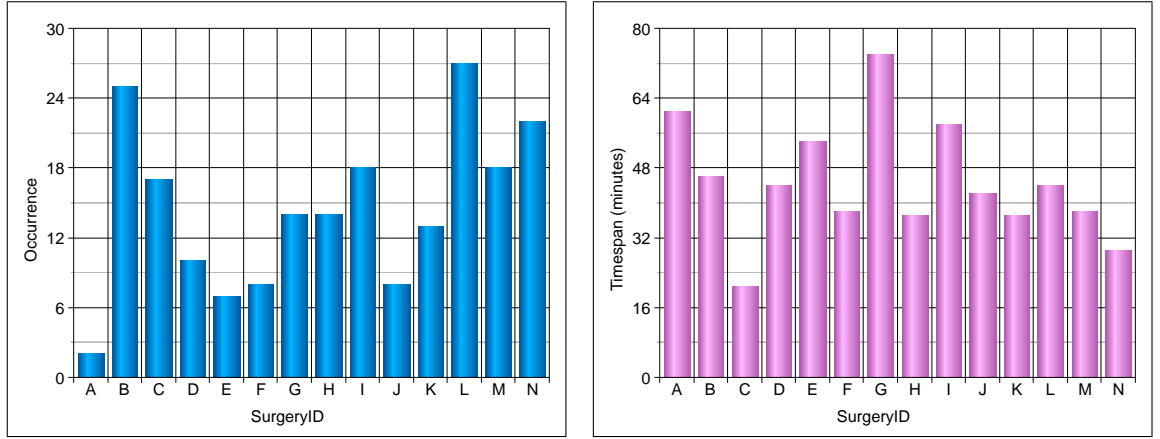
```
SELECT SURGERY.SurgeryID, INSTRUMENT_TYPE.Name, INSTRUMENT.Name,
       COUNT(*) AS Occurrence, AVG(Duration) AS Average_duration
FROM WORKSTEP, INSTRUMENT, INSTRUMENT_TYPE, ACTIVITY,
     WORKFLOW, SURGERY, PARTICIPANT, POSITION
WHERE SURGERY.SurgeryID IN
      (SELECT SurgeryID FROM SURGERY_DISCIPLINE WHERE DisciplineID IN
        (SELECT DisciplineID FROM Discipline WHERE name = 'discectomy'))
AND WORKSTEP.InstrumentID = INSTRUMENT.InstrumentID
AND INSTRUMENT.TypeID = INSTRUMENT_TYPE.TypeID
AND INSTRUMENT_TYPE.Name = 'bone ablating'
AND WORKSTEP.ActuatorID = PARTICIPANT.PersonID
AND PARTICIPANT.PositionID = POSITION.PositionID
AND POSITION.Name = 'surgeon'
AND WORKSTEP.ActivityID = ACTIVITY.ActivityID
AND ACTIVITY.RecordID = WORKFLOW.RecordID
AND WORKFLOW.Run = 0
AND WORKFLOW.SurgeryID = SURGERY.SurgeryID
GROUP BY ROLLUP (SurgeryID, INSTRUMENT_TYPE.TypeID, INSTRUMENT_TYPE.Name,
                 INSTRUMENT.InstrumentID, INSTRUMENT.Name)
```

		Measures							
		● Occurrence				● Average duration			
Dimensions		SurgeryID							
Instrument Group	Instrument	A	B	C	D	A	B	C	D
- bone ablating	mallet/chisel	0	3	1	1	00:00	00:23	00:34	00:50
	punch	9	22	10	9	02:38	00:35	00:46	01:27
	trephine	3	0	7	0	02:18	00:00	00:43	00:00
bone ablating Total		12	25	18	10	02:33	00:33	00:45	01:24

Figure 6.14: Instrument usage statistics as a pivot table

Query III. For each intervention of type discectomy, return the number of those work steps, in which a surgeon used any bone ablating instrument.

The result of this query is obtained from the results of Query I as a roll-up step by removing Instrument from the grouping set. The results of the query arranged into a bar-chart are presented in Figure 6.15a.



(a) Total number of bone ablation steps (Query III)

(b) Total timespan of bone ablation phase (Query IV)

Figure 6.15: Occurrence and duration of bone ablation steps in discectomy interventions

Query IV. For each intervention of type *discectomy*, calculate the total time span between the begin of the first and the end of the last ‘bone ablating’ activity.

The query is answered by defining a new composite measure Timespan as $\text{MAX}(\text{End}) - \text{MIN}(\text{Start})$ in fact table ACTIVITY. The aggregates are computed as a roll-up of Timespan by Surgery with a selection condition on Action (‘bone ablation’). A bar chart with the results of this query is shown in Figure 6.15b.

The above queries describe a real-world example from the field of medical engineering. The aggregates obtained in the above queries describe the usage pattern for bone ablating instruments and provide crucial information for predicting the success of a new surgical instrument in this field [130]. This new system is a power driven milling system, whose evolution speed is controlled by its spatial position in relation to the patient’s body [71]. The system is intended to replace the conventional bone ablating instruments and to enable the surgeon to perform the entire removal procedure in a single work step.

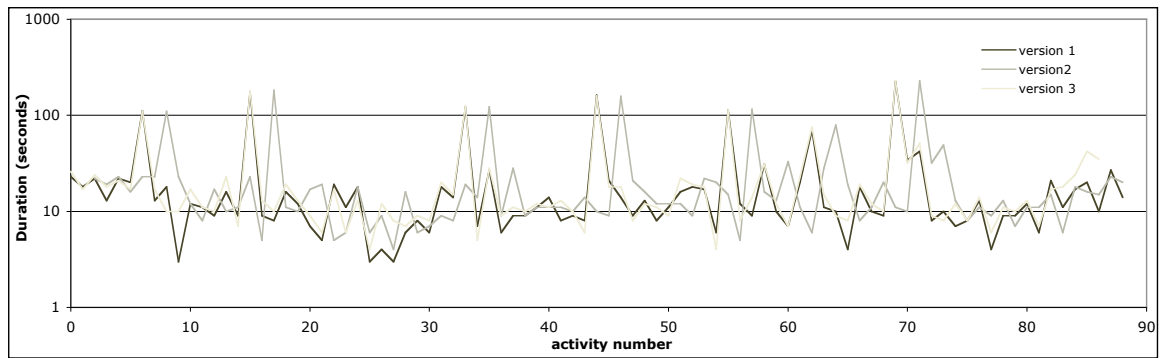
6.3.2 Usage Scenario 2: Functional Endoscopic Sinus Surgery

The second usage scenario presents the experimental results of recording three cases of *functional endoscopic sinus surgery* (FESS) in the discipline *ENT surgery*. Ten workflow versions of each case were generated by respectively ten recorders with a similar level of training. The task was limited to storing only the data of type WORKSTEP and its dimensions. Each surgical case was executed by one surgeon and one assistant. Table 6.1 shows further statistics about the obtained data for each of the three cases. Incompleteness of data records was caused primarily by missing values for Treated Structure, presumably due to insufficient visibility or lack of time to enter the values as Treated Structure appears as the last input field in the recording software.

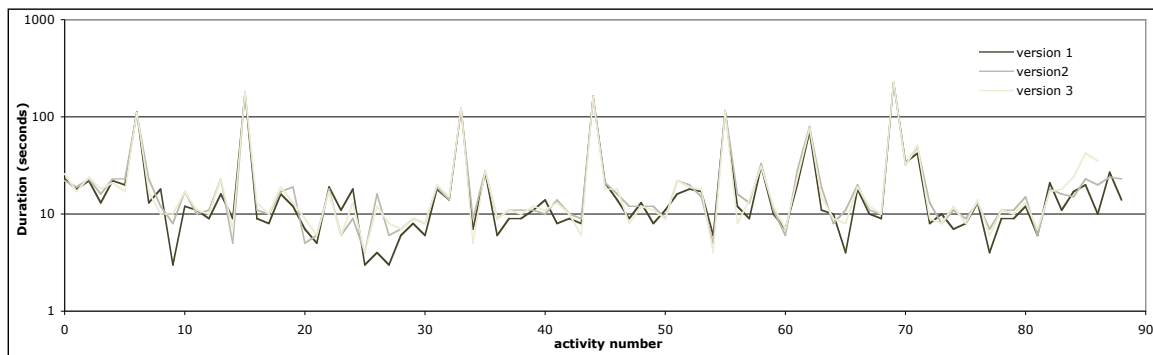
Figure 6.16 shows an example of using line-charts to validate the quality of the obtained data flows. Both line-charts show three recordings (versions) of the same intervention. The X-axis corresponds to the start time dimension of ACTIVITY, normalized to display simple auto-incremented timestamps (activity number

Table 6.1: Statistical overview of the acquired data

Parameter	Case 1	Case 2	Case 3
Total number of records	590	848	651
Portion of incomplete records	5.9%	4.2%	4.3%
Minimum number of records per workflow	57	77	61
Maximum number of records per workflow	63	89	68
Number of distinct action types	14	12	20
Number of instruments	11	11	13
Number of treated body structures	6	6	5
Average activity duration (seconds)	37.5	25.3	34.9
Minimum activity duration (seconds)	12.9	6.6	9.8
Maximum activity duration (seconds)	81.3	106	135.2



(a) Three versions of a workflow with a synchronization error



(b) Three versions of a workflow with correct synchronization

Figure 6.16: Synchronizing multiple workflow versions of the same surgical intervention

Measure	frequency				duration (minutes)			
Dimension	Surgery				Surgery			
Instrument	1	2	3	all	1	2	3	all
Blakesley forcep	16	29	16	61	10.5	6.9	7.4	24.8
Camera	2	2	3	7	0.8	0.2	0.5	1.5
Curette, sharp			3	3			2.4	2.4
Elevator, doble-ended	2	2	3	7	0.7	0.9	1.1	2.7
Forceps		2	2	4		0.7	0.4	1.1
Nasal speculum	2	2	2	6	0.7	0.8	0.5	2.0
Optics + Endoscope	14	14	15	43	18.0	17.9	18.8	54.6
Sterile towels		1	1	2		0.3	0.1	0.4
Suction unit	19	35	16	70	4.3	5.6	4.1	836
Syringe	2	2	2	6	0.7	0.6	0.3	1.5
Swab	2	2	2	6	0.7	0.7	0.4	1.8
Tamponades	2	2	2	6	0.5	0.4	0.4	1.3
Telephone			3	3			0.9	0.9
Total	61	93	70	224	36.8	34.9	37.2	108.9

Figure 6.17: Pivot table view of the instrument usage statistics

1, 2, ..., n) instead of absolute time values. The Y-axis shows the duration of each step. As expected, the trajectories of the resulting graphs are very similar. However, in the original data shown in Figure 6.16a, an anomaly is evident: while versions 1 and 3 appear well synchronized, version 2 seems shifted in time. The chart in Figure 6.16b shows the same data, with version 2 shifted backwards by two steps. Now, all versions appear correctly synchronized. Similar visualizations can be employed to instantaneously detect and resolve recording inconsistencies that may otherwise be extremely tedious to identify.

Once the “cleansed” data has been written to the data warehouse in form of final workflow versions, it is available for analysis and exploration. Figure 6.17 shows a pivot table with the results of the following query:

Query I. For each FESS intervention, calculate the frequency of using each instrument as well as the total duration of each instrument’s usage throughout the intervention (i.e., the number and the duration of the respective work steps).

A two-dimensional view of the instrument usage numbers with totals and subtotals in Figure 6.17 reveals valuable insights to surgeons and medical engineers. For instance, it confirms the expectation of similar usage patterns for most of the instruments across multiple instances of the same surgery type. Also, one can notice that two instruments were used only in the third case, which could be an indication of an exception in the workflow. Combination of frequency and duration in the same query is helpful for analyzing the intensity of usage. For instance, *Blakesley forcep* and *Suction unit* were used with similar frequency (61 and 70, respectively), however, with immense difference in duration (24.8 versus 836 minutes).

Availability of the multidimensional perspective of fine-grain workflow data enables effortless retrieval of relevant measures by the end-users, whereas advanced OLAP frontends can be employed for more sophisticated analysis tasks, such as pattern recognition, anomaly detection, etc. In Chapter 8 we continue using the SWA usage scenario for demonstrating advanced visual exploration options, such as ad hoc specification of user-defined measures.

Chapter 7

Relational Implementation of the Multidimensional Data Model

THE AIM OF THIS CHAPTER is to demonstrate the implementability of the proposed extended multidimensional model in a state-of-the-art ROLAP system. We elaborate on the aptitude of the relational technology for handling complex multidimensional structures and formulate general guidelines for logical design. A well-established requirement of summarizability is used to determine the necessary scheme and/or instance transformations. Major challenges arise when dealing with complex dimension hierarchies. We propose a two-phase normalization approach: in the first phase, heterogeneous hierarchy schemes are transformed to eliminate incomplete and overlapping specialization, mixed-grain, and non-covering roll-ups; in the second phase, multiple and generalized hierarchies are decomposed into constituent homogeneous subtrees in order to identify non-onto and non-strict mappings and eliminate those via instance normalization. Finally, we consider the issue of the metadata, which acts as an intermediary between the relational model and the application layer by capturing the multidimensional semantics behind “plain” data tables.

Contents

7.1	Mapping the Multidimensional Model to the Relational Model	132
7.1.1	Mapping Fact Schemes	133
7.1.2	Handling Derived Elements	134
7.1.3	Mapping Dimension Hierarchies	135
7.2	Mapping Heterogeneous Hierarchy Schemes	136
7.2.1	Mapping Non-Covering Hierarchies	136
7.2.2	Mapping Generalized Hierarchies	140
7.3	Enforcing Summarizability in Homogeneous Hierarchies	148
7.3.1	Mapping to Covering	149
7.3.2	Mapping to Strict	150
7.3.3	Mapping to Onto	153
7.4	Metadata for the Analysis Layer	154
7.4.1	Overview of the CWM	155
7.4.2	Representing multidimensional properties in the CWM	162

7.1 Mapping the Multidimensional Model to the Relational Model

In Section 2.2.3 we highlighted the major data warehouse implementation alternatives and explained why we favor the relational OLAP architecture as the target platform for implementing our extended conceptual model. We also discussed various logical design options for the relational implementation, such as star, snowflake, galaxy, fact constellation, and star cluster schemata. We adopt the galaxy schema based on normalized storage of dimension hierarchies, as in the snowflake schema, with centralized, i.e., non-redundant, maintenance of semantically related categories. Advantages of this schema can be summarized as follows:

- ◆ Normalized storage of dimension hierarchies enforces consistent maintenance of data hierarchies by using foreign key constraints for mapping child-parent relationships.
- ◆ Hierarchical structure of a dimension is explicit in the logical schema as each dimension level is mapped to a separate dimension table referencing its parent level(s).
- ◆ Non-redundant storage guarantees anomaly-free maintenance of dimension hierarchies.
- ◆ Since each category is mapped to a separate dimension table, unification of the multidimensional space becomes trivial: each set of conform or compatible categories is managed in a centralized dimension table of the respective category type.
- ◆ Properties of a particular dimension level can be easily associated with that level by adding them as attributes to the respective dimension table.
- ◆ Semantically related fact schemes and dimensions are mapped to related logical structures. Valid roll-up paths can be derived simply by “tracing” the foreign key relationships between tables.
- ◆ Interchangeability of fact and dimension roles is given as both are mapped to relational tables.
- ◆ Interchangeability of dimension category, property attribute, and measure role is given as all characteristics are stored as attributes in relational tables.
- ◆ Non-strict mappings can be handled using an established data warehousing practice of bridge tables.
- ◆ Inheritance in fact and dimension schemes can be handled using (materialized) views and/or integrity constraints (e.g., triggers).

A conceptual-to-relational mapping of the multidimensional data model is a thoroughly investigated field of data warehousing, with many established methodologies and designer tools, which foster automated acquisition of logical and physical schemes from the conceptual ones. However, existing approaches perform poorly or even fail when it comes to handling extensions and modifications of the multidimensional model. To identify the actual challenges of providing a logical mapping of our extended model, we proceed by “reviving” the general rules of the relational data warehouse design according to the galaxy schema [7, 8, 92]:

- ◆ The two types of tables are a *fact table* and a *dimension table*.
- ◆ Each fact type is mapped to a relation of type fact table that includes all measure characteristics of the fact type as well as a foreign key reference to each associated dimension.
- ◆ Each dimension level is mapped to a separate dimension table that includes all attributes of that level as well as a foreign key reference to each associated parent dimension level.
- ◆ If a roll-up relationship between a pair of dimension levels is non-strict, it may not be referenced via a foreign key from within the child level table. Instead, such relationship is extracted into a so-called “bridge table” that includes a foreign key reference for each of the two affected dimension levels.
- ◆ The fact table is associated with each of its dimensions by referencing the dimension’s bottom level, i.e., each dimension is represented in the fact by the primary key attribute of its bottom category.
- ◆ A partial roll-up relationship is expressed by declaring the respective parent key reference as nullable.
- ◆ Each set of semantically related categories (conform or compatible ones) is maintained centralized in the same dimension table.

The above relational model with “snowflaked” dimension hierarchies, arranged into galaxies by sharing semantically related categories, produces a rather self-describing logical scheme. Fact and dimension category types are mapped to fact and dimension tables, respectively, and roll-up relationships are represented by foreign keys. Related fact schemes are also evident as their logical schemata share dimension tables. However, the relational model is incapable of capturing multidimensional semantics in its entirety. For instance, it provides no constraints for specifying measure additivity, exclusivity or compatibility of multiple hierarchies, degenerate fact types, etc. Semantic description of the logical schema in a data warehouse system is handled by the metadata layer. The metadata repository is even considered the key component in the data warehouse architecture [41] as it manages all the information necessary for successful interplay of all system’s components. Metadata management is the subject of the last section in this chapter.

Application of the relational data warehouse design principles is rather straightforward for most of the multidimensional constructs. Ambiguities or complications arise when dealing with non-conventional properties, such as degenerated and derived elements or irregular dimension hierarchies. In the remainder of this section we discuss the overall process of obtaining logical representations of all types of fact and dimension schemes supported by our extended model. Logical schemes are constructed using the UML profile for logical data warehouse design proposed in [102] and introduced in Section 2.2.4.

7.1.1 Mapping Fact Schemes

Each non-derived fact type is mapped to a separate fact table consisting of the fact’s measures and their dimensional characteristics. The relational model does not distinguish between normal and degenerate facts. Fact tables of non-measurable facts consist solely of the respective dimensional characteristics, which, together, build a fact entry. If a fact identifier dimension exists in the scheme, it is used as a primary key of the fact table. Otherwise, the entire set of the dimensional attributes form a composite key or, alternatively, a surrogate (i.e., system-generated) fact identifier attribute can be added to the scheme. As an example, consider a multidimensional fragment consisting of a non-measurable fact SURGERY and its degenerate measurable fact SURGERY-PARTICIPANT in Figure 7.1a with the corresponding logical implementation shown in Figure 7.1b. As expected, each fact scheme maps to a fact table of its own. The fact table of a measurable fact

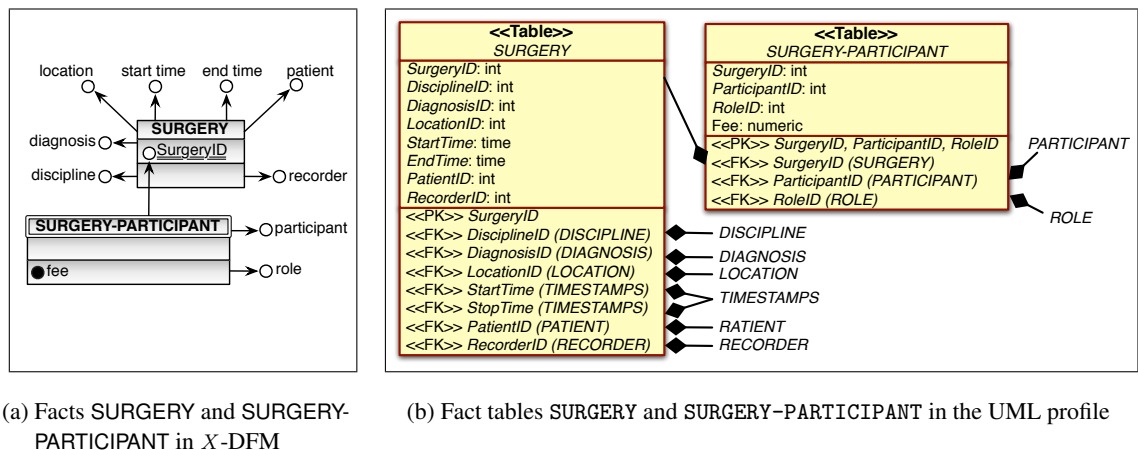


Figure 7.1: Example of conceptual fact schemes and their logical representations

contains each measure as an attribute (e.g., *Fee* in *SURGERY-PARTICIPANT*), whereas the fact table schema of a non-measurable fact consists solely of dimension attributes.

Dimension attributes of a fact table correspond to fact-dimensional roll-up relationships and are specified as foreign key references to the respective dimension tables (e.g., *PatientID* in *SURGERY* references dimension table *PATIENT*). Multiple roles of the same dimension type in a fact scheme result in multiple foreign key references targeting the same dimension table. For example, dimension table *TIMESTAMPS* is referenced as *StartTime* and as *EndTime* characteristic in fact table *SURGERY*.

The only kind of dimension attribute not referencing another table is the one representing a degenerated dimension, which exists only in the context of its containing fact scheme and, therefore, is not extracted into a dimension table of its own. In our example, *SurgeryID* is a degenerated dimension of *SURGERY*. Moreover, *SurgeryID* is a fact identifier and, as such, it qualifies as a primary key in *SURGERY*. In fact schemes with no fact identifier, the primary key is composed of the whole set of dimension attributes, as in *SURGERY-PARTICIPANT*, where *SurgeryID*, *ParticipantID*, and *RoleID* build the key.

Notice that there is no evident distinction between fact tables of fully-fledged and degenerate facts. An indication of degeneration is given, if the fact table contains a dimension attribute referencing another fact table. For example, the schema of *SURGERY-PARTICIPANT* contains the attribute *SurgeryID*, i.e., references *SURGERY* as a dimension table. Thereby, the relational representation inherently resolves the duality of facts and dimensions: both types are mapped to relations and both kinds of roll-up relationships – fact-dimensional and intra-dimensional ones – are implemented as foreign keys. Subsequently, metadata is used to specify the role of each table in a particular fact's context.

7.1.2 Handling Derived Elements

According to the rules of “good” database design, derived elements – attributes or relations – should not be materialized to avoid redundancy and update anomalies. Since the instance of a derived element can be calculated by querying its base element(s), views (virtual relations) should be employed for storing the definition of such elements. In the data warehouse design, however, materialization of calculated data views is a common practice, aimed primarily at boosting the performance.

DERIVED FACTS

Typically, materialized views are constructed as projections of primary facts by aggregating their measures to a subset of the fact's dimensions and/or to a desired granularity within a dimension. Such aggregated views are commonly referred to as “cube computations” or “summary tables” and represent derived multi-dimensional structures of type *fact*. Further examples of derived facts are those obtained as an outcome of scheme-transforming OLAP operators, such as drill-across, push, and pull. In the UML notation, derived facts are modeled using the *view* element, linking the former to the fact's base tables. As an example, consider the derivation of fact *DISCIPLINE-PARTICIPANT* defined as a drill-across of *SURGERY*'s two degenerate facts *SURGERY-PARTICIPANT* and *SURGERY-DISCIPLINE* (the conceptual scheme of the base facts is depicted in Figure 6.11). The resulting fact scheme *DISCIPLINE-PARTICIPANT* should include dimensions *discipline* and *participant* and a measure *surgeries* enumerating the number of surgeries a participant had in the respective discipline. Figure 7.2 shows the logical schema fragment of this derived fact and its input fact tables.

At the logical design stage, it is not mandatory to distinguish between standard and materialized views since the actual implementation decision can be taken during the physical design phase depending on storage availability and other constraints.

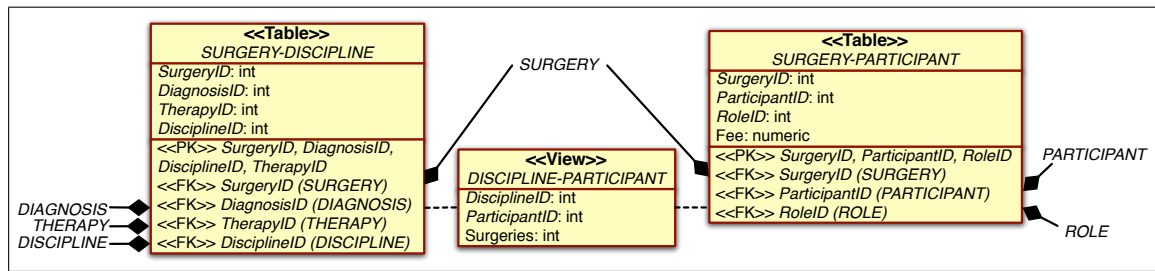


Figure 7.2: Derived fact modeled as a view linked to the base fact tables

DERIVED MEASURES AND DIMENSION CATEGORIES

Besides derived facts, our conceptual model supports derived elements of type *measure* and *dimension category*. To our knowledge, there exist no established guidelines in the data warehouse design for handling these kinds of derivation. We have come to realize that in most cases it is unfeasible to implement derived measures and dimension categories using the (materialized) view mechanism.

In case of a measure, materialized storage of its values in the fact table is crucial for query performance. The question is how and where exactly derived measures should be stored. We distinguish between *simple* and *complex* derivation.

In the simple case, a new measure is computed from one or more measure attributes of the same fact scheme and, therefore, it has the same set of dimensional characteristics as its input measure(s). For example, net-profit can be computed as a difference between revenue and expenditure. If the new measure were to be realized using a materialized view, the view would have to include the entire set of dimensional attributes and the same number of fact entries as the base fact table. Obviously, it is much cheaper to materialize just the measure attribute itself by adding it as a column to the base fact table. Since existing fact entries are not subject to modification, there is no overhead for guarding the consistency of derived measures.

Derivation is complex, if the granularity of the derived measure differs from that of its input measure(s). In such a case, it would be incorrect to materialize the derived attribute in its base fact scheme. Therefore, complex measure derivation is considered to be a special case of fact derivation and is handled accordingly, i.e., using materialized views.

Derived dimension categories represent an additional challenge of the relational design as dimension tables need to have full-fledged schemata in order to be usable as targets of foreign key relationships and to reference other dimension tables. Therefore, derived categories are mapped to tables rather than to views. Initial data loading is performed using an appropriate query while subsequent maintenance relies on triggers (updates of the base dimension tables are propagated to the table of the derived category).

7.1.3 Mapping Dimension Hierarchies

“Snowflaked” dimension storage is achieved by structuring and normalizing dimension hierarchies according to the Third Normal Form (3NF) [41]. As a result, each category (or category type) is mapped to a dimension table of its own, containing the respective dimension level attribute along with all property attributes as well as a foreign key attribute for each outgoing roll-up relationship of that category. Surrogate keys are commonly used to achieve efficient implementation of foreign key references.

Normalized hierarchy maintenance has a number of advantages when dealing with complex dimensions. For example, it provides a more stable context for adding new hierarchies: new roll-up relationships of an

existing category can be introduced without affecting other categories in the dimension. Snowflake schema appears to be the most appropriate solution for dimensions with complex schemes, e.g., with a large number of levels or containing multiple related and/or independent hierarchies. However, the 3NF appears too restrictive or provides no guidelines for modeling partial related roll-up relationships and irregular hierarchies. For instance, consider a non-covering hierarchy of project in Figure 7.6a. Intuitively, its relational mapping should consist of the dimension tables PROJECT, OFFICE, BUILDING, and CITY. As for foreign key relationships, the bottom level project has a twofold roll-up relationship with city, a direct and a transitive (via office and building) one. Consequently, two conflicting mapping options arise: a reference to CITY should be stored within PROJECT, on the one hand, and within its upper level BUILDING, on the other hand.

Non-strict roll-up relationships represent another challenge: existence of multiple parent-level values for the same element makes it impossible to map this relationship to an atomic-valued parent column.

Apparently, major challenges of mapping dimension schemes to relations arise when dealing with heterogeneous, ragged, non-balanced, and non-strict hierarchies. Our approach to handling such non-summarizable mappings is based on the concepts of *controlled heterogeneity* and *controlled non-strictness*, which restore summarizable aggregation behavior within a hierarchy by forcing the latter to undergo a series of transformations at the scheme level and, subsequently, at the instance level. The first phase is concerned with scheme transformation of heterogeneous hierarchies: a dimension scheme is normalized as to eliminate incomplete and overlapping specialization, mixed-grain, and non-covering roll-up relationships. In the second phase, instance normalization is conducted: multiple and generalized hierarchies are decomposed into constituent homogeneous subtrees in order to identify non-onto and non-strict mappings and eliminate those by applying instance normalization techniques. In the next two sections, the above two phases of normalizing non-summarizable hierarchies and obtaining their relational mappings are described in detail.

7.2 Mapping Heterogeneous Hierarchy Schemes

Relational data warehouse systems have no methodological guidelines for handling heterogeneous dimensions. Obviously, the concerns of summarizability and adequate mappings of all aggregation paths remain valid in this context. To realize the requirements of supporting heterogeneity, let us consider various types of heterogeneous hierarchies defined by our metamodel (see Figure 4.14). The first distinction is made between non-covering and generalized hierarchies: in the former, the members are simply allowed to skip levels, while generalization implies existence of superclass/subclass relationships between categories. We expect different analysis requirements for these two types and propose to handle their modeling accordingly:

- ◆ A *non-covering* hierarchy scheme is transformed into a summarizable homogeneous symmetric scheme by applying a well known instance normalization technique, which eliminates skipped levels by inserting “placeholder” nodes. Additionally, non-covering hierarchy can be transformed into a generalization, with the latter added as an alternative hierarchy to the homogeneous one.
- ◆ A *generalized* hierarchy is first normalized at the conceptual level by eliminating mixed grain and overlapping specialization. Subsequently, the inheritance relationships are mapped using object-relational inheritance mechanisms or imitations thereof.

7.2.1 Mapping Non-Covering Hierarchies

The traditional approach to handling a non-covering mapping is to normalize it into a covering one by inserting placeholder elements to fill the gaps of the missing parent nodes. However, this method is criticized for “polluting” the data with artificial entries and losing the original roll-up relationships. Besides, it is not clear

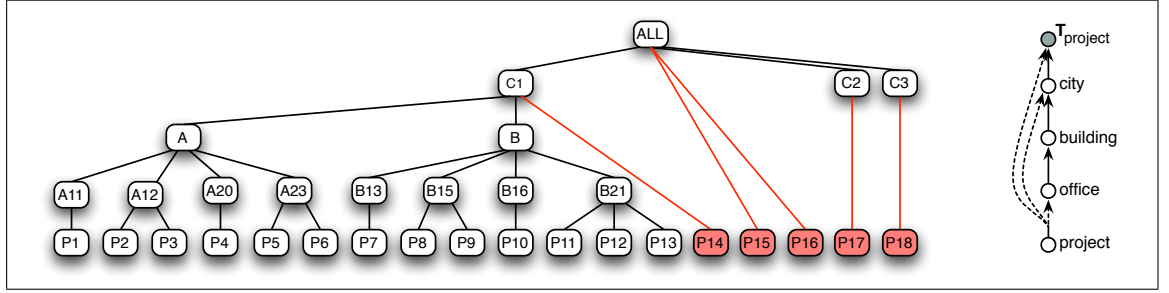


Figure 7.3: A non-covering hierarchy of project locations: instance (left) and scheme (right)

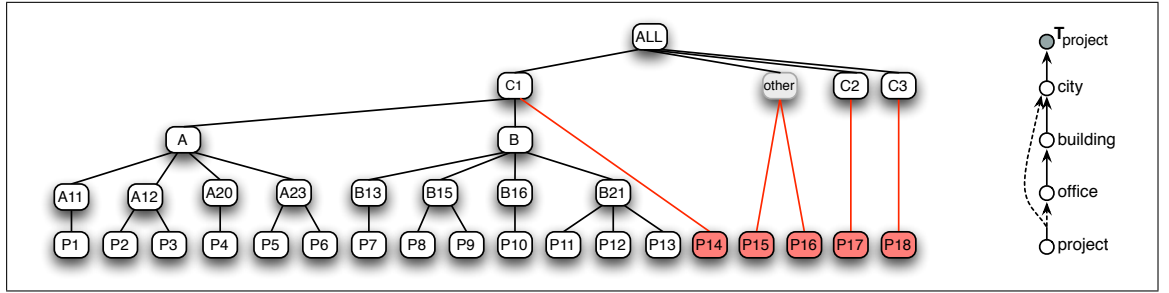


Figure 7.4: State of the non-covering hierarchy with city normalized to covering

how each new node should be named. We propose a slightly modified mapping-to-covering method, which partially overcomes the above shortages and provides a fully automated normalization routine.

As an example of a non-covering mapping, let us consider the location hierarchy in project dimension, with a sample instance and its underlying scheme shown in Figure 7.3. Member values with skipping edges are marked with red color. We use a slightly modified example than the one used previously in Figure 4.15 in order to increase the complexity of the non-covering behavior. In the current example, project members roll up either to office or directly to city, or have no location at all (i.e., roll up directly to the root).

Normalization of a non-covering hierarchy scheme \mathcal{H} and its instance H evolves as follows:

1. The hierarchy scheme is inspected top-down to identify levels at which skipping occurs. Category \mathcal{C}_j qualifies as a *non-covering level*, if it is targeted by a full roll-up edge $\mathcal{C}_k \sqsubseteq^{(\text{full})} \mathcal{C}_i$, on the one hand, and there exists at least one bypassing (i.e., rolling up to an upper level of \mathcal{C}_j) partial roll-up edge $\mathcal{C}_i \sqsubseteq^{(\text{part})} \mathcal{C}_m$, where $\mathcal{C}_i \sqsubseteq^* \mathcal{C}_j$ and $\mathcal{C}_j \sqsubseteq^* \mathcal{C}_m$, on the other hand. In our example, city, building and office represent a set of non-covering categories.
2. The instance of each non-covering level is traversed in order to identify missing member nodes:
 - ◆ If a non-covering category \mathcal{C}_j represents the highest aggregation level ($\mathcal{C}_j \sqsubseteq \top_{\mathcal{H}}$), a placeholder element named *other*¹ is added to the member set of \mathcal{C}_j and all skipping edges are re-assigned to roll up to this new element. In our example, depicted in Figure 7.4, such a node (marked with grey filling) is added to city as a parent of project members *P15* and *P16*. The scheme is modified accordingly by removing the partial roll-up relationship $\text{project} \sqsubseteq^{(\text{part})} \top_{\text{project}}$.

¹Another suitable name, such as *n/a*, *unknown*, *none*, etc., can be used instead of *other*.

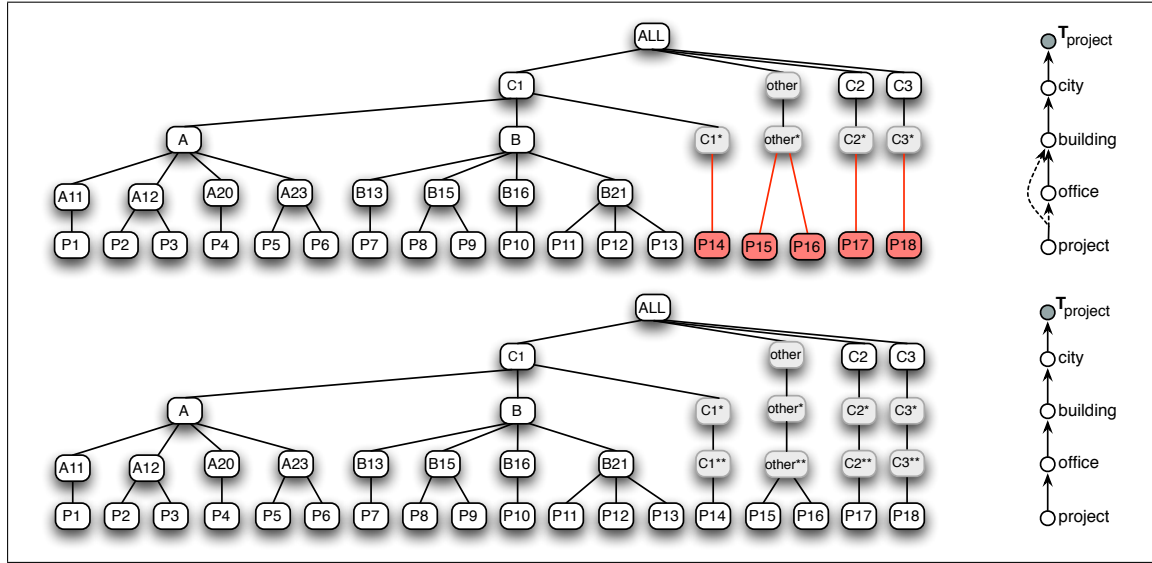


Figure 7.5: Mapping to covering by normalizing non-covering levels building (top) and office (bottom)

- ◆ A non-covering category C_i at any lower aggregation level is normalized by inserting a new member for each roll-up edge bypassing C_i and adjusting that edge to roll up to the respective new member in C_i . The states of the hierarchy after normalizing building and office levels are shown in Figure 7.5. The obtained hierarchy is homogeneous and symmetric.

3. The following naming convention is applied to placeholder elements: the name of the respective parent member, supplemented by a '*' symbol, is inherited². For example, skipping edge $P14 \subseteq C1$ (see Figure 7.4) is resolved by creating a placeholder element named $C1^*$ in building, resulting in edges $P14 \subseteq C1^*$ and $C1^* \subseteq C1$ (see Figure 7.5 (top)). In case of a recursive name propagation, the number of '*' symbols indicates the depth of the normalization.

The advantage of the proposed normalization approach along with the applied naming strategy is that it enables a fully automated generation of placeholder nodes with meaningful names, which contain the information about the original parent element (the propagated name) and the depth of the normalization (the number of supplement symbols).

The penalty of losing heterogeneity can be overcome by creating multiple alternative hierarchies: *i*) the covering hierarchy obtained by applying the above normalization technique and *ii*) a generalization hierarchy that extracts each aggregation path from the non-covering scheme into a homogeneous subclass hierarchy, as described in Section 4.4.3 and depicted in Figure 4.16. With our modified example of a non-covering hierarchy, the transformation into a generalization would yield three specializations of project: *i*) internal for instances that roll up to office, *ii*) external for instances that roll up to city, and *iii*) others for instances with no parent location category. Figure 7.6 illustrates the process of obtaining the final normalized hierarchy scheme: (a) is the original non-covering hierarchy scheme of project locations; (b) shows the two normalized schemes – a covering one (left) and a generalization (right); 7.6c is a final normalized scheme consisting of the two

²Any other symbol or string can be used as a supplement.

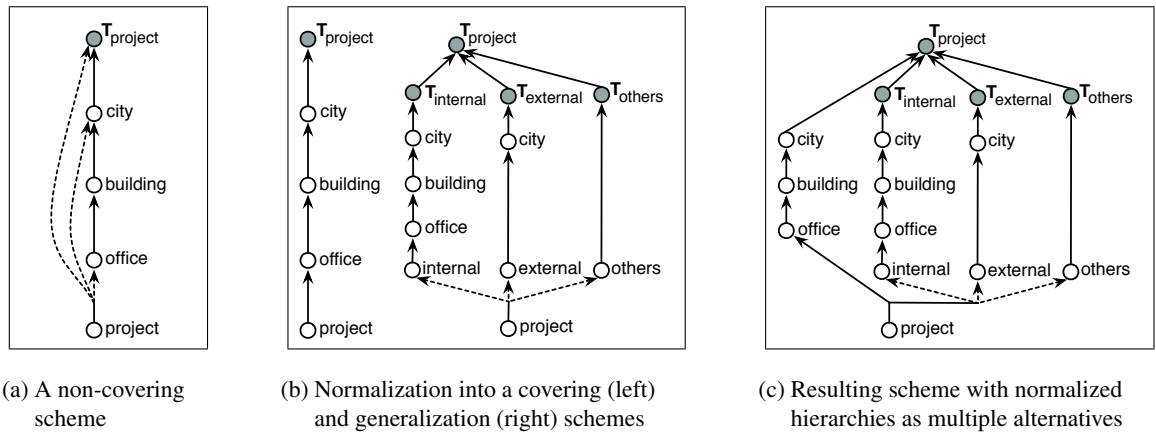


Figure 7.6: Transformation of a non-covering hierarchy into a set of multiple alternative hierarchies

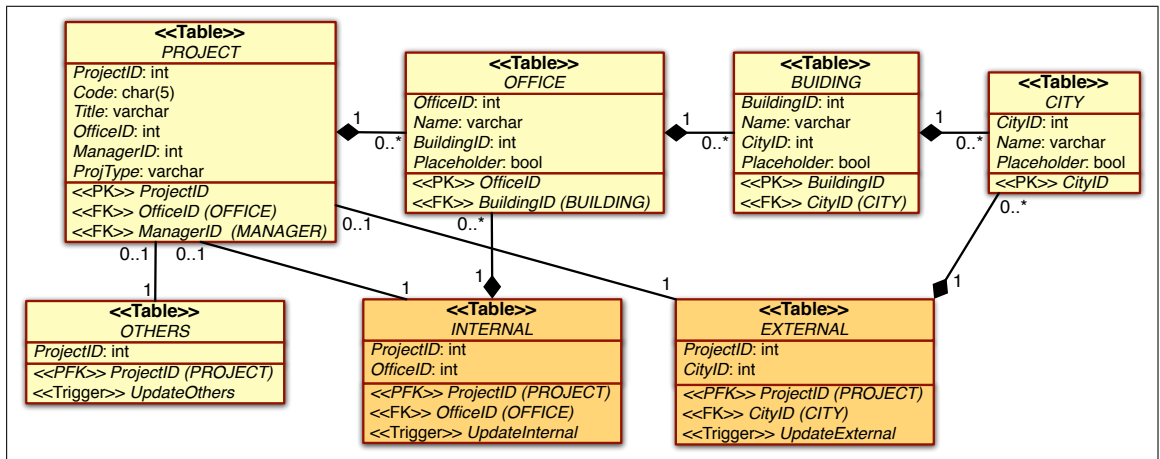


Figure 7.7: Logical schema of project location hierarchies

normalizations as alternative hierarchies. The obtained scheme is summarizable and supports aggregation of project instances either along the uniform (covering) path or along subclass-specific hierarchies.

The actual relational mapping of a balanced hierarchy according to the galaxy schema is straightforward: each level is stored in its own dimensional table that references the respective parent-level table. Back to our example, the four hierarchy levels are mapped to tables PROJECT, OFFICE, BUILDING, and CITY. Figure 7.7 shows the logical schema of project hierarchy. Dimension levels affected by the normalization to covering contain a boolean field Placeholder for marking placeholder entries.

The alternative specialization-based hierarchy is also present in the logical schema. There exist different approaches to the relational representation of inheritance. The actual choice depends on the capacities of the backend system (object-relational features, triggers, rules, materialized views, etc.) and storage constraints. We propose a rather simple solution based on *derived tables*, which do not require any advanced database

procedures. A derived table is similar to a materialized view: both are created as data copies populated with the data queried from other tables. However, derived tables are superior to materialized views as the former have a full-fledged scheme definition and, therefore, support the use of integrity constraints, such as primary and foreign keys, triggers, and check clauses. Foreign keys are imperative for linking fact tables to dimensions as well as for assembling dimension tables into hierarchies. Triggers and check constraints are useful for keeping the derived data consistent and up-to-date.

Back to our example, the two project specializations into internal and external are represented by the derived tables shown with a darker background color in Figure 7.7. In case of a non-covering hierarchy, its normalized covering instance is captured by a homogeneous hierarchy scheme. The alternative specialization hierarchy, however, is used to imitate the original non-covering instance and its heterogeneous hierarchy paths. Therefore, the bottom-level table PROJECT contains the properties of both alternatives: a foreign key *OfficeID* for the covering mapping and a *ProjType* attribute for storing the belonging to a specialization class of the non-covering mapping. That is why the specialization classes inherit only those properties of the generalized level, attributing to the non-covering mapping.

Specialization tables of a non-covering hierarchy do not have subtype-specific characteristics apart from the roll-up relationships. The logical mapping of such a specialization is a table containing the key attribute of the generalized type and the subtype-specific reference to the parent level. The table is populated via a query extracting the necessary data from the underlying relations. As an example, consider the maintenance of the derived table EXTERNAL, instantiated using the following SQL statement:

```
INSERT INTO EXTERNAL
SELECT PROJECT.ProjectID, BUILDING.CityID
FROM PROJECT, OFFICE, BUILDING
WHERE PROJECT.ProjType = 'External' AND PROJECT.OfficeID = OFFICE.OfficeID
AND OFFICE.BuildingID = BUILDING.BuildingID
```

The consistency of the data copy with respect to its sources is guarded using *triggers*. A trigger is a special kind of stored procedure that automatically executes when a specified event (such as insert, update, or delete) occurs in the specified data source. In case of the table EXTERNAL, its trigger function *UpdateExternal* is defined as to react to the changes in any of the source tables, i.e., in PROJECT, OFFICE or BUILDING.

7.2.2 Mapping Generalized Hierarchies

The conventional relational model offers three representation alternatives for handling generalization: *i*) creating a separate table for the superclass and for each of its subclasses, *ii*) creating only the subclass tables, with all superclass-level properties stored within each subclass, and *iii*) creating a single superclass table hosting all common as well as subclass-specific attributes and admitting NULL values for the latter. Presence of hierarchies aggravates straightforward application of these mappings to multidimensional data.

Some approaches on adopting standard relational mappings of generalization for ROLAP have been proposed in the literature. Jagadish et al. [70] suggest normalizing dimension levels into separate tables with subclass-specific aggregation paths, resulting in the loss of their common hierarchy paths. Flattening of the hierarchy into a single table with optional attributes, as proposed in [19, 96], provides a homogeneous hierarchy scheme with no subclass-specific hierarchies. Some authors advocate decomposition of a fact table itself into subclass partitions resulting in a set of homogeneous stars [9, 53]. As a consequence, the entire set of facts can no longer be explored as one cube. A more flexible approach based on creating one table for common paths and separate tables for subclass-specific paths is proposed in [9]. In their recent work, Malinowski and Zimányi [109] propose to supply the normalized dimension tables of splitting levels with two kinds of roll-up references: *i*) to the subclass-specific parent level and *ii*) to the superclass' parent level.

The resulting hierarchy paths enable flexible traversal of heterogeneous trees: from the splitting level, the user can drill into a subclass tree or use the generalized link to remain in the common path.

Our approach to mapping generalized hierarchies is similar to that of [109]. However, it adds even more exploration options by handling the “mixed grain” phenomenon and overlapping subclass aggregation paths. The major challenge of building a dimension with heterogeneous instances and hierarchy paths is handled at the conceptual level. In Section 4.4.3 we defined various properties of generalized hierarchies, such as completeness, disjointness and raggedness. We also stated that generalized hierarchies of type mixed grain, incomplete and overlapping specialization lead to non-summarizable aggregation behavior. Therefore, these hierarchy types undergo scheme normalization to a complete disjoint specialization with no mixed grain.

NORMALIZING SPECIALIZATION

Incomplete specialization can be easily overcome by adding a superclass to cover those members that cause incompleteness, as described in Section 4.4.3 and exemplified in Figure 4.23b, in which category other staff was created as a superclass, additional to admin, staff and teaching staff. This normalization is similar to adding placeholder parent elements in a non-covering hierarchy, except that not just a new member but a new category has to be inserted to resolve edge skipping.

With overlapping specialization, various normalization strategies are conceivable. A rather simple strategy is to eliminate roll-up edges that cause overlap by “sacrificing” a subset of the original relationships. In many applications, however, such a trade-off is inadmissible for correct analysis. To meet the requirement of preserving the original relationships, we propose the following approach to resolving overlap:

1. Each overlap of multiple subtypes is extracted into a subclass of its own. Figure 7.8 demonstrates this process at the example of an overlapping specialization into categories *A*, *B*, *C*, *D*, and *E*: the hierarchy on the top is the original one (member values causing overlap are marked with red background color) and the bottom one is the transformed state after adding a new subclass for each set of overlapping classes (e.g., overlap of *B*, *C* and *D* is extracted into category *BCD*, and so on).
2. New categories are augmented with hierarchy paths inherited from all their contained subclasses.
3. Optionally, an additional abstraction level can be introduced to group new subclasses with the original ones according to the needs of the analysis. Each “composite” subclass is attached to one of its subclasses, e.g., *BCD* is attached to *B*. Alternative groupings are specified as alternative specialization hierarchies. Back to our example, let us suppose that two grouping options are feasible: *i*) subclasses containing *B* are merged with *B* into a superclass *B** and those containing *D* are merged with *D* into

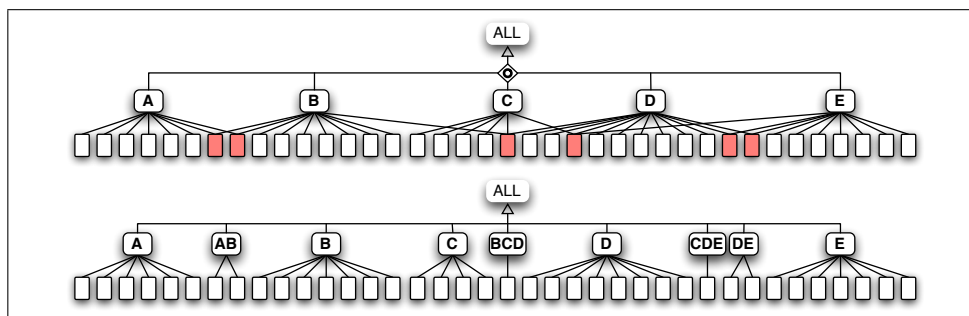


Figure 7.8: Transforming overlapping specialization (top) into a disjoint one (bottom)

D^* and ii) first, subclasses containing C are merged with C , then those containing E are merged with E and, finally, those containing B are merged with B . Figure 7.9 shows the resulting alternative specialization hierarchies, as (a) and (b), respectively (bottom level is omitted for compactness).

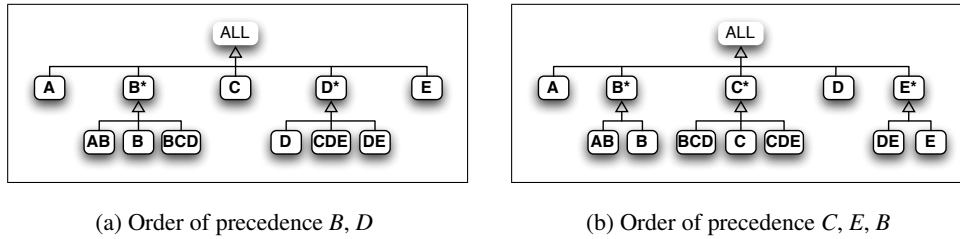


Figure 7.9: Adding alternative generalization layers for grouping disjoint subclasses

Back to the example of overlapping specialization of person into student and staff, presented in Chapter 4 (see Figure 4.24), let us suppose that students may actually function only as teaching staff and not as administrative one. Application of the above transformation steps would yield a new subclass student teaching staff, inheriting from student, staff, and teaching staff. Let us further suppose that associating this category with student is more relevant for the analysis than that with teaching staff. Figure 7.10 shows the transformed dimension scheme of person, in which the overlap of student and teaching staff is extracted into a specialization of student named student staff, the remainder of student members form the counterpart specialization student non-staff. Notice, that inheritance of student staff from teaching staff is implied by adding aggregation hierarchies of staff and teaching staff as multiple parallel hierarchies to student non-staff.

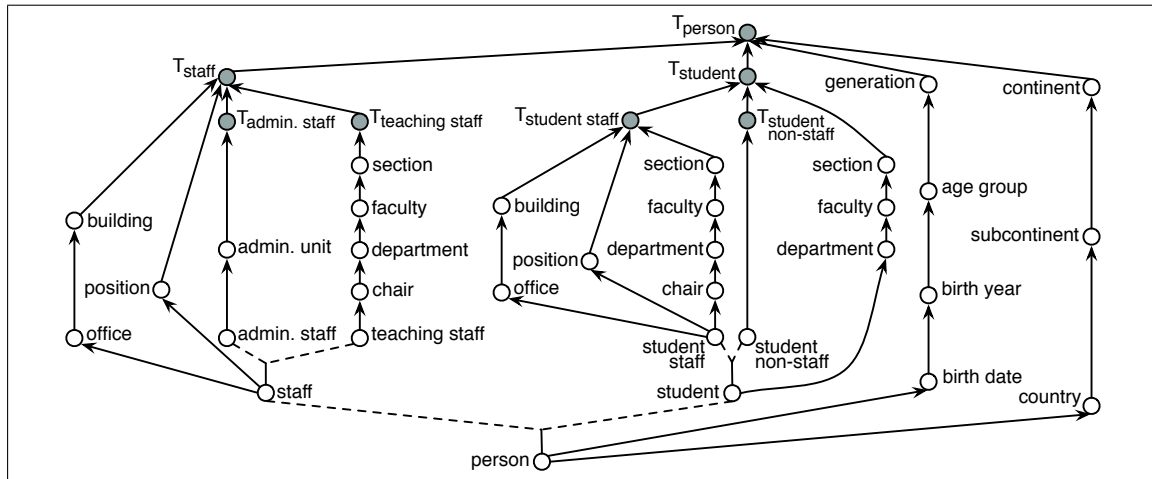


Figure 7.10: Generalized hierarchy person after eliminating overlapping specialization

NORMALIZING MIXED GRAIN

We propose an approach to transforming mixed hierarchies into summarizable generalization hierarchies by adding missing abstraction levels. The approach evolves largely in the same steps as the one for generalized hierarchy schemes described in Section 4.4.3. However, the awareness of ragged categories is “injected” into the scheme in form of self-specialization in the very first step. *Self-specialization* is a role-driven phenomenon of specializing a category according to its twofold role (i.e., bottom and non-bottom grain) into two subtypes of the respective roles. The original mixed-grain category turns into a generalization of the two subtypes providing correct summarization. Resolution of mixed grain evolves in the following steps:

1. In the plain inheritance hierarchy, each mixed-grain category is presented as a generalization of its two roles, e.g., faculty is specialized into faculty-self and faculty*, with faculty-self denoting faculties as purchaser units per se and faculty* as a hierarchy level for grouping department units. Figure 7.11a shows the inheritance tree of teaching unit with self-specializations of department and faculty.
2. The inheritance hierarchy is turned into a dimension scheme by adding abstract local root nodes on top of each specialization node and replacing shared-target style specialization edges by a set of distinct edges. The resulting hierarchy scheme is depicted in Figure 7.11b.
3. The scheme is augmented with roll-up relationships between categories. In the first place, hierarchical relationships between the bottom categories are added, following the rules of the unified multidimensional space, i.e., avoiding redundancy, with the result shown in Figure 7.12. Thereby, chair rolls-up to department*, which is a purely hierarchical subtype of department.
4. The remaining two steps, namely, obtaining a unified bottom level and pushing down specializations, are performed by integrating the scheme obtained in the previous step into purchaser dimension (see Figure 4.19). These steps are executed exactly as described for the case of generalized hierarchies. The final state of the entire dimension scheme purchaser is given in Figure 7.13.

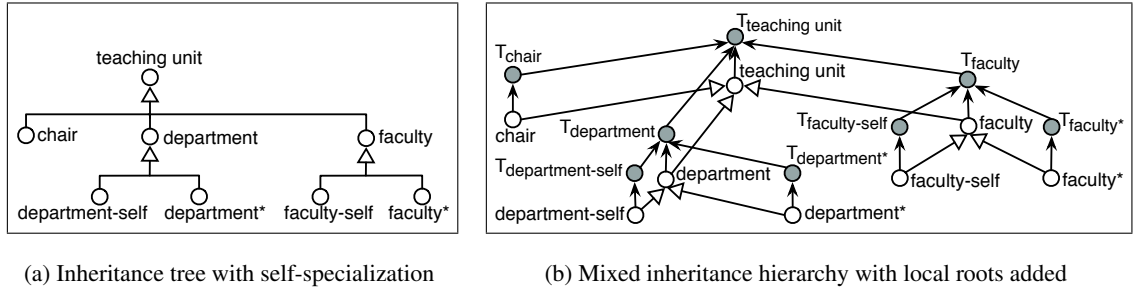


Figure 7.11: Resolving mixed grain into self-specializations

THE OVERALL MAPPING PROCEDURE

The conceptual scheme is converted into a relational one by creating a separate dimension table for each distinct aggregation level including superclass and subclass levels and excluding abstract subclass root categories. Generalized levels are used to unify members of different categories into one category, raising the problem of global primary key uniqueness. Let us consider the example of a highly heterogeneous scheme

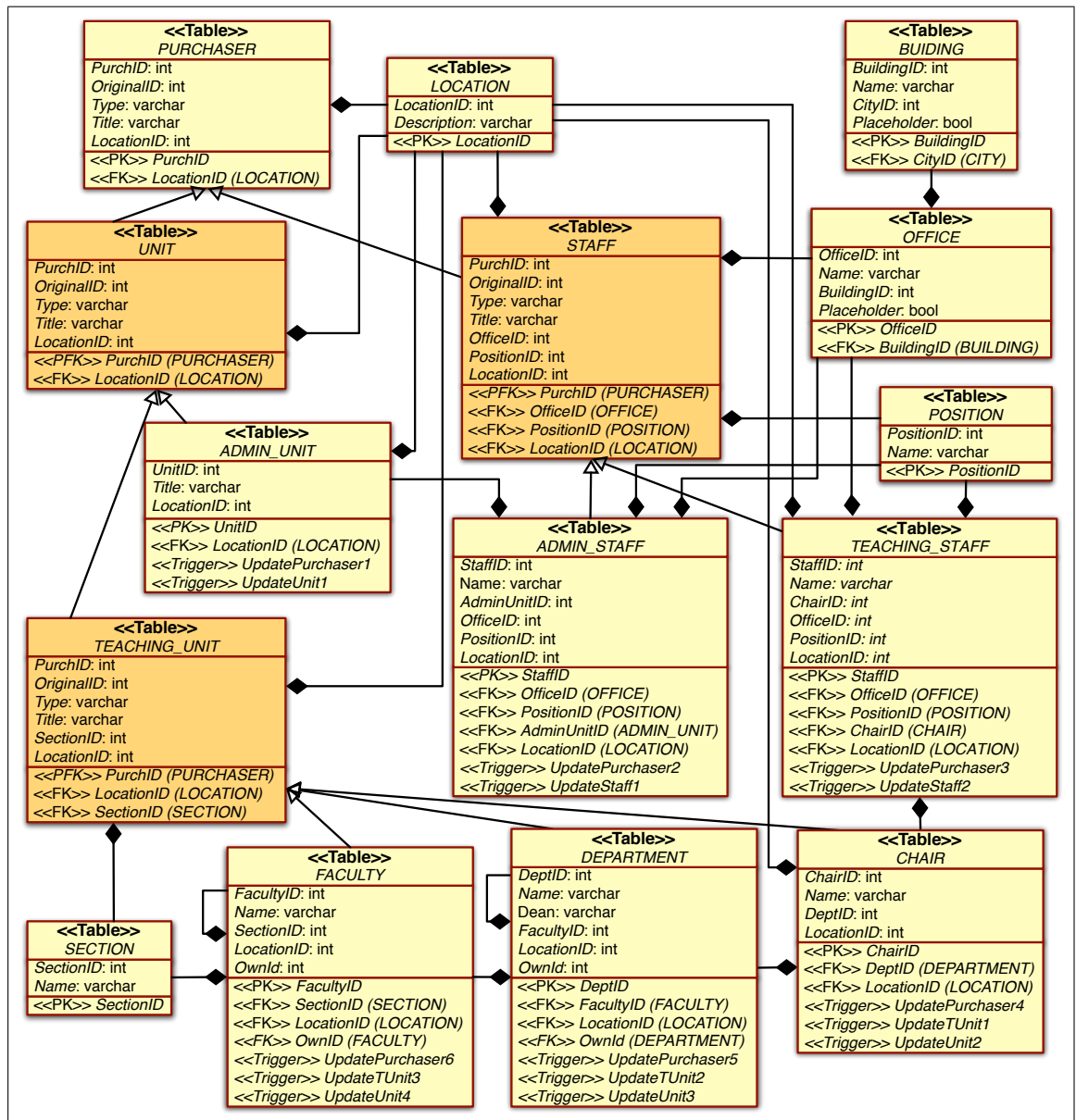


Figure 7.14: Logical schema of a generalized hierarchy purchaser

and its source table, respectively. Figure 7.15 exemplifies the proposed key allocation approach by showing sample instances of DEPARTMENT and FACULTY tables and their representation in PURCHASER.

New key values can be generated using auto-increment or some other function. The generalized relation is instantiated via a query that builds a union over all specialization tables at the bottom of the inheritance tree. PURCHASER instance is created as a union of 5 specializations with the following SQL statement:

```

INSERT INTO PURCHASER(OriginalID, Type, Title, LocationID)
      SELECT UnitID, 'ADMIN_UNIT', Title, LocationID FROM ADMIN_UNIT
UNION SELECT StaffID, 'ADMIN_STAFF', Name, LocationID FROM ADMIN_STAFF
UNION SELECT FacultyID, 'FACULTY', Name, LocationID FROM FACULTY
UNION SELECT DeptID, 'DEPARTMENT', Name, LocationID FROM DEPARTMENT
UNION SELECT ChairID, 'CHAIR', Name, LocationID FROM CHAIR

```

Due to the pursued key propagation strategy, non-transitive specialization relations are created first, followed by the most generalized superclass. Thereupon, transitive specialization classes are defined as derived tables (see relations UNIT, STAFF and TEACHING_UNIT in Figure 7.14). Once the schema of the entire dimension is completed, trigger constraints are added to each terminal specialization table to guard the consistency of the affected generalized level(s). For instance, FACULTY provides a trigger for each of its generalizations: UpdatePurchaser6 propagates modifications in FACULTY instance to the utmost generalization table PURCHASER, while UpdateUnit3 and UpdateUnit4 are responsible for keeping the copies of FACULTY members in the derived generalization tables TEACHING_UNIT and UNIT, respectively, up-to-date.

Let us now consider how the logical schema handles roll-up relationships in a generalization. Such relationships are propagated in the inheritance hierarchy bottom-up, i.e., starting from the terminal specialization categories and proceeding towards the next-level generalized category, until the utmost generalization is reached. Those parent-level references that are present in each specialization category, are replicated to the generalized one. For instance, OFFICE and POSITION references in TEACHING_STAFF and ADMIN_STAFF reoccur in STAFF, whereas LOCATION reference, common for all terminal specializations, reoccurs at each generalization level. In addition to the above propagation of parent references, generalized categories should account for transitive roll-up relationships common for all specializations. In our example, the hierarchy schemes of TEACHING_UNIT specializations CHAIR, DEPARTMENT, and FACULTY converge at SECTION. As a result, a roll-up to SECTION should be added to TEACHING_UNIT.

Finally, the issue of mapping mixed-grain generalization remains to be considered. Mixed grain occurs in DEPARTMENT and FACULTY as their member values play a double role by acting as non-hierarchical PURCHASER specializations, on the one hand, and as aggregation levels for CHAIR and DEPARTMENT, respectively, on the other hand. In the conceptual scheme, we proposed to resolve this duality by making

DEPARTMENT					
DeptID	Name	Dean	FacultyID	LocationID	OwnID
1	Computer Science	Peter Jung	1	511	1
2	Information Science	Hans Stolz	1	514	2

FACULTY				
FacultyID	Name	SectionID	LocationID	OwnID
1	Computer and Information Science	1	510	1
2	Mathematics and Statistics	1	232	2

PURCHASER				
PurchID	OriginalID	Type	Title	LocationID
1	1	FACULTY	Computer and Information Science	510
2	2	FACULTY	Mathematics and Statistics	232
3	1	DEPARTMENT	Computer Science	511
4	2	DEPARTMENT	Information Science	514

Figure 7.15: Example of obtaining the instance of the generalized relation PURCHASER

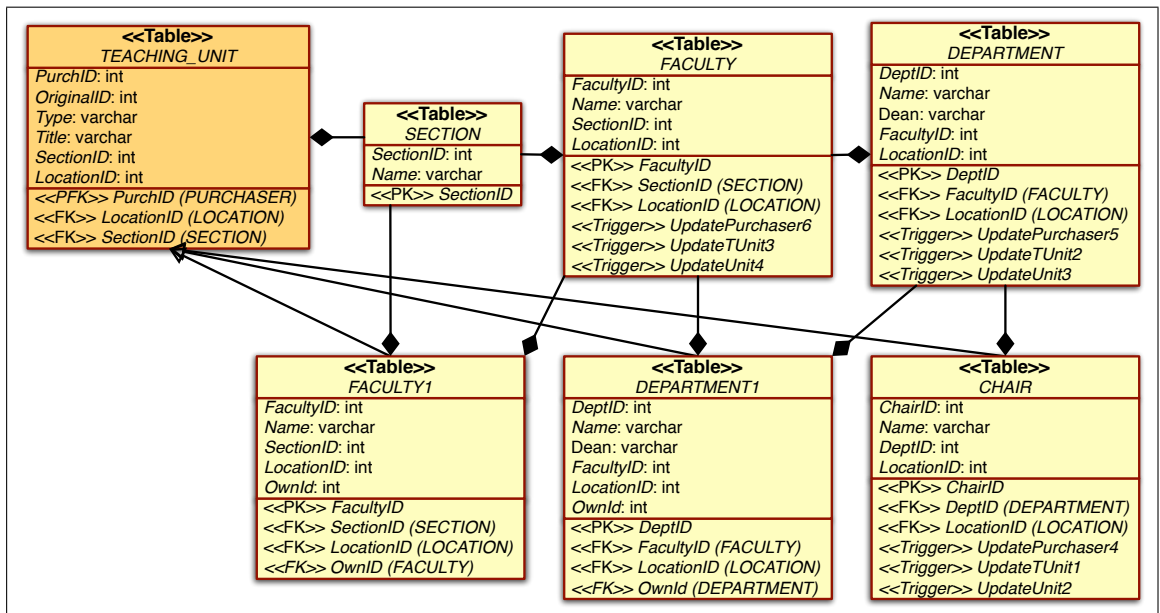


Figure 7.16: Resolving mixed grain into multiple dimension tables

a non-hierarchical “version” of the ragged category to roll up to its hierarchical counterpart. In the relational schema, this kind of recursive roll-up relationship can be conveniently expressed using a self-reference attribute `OwnID`. Figure 7.16 exemplifies the idea behind the self-reference by showing a variant of the same scheme fragment, in which mixed dimension levels are resolved each into two tables, making the double role evident: `DEPARTMENT1` and `FACULTY1` represent non-hierarchical `PURCHASER` specializations, whereas `DEPARTMENT` and `FACULTY` are hierarchy levels. As expected, `DEPARTMENT1` as a purchaser rolls up to `DEPARTMENT` as a hierarchy level. Obviously, the above resolution of a category into a pair of nearly identical tables produces unnecessary redundancy in the physical storage. Therefore, we suggest to merge the two relations into one and capture the different views and roles of the respective category through metadata. Figure 7.17 shows the state of the relational scheme from Figure 7.16 after incorporating `DEPARTMENT1` and `FACULTY1` tables into `DEPARTMENT` and `FACULTY`, respectively.

The approach to implementing heterogeneous hierarchies, described in this section, is based on the materializing generalized categories and replicating parent-level references upwards along the entire generalization hierarchy. Even though at the expense of redundant storage, it achieves the preservation of all valid aggregation paths and their combinations, as implied by the underlying conceptual scheme:

- ◆ The logical representation of any hierarchy level including generalized ones is complete as it encompasses all valid attributes and roll-up relationships. Therefore, any level can be reused outside of its original dimensional context. For instance, the generalized category `STAFF` and its specializations in `PURCHASER` dimension are ready to be used as `staff` dimension in any fact scheme.
- ◆ All valid aggregation paths of a generalized category are available directly, i.e., without accessing its specializations.
- ◆ Hierarchies elevated to the generalized level remain available at the subclass level as hierarchies, parallel to the subclass-specific ones.

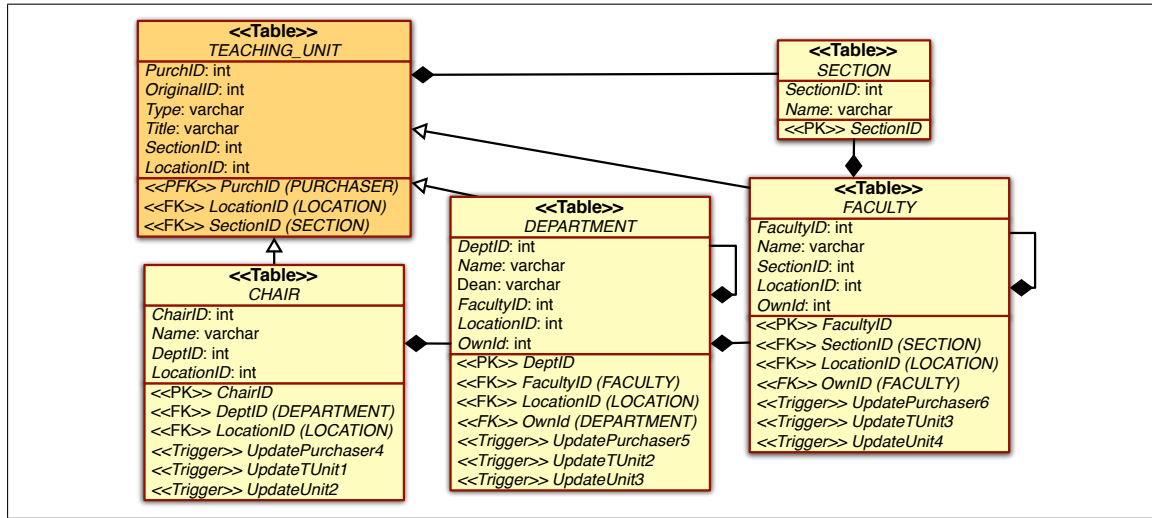


Figure 7.17: Optimized logical schema of dimension tables with mixed grain

In this section, we focused on the challenges of handling heterogeneity in dimension hierarchies and presented a relational mapping that enables adequate storage of such hierarchies and lays the foundation for correct aggregation behavior. The relational schema alone, however, is insufficient for recognizing all valid aggregation paths. This task is carried out by the metadata, which specifies the actual structure of facts and dimensions. Further insights on metadata are provided in Section 7.4.

7.3 Enforcing Summarizability in Homogeneous Hierarchies

An important requirement towards the logical schema is that it must fulfill the summarizability constraint. Non-summarizable hierarchy types have been classified by our dimensional metamodel, depicted in Figure 4.5. In the previous section, we presented scheme normalization techniques for enforcing summarizable behavior in heterogeneous hierarchies of types *non-covering*, *incomplete*, *overlapping*, and *mixed-grain*. Therefore, it remains to handle *non-strict* and *non-onto* mappings in homogeneous hierarchy schemes.

Various transformation techniques for enforcing summarizability have been proposed in the literature [19, 109, 145]. In general, the choice of a particular technique depends largely on the semantics behind the data and the requirements of the analysis. If irregularity is caused by missing or imprecisely captured values and it is crucial to produce imprecision-aware queries and results (e.g., in clinical diagnosing or risk assessment), the approach of Pedersen et al. [145], in which the original data remains de-normalized and imprecision is made explicit to the user by providing a set of alternative queries, may provide an adequate solution. However, if the data hierarchy is intrinsically asymmetric, such as the hierarchical structure of an organization, it needs to be normalized to ensure correct aggregation using conventional OLAP operators. Our choice of instance normalization techniques is driven by the considerations of minimality, transparency, and intuitiveness for the user.

Recall that a hierarchy is summarizable, if it is *onto*, *covering* and *strict*. Hierarchies in the real world may violate one or more of those properties. If more than one property is violated at the same hierarchy level, it is imperative to determine the precedence in which the summarizability conditions are enforced. Pedersen et al.

[145] propose to normalize irregular hierarchies by mapping to covering, followed by mapping to onto and, finally, to strict. By investigating various data sets, we figured out, that the order of the transformation steps depends on the kinds of techniques invoked at each step. For instance, mapping to strict by eliminating multi-parent edges may produce a non-onto hierarchy. In such a case, it appears feasible to handle non-strictness prior to non-onto in order to avoid cyclic normalization.

The actual transformation of the hierarchy instance is preceded by exact localization of each summarizability violation pattern in the hierarchy scheme. As an example of a hierarchy that contains fragments of types non-onto, non-strict, and non-covering, let us consider a sample organization hierarchy of university courses, depicted in Figure 7.18, with the hierarchy scheme shown to the right of the instance. Courses are offered by departments, departments belong to faculties and the latter are grouped into sections.

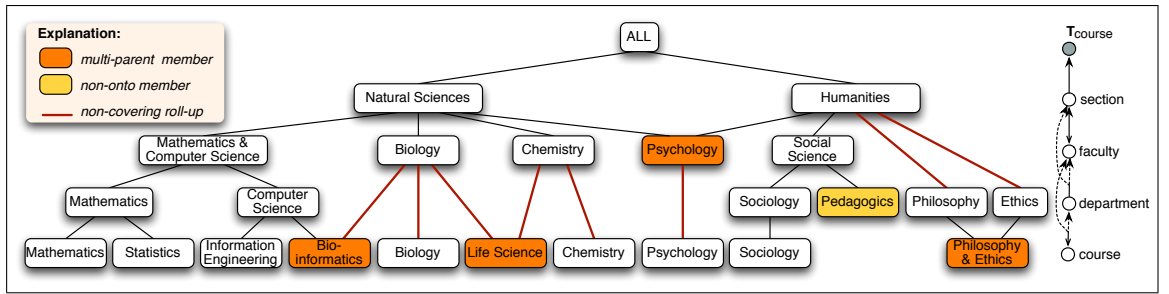


Figure 7.18: A sample hierarchy containing non-onto, non-covering and non-strict elements

However, there exist faculties with no subdivision into departments, which also offer courses (e.g., *Biology*) and there exist departments not assigned to any faculty (e.g., *Philosophy*). These two kinds of deviating roll-up behavior result in non-covering mappings between course and department and between department and faculty (non-covering edges in the hierarchy instance are marked with dark-red color).

There exist departments with no courses (e.g., *Pedagogics*), resulting in a non-onto mapping between course and department (non-onto elements in the hierarchy instance have yellow background color).

Finally, there are courses offered jointly by multiple departments and/or faculties (e.g., *Bioinformatics*) as well as there are faculties assigned to multiple sections (e.g., *Psychology*), resulting in non-strict mappings between course and department/faculty and between faculty and section (multi-parent members are shown with orange background color). The results of localizing non-balanced hierarchy fragments can be summarized as follows:

- ◆ *Non-covering*: $\text{course} \sqsubseteq^{(\text{part})} (\text{department}|\text{faculty})$ and $\text{department} \sqsubseteq^{(\text{part})} (\text{faculty}|\text{section})$
- ◆ *Non-onto*: $\text{course} \sqsubseteq \text{department}$
- ◆ *Non-strict*: $\text{course} \sqsubseteq^{(\text{part})} (\text{department}|\text{faculty})$ and $\text{faculty} \sqsubseteq \text{section}$

7.3.1 Mapping to Covering

The first transformation step maps a hierarchy instance to covering. It is achieved by applying the combined scheme and instance normalization technique presented in Section 7.2.1. Back to the above example, the resulting covering hierarchy is depicted in Figure 7.19 (the inserted placeholder elements are shown with grey background).

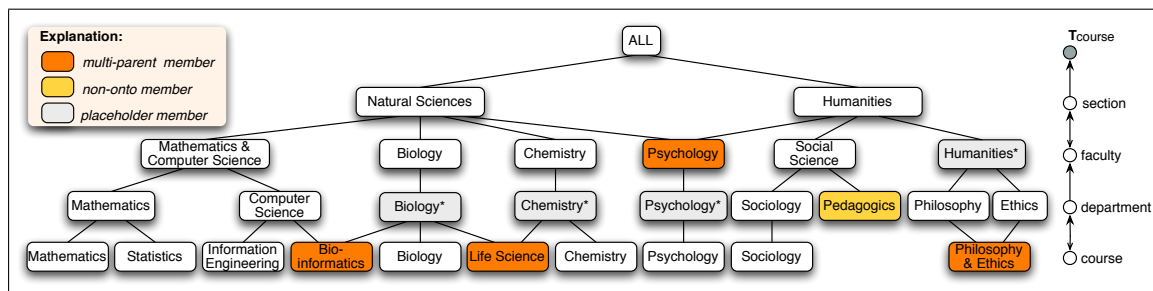


Figure 7.19: The state of the hierarchy after mapping to covering

7.3.2 Mapping to Strict

A data warehouse designer confronted with non-strict hierarchies has two major choices: *i*) to preserve non-strictness in the data and to provide mechanisms for correct aggregation in such hierarchies, and *ii*) to transform a non-strict hierarchy into a strict one.

If the analysis admits or even requires a hierarchy to be mapped to strict, the actual choice of the transformation technique depends on the semantics behind non-strict roll-up relationships. In what follows, we present two strategies for mapping to strict, namely, a manual one based on edge elimination and an algorithmic one based on fusing multiparent elements.

EDGE ELIMINATION

If the accuracy of many-to-many relationships is not crucial for the analysis, the hierarchy can be transformed into strict via a simple edge elimination: each set of multi-parent roll-up relationships is reduced to a single “priority” edge. Priority edges can be specified manually based on the use preferences, or picked randomly if no such preferences are available. Only the resulting strict hierarchy is implemented. Figure 7.20 shows a sample strict mapping of the sample course hierarchy, obtained via edge elimination. Note that this technique may have a side effect of leaving some of the affected parent elements non-onto (childless), as is the case with *Ethics Department* in our example.

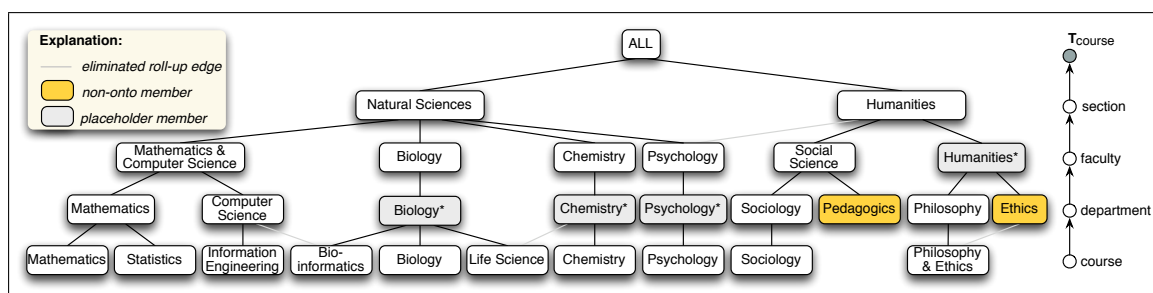


Figure 7.20: The state of the hierarchy after eliminating multi-parent roll-ups

“FUSED” MULTI-PARENT ELEMENTS

Pedersen et al. [144] propose a solution based on turning a non-strict hierarchy into a strict one based on “fusing” each set of multiple parent elements into one “fused” value. The hierarchy is normalized bottom-up. First, explicit and implied non-strict roll-ups between hierarchy levels are to be identified. In our sample hierarchy, depicted in Figure 7.19, overlapping subtrees exist at department, faculty, and section level. New parent elements are inserted as a new category between the original parent and child categories of a non-strict roll-up and is named set-of ‘p’, where ‘p’ is the name of the parent category. For example, category set-of department is inserted in-between course and department, resulting in a strict roll-up relationship between course and set-of department. Elements of all new categories at different levels are linked to one another to produce a strict hierarchy. Figure 7.21 shows the state of the hierarchy after inserting three additional levels. Since the original algorithm in [144] prohibits non-onto nodes in the input hierarchy, we suggest treating them just like onto nodes within the same category at this stage. The resulting scheme $\text{course} \sqsubseteq \text{set-of department} \sqsubseteq \text{set-of faculty} \sqsubseteq \text{set-of section} \sqsubseteq T_{\text{course}}$ is strict.

Note that the elements of the new category are also linked to the relevant values of the original parent category (e.g., set-of department rolls up to department) resulting in a non-strict mapping between the two, as reflected in the hierarchy scheme shown on the right of its instance in Figure 7.21. The authors propose to disable non-summarizable aggregation along such paths by “unlinking” the original parent category from its upward roll-up path. Thereby, upper-level aggregates can be reached only through the hierarchy of fused categories and not the original one. For our example, it means detaching department from set-of faculty, faculty from set-of section and section from T_{course} . In Figure 7.21, the edges affected by this elimination are marked with a red cross in both the instance and the scheme.

Unlinking elements from their upward hierarchy paths results in the existence of subtrees not reaching the root node. In [144], unlinked categories are denoted “unsafe” and are exempted from the aggregation. However, the algorithm preserves unlinked subtrees in the hierarchy instance along with the corresponding scheme fragments, detached from the root category. In our model, such fragments are inadmissible by definition (each member in a dimension finally rolls up to the root) and, therefore, have to be entirely removed from the final state of the hierarchy. Figure 7.22 shows the resulting strict hierarchy after the removal of the “unsafe” categories. Note that we also marked the childless fused element *Pedagogics* as onto.

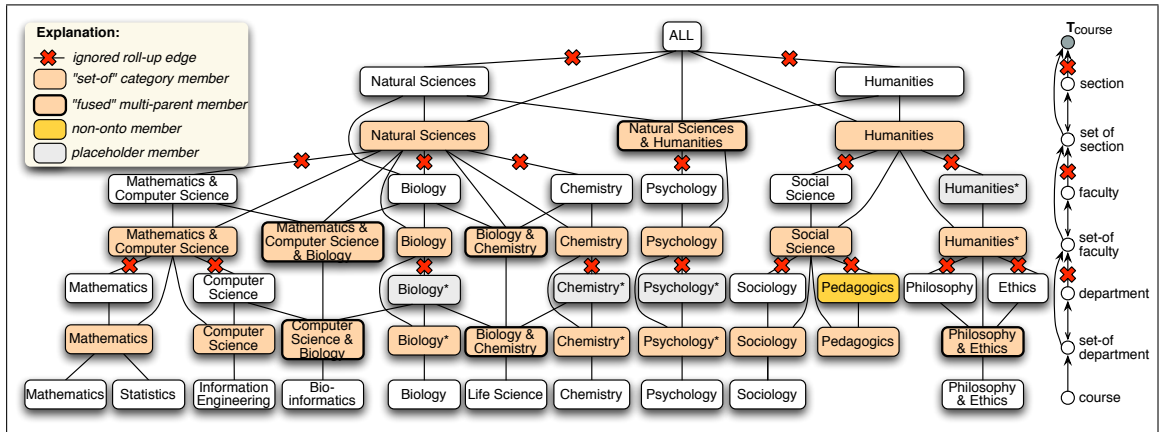


Figure 7.21: The state of the hierarchy after adding categories with “fused” elements

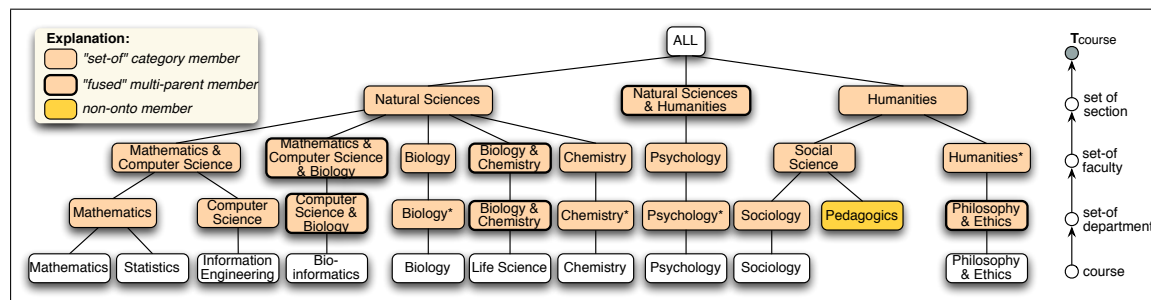


Figure 7.22: The state of the sample hierarchy after unlinking non-strict roll-up relationships

BRIDGE TABLES

Another group of techniques aims at storing a non-strict mapping “as it is” or with additional hints for correct aggregate computation. A currently popular data warehouse implementation of a many-to-many relationship between a pair of categories is known as *bridge table* [87]. A separate bridge table is created for each non-strict roll-up relationship in the hierarchy to compensate for the missing parent level reference in the child category’s dimension table. In addition to the child-parent pair itself, a bridge table must include information about measure distribution, i.e., how a lower-level aggregate is to be split between its multiple upper-level parent aggregates. This information can be made available in one of the following representations:

1. **“Weighted” non-strictness.** In the conceptual model, we introduced a summarizable hierarchy of type *weighted non-strict* (see Section 4.4.1): whenever an element rolls up to multiple parent elements, each of these roll-ups is assigned a degree of belonging (weight) valued between 0 and 1. The sum of one child node’s degrees of belonging should be exactly 1.0 to ensure summarizable aggregation. The sample covering hierarchy from Step 1, augmented with edge weights for resolving non-strict roll-up relationships at department and section levels is depicted in Figure 7.23.

As for the implementation of the resulting hierarchy, the dimension tables *COURSE* and *FACULTY* have no parent level reference. Instead, two bridge tables, *COURSE_DEPARTMENT* and *FACULTY_SECTION*, have to be created to capture the respective weighted non-strict roll-up relationships.

In the process of aggregate computation, consistent results are ensured by multiplying the measure, aggregated along a weighted non-strict hierarchy, with each enclosed element’s degree of belonging.

2. **Ad hoc edge elimination.** If multi-parent relationships are relatively infrequent or if a hierarchy is used in different contexts, “priority” parent values can be specified ad hoc. Whenever a roll-up operation encounters a multi-parent relationship, the user is prompted to specify interactively, to which of the parent aggregates the child aggregate should be assigned. The advantage of this strategy compared to simple edge elimination is leaving the resolution up to the user, thus supporting various user preferences. Figure 7.24 shows a non-strict hierarchy and its ad hoc strict variant, obtained by marking the edges to be excluded when performing a roll-up. Edge elimination can be seen as a degeneration of weighted non-strictness, in which one of the parent elements is assigned the weight of 1 and all the others are set to 0. Therefore, this technique can be implemented similarly, i.e., using bridge tables with a data field for storing the degree of belonging. Ad hoc specification of a user-defined strict hierarchy variant is done on that user’s copy of the bridge table.

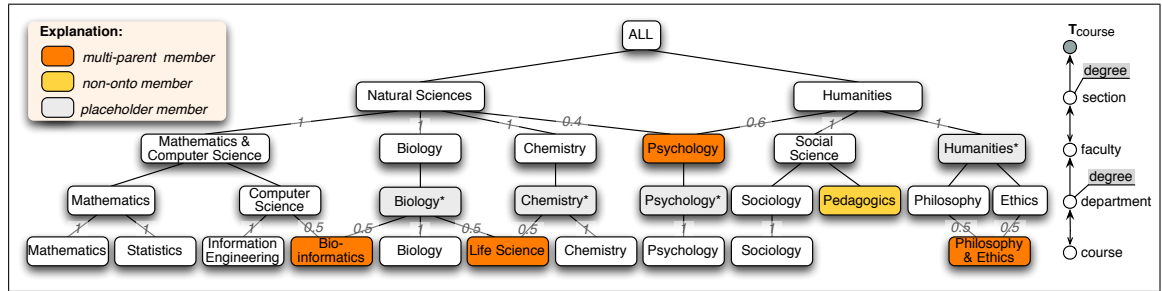


Figure 7.23: Resolving multi-parent relationships by assigning weights to roll-up edges

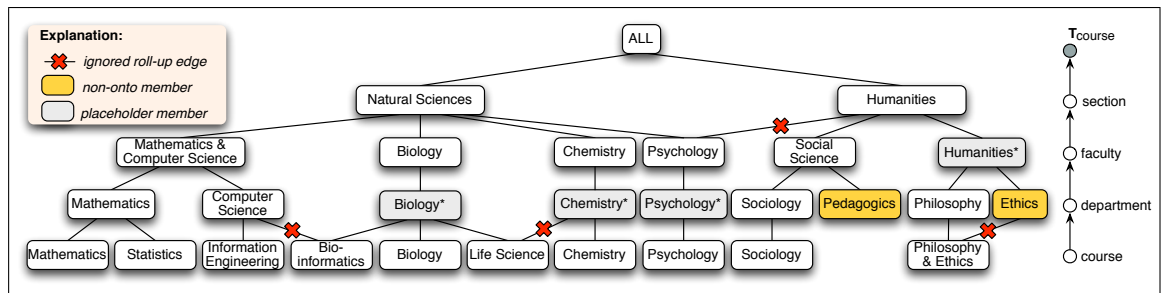


Figure 7.24: The state of the hierarchy after ad hoc edge elimination

7.3.3 Mapping to Onto

Mapping to *onto* is performed as the final step since a symmetric hierarchy may become non-onto as a result of eliminating parent edges for resolving non-strictness, as in the case depicted in Figure 7.24. Interestingly, this last step may be omitted altogether in those cases, where childless members in the hierarchy are guaranteed to have no associated fact entries. For example, the member *Pedagogics Department*, which offers no courses, is simply irrelevant for the course hierarchy.

Mapping to onto is crucial for inherently asymmetric hierarchies, in which leaf nodes actually belong to different hierarchy levels, resulting in mixed granularity. As an example of such asymmetry, let us consider the structure of educational units themselves, i.e., without the offered courses, from Figure 7.18. The original state of this hierarchy shown in Figure 7.25. This hierarchy is non-onto since its bottom granularity is composed of two levels, namely, department and faculty. Let us further assume that there exists a fact scheme, which includes the above hierarchy as one of its dimensions. Linking the fact scheme to the dimension's bottom level department will result in the impossibility to capture those fact entries referring to faculties with no departments. Obviously, mapping to onto is indispensable in such a scenario.

A traditional technique for mapping to onto is the same as the one used for mapping to covering: asymmetric hierarchy paths are normalized by inserting placeholder child elements [109, 144]. Back to our example, the irregular hierarchy is balanced by creating “dummy” department nodes for the faculties *Biology*, *Chemistry*, and *Psychology*. Note that the hierarchy had to undergo normalization to covering (placeholder nodes at faculty level) and to strict (“fusing” two section values) prior to its mapping to onto. The obtained balanced hierarchy instance is shown in Figure 7.26.

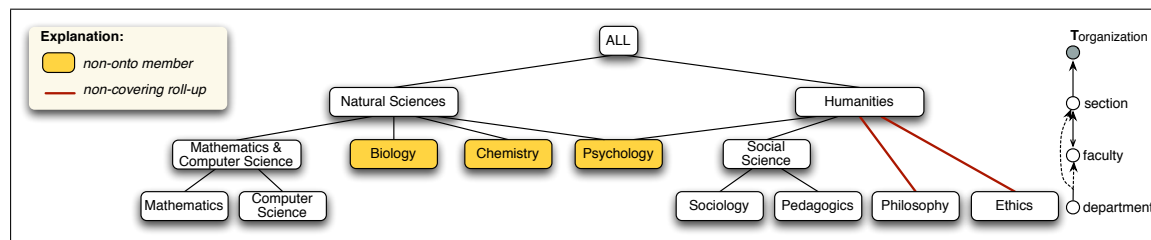


Figure 7.25: An inherently asymmetric department hierarchy

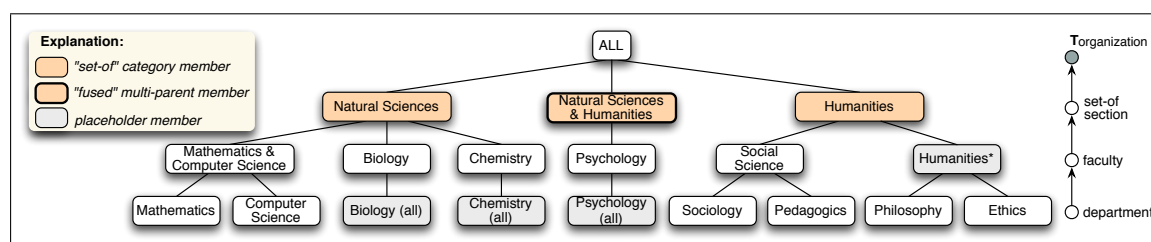


Figure 7.26: The state of the department hierarchy after mapping to covering, strict, and onto

7.4 Metadata for the Analysis Layer

Generally, metadata refers to an abstraction of lower-level data into knowledge, or “data about data”. In the world of databases, metadata is used to provide a concise description of the actual data contents, necessary for its management. In data warehouse systems, metadata turns into a core ingredient as it provides the foundation for all interactions between the architectural layers and components. Further, it defines an interface for data interchange and communication between the data warehouse and relevant external sources and tools.

In this section, we focus on a subclass of metadata concerned with bridging the gap between the multidimensional data model and its relational implementation. Its goal is to map relational structures (tables, views, columns, constraints, etc.) to the constructs of the conceptual model (facts, measures, dimensions, hierarchies, levels, properties, etc.). Thereby, in the frontend layer, the data is represented in a way adequate for the analysis with implementation details hidden from the user.

There exist different classifications of metadata in data warehouse systems [8, 62, 86]. Bauer and Günzel [8] summarize the existing classifications according to five criteria:

1. According to the *data type*, metadata refers to primary data (schemata, relations, attributes), processes (ETL), or enterprise organization (users, access rights).
2. The *level of abstraction* suggests distinguishing between conceptual (application relevant definitions), logical (relational schemata) and physical (SQL code) metadata.
3. The *application perspective* divides metadata into business (terminology, reports, documentation) and technical (schema definitions, transformation rules) metadata.
4. The *origin* of metadata is determined by the tool or the source, by which it was supplied.
5. According to the *creation/usage time*, the metadata refers to the design (schema definitions, transformation rules), setup (log files) or usage (access statistics and patterns).

In addition to the above categorization, metadata can be classified into ETL, data management, and query management metadata based on the reference architecture layers [33]. Huynh et al. [62] also distinguish between static (data structure) and dynamic (access statistics) metadata.

A key to successful interplay of heterogeneous sources and components involved in the operation of a data warehousing system is their adherence to a common standard in part of metadata management. Ideally, a unified metamodel is necessary for enabling unrestrained data integration and interchange across vendor-specific platforms. As a response to this need, two major standardization efforts arose: *i*) the Common Warehouse Metamodel (CWM) [135] by the Object Management Group (OMG) and *ii*) the Open Information Model (OIM) [121] by the Meta Data Coalition (MDC). Both metamodels are UML-based and use XML as a format for data interchange. In 2000, the MDC merged with the Object Management Group (OMG) in favor of the CWM as a prospective single standard for metadata modeling in data warehouse systems.

7.4.1 Overview of the CWM

The initial version of the CWM specification was issued in 1999; the current version is 1.1, dated March 2003 [135]. CWM provides an open industry standard for metadata-based integration of BI tools and components in a product and vendor independent fashion across all major hardware platforms and operating systems.

CWM builds upon three key industry standards provided by OMG:

- ◆ *UML* [136] as a modeling standard,
- ◆ *MOF* (Meta Object Facility) as a metamodeling and metadata repository standard,
- ◆ *XMI* (XML Metadata Interchange) as a metadata interchange standard.

These three standards form the core of the OMG metadata repository architecture and provide a foundation for adequate representation of the data warehousing semantics. CWM acknowledges the heterogeneity of metadata requirements in a data warehouse system by structuring its provided metamodel into a set of sub-metamodels, as depicted in Figure 7.27, for representing metadata in the following major areas of interest:

1. *Warehouse Management* includes metamodels that represent warehouse processes and operations.
2. *Data Analysis* includes metamodels that represent data transformations, OLAP, data mining, information visualization, and business nomenclature.
3. *Data Resources* include metamodels that represent object-oriented, relational, record, multidimensional, and XML data resources.

<i>Management</i>	Warehouse Process			Warehouse Operation		
<i>Analysis</i>	Transformation		OLAP	Data Mining	Information Visualization	Business Nomenclature
<i>Resource</i>	Object Model	Relational	Record	Multidimensional		XML
<i>Foundation</i>	Business Information	Data Types	Expressions	Keys and Indexes	Type Mapping	Software Deployment
<i>Object Model</i>	Core		Behavioral	Relationships		Instance

Figure 7.27: The multi-layered metamodel of the CWM and its constituent packages

Foundation layer provides packages for various concepts and structures (business information, data types, expressions, key and indexes, software deployment, and type mapping) to be used by the packages of the above enumerated higher layers. The base of the entire multilayer model is the *Object Model*, which supplies packages defining the fundamental metamodel services for all upper layers.

The CWM is organized into a modular architecture consisting of 21 separate packages (each rectangle in Figure 7.27 represents a package) grouped into five stackable layers, so that metamodels residing at one particular layer are dependent only on metamodels residing at the layer underneath it. As a result, vertical coupling between packages is reduced to a minimum and there is no horizontal coupling whatsoever.

OLAP PACKAGE

Description of the multidimensional properties behind a relational resource is handled by the Analysis layer, more precisely, by its OLAP package. In OLAP, the data is analyzed as multidimensional cubes consisting of measures distributed along a set of dimensions, with the latter typically structured into one or several classification hierarchies. Obviously, those are the elements of the conceptual data model. The OLAP package of the CWM provides a metamodel that describes those elements in terms of classes and relationships between them. Figure 7.28 shows the major classes and associations of the OLAP metamodel in the UML notation.

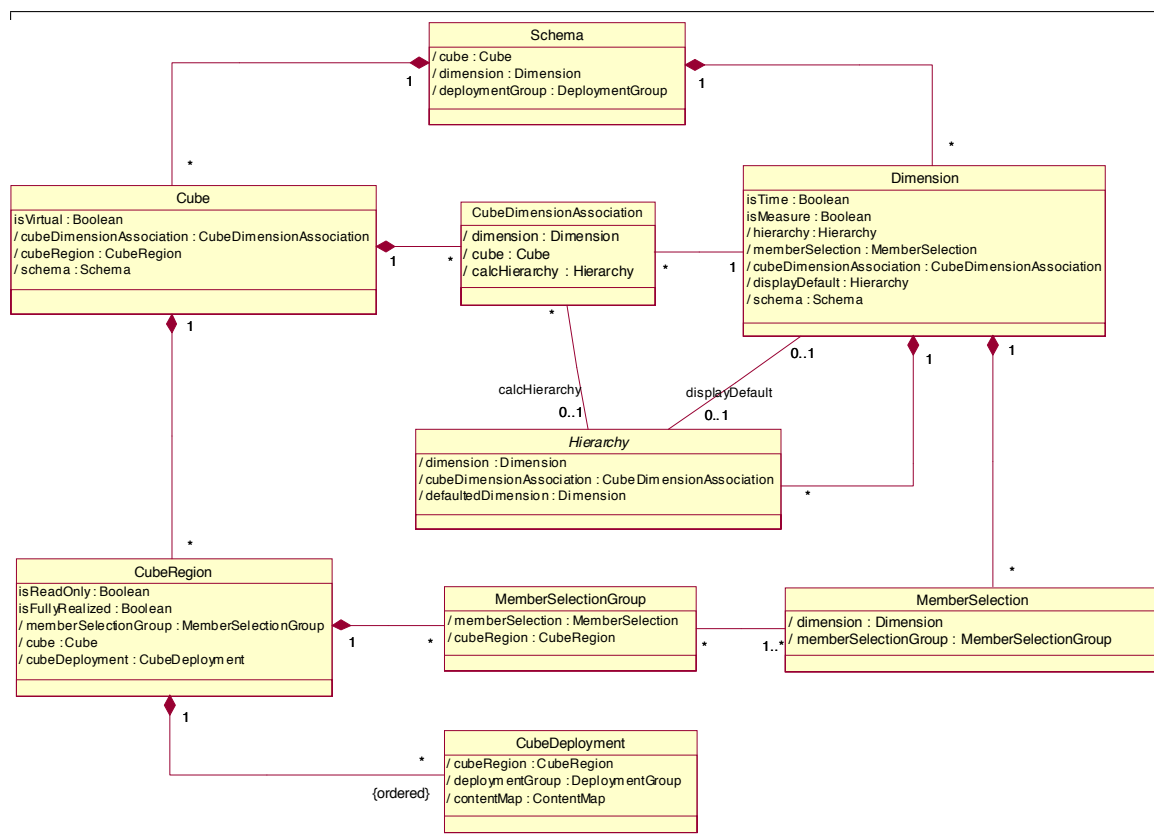


Figure 7.28: OLAP Metamodel of the CWM: Major Classes and Associations (adopted from [135])

The root element of the model as well as of the resulting data navigation hierarchy is Schema, which is a logical container of all other elements. A Schema is structured into elements of types Cube and Dimension. CubeDimensionAssociation relates a Cube to its defining Dimensions. A Dimension is a collection of unique values, optionally structured into one or several Hierarchies. A Hierarchy is defined in terms of parent/child relationships between members of a Dimension. It is possible to designate a default Hierarchy within a Dimension. MemberSelection is a mechanism for identifying partitions within the instance of a Dimension, such members belonging to a specific hierarchy level. CubeRegion defines a subcube of the same dimensionality as the Cube itself. A dimension in a CubeRegion is specified via a corresponding MemberSelection. CubeRegion enables implementation of a Cube as a set of regions, each mapping a portion of the logical cube to a physical data source. Multiple MemberSelections in a CubeRegion can be grouped into MemberSelectionGroup, enabling definition of CubeRegions with specific semantics. CubeDeployment represents an implementation strategy for a CubeRegion.

Figure 7.29 reveals further specification details referring to the metaclasses Dimension and Hierarchy. The metamodel defined two special Dimension types: time (isTime) and measure (isMeasure). Time dimension is used for representing temporal values thus laying the foundation for providing advanced “time-intelligent” functionality. Measure dimension describes the measure set of a multidimensional structure:

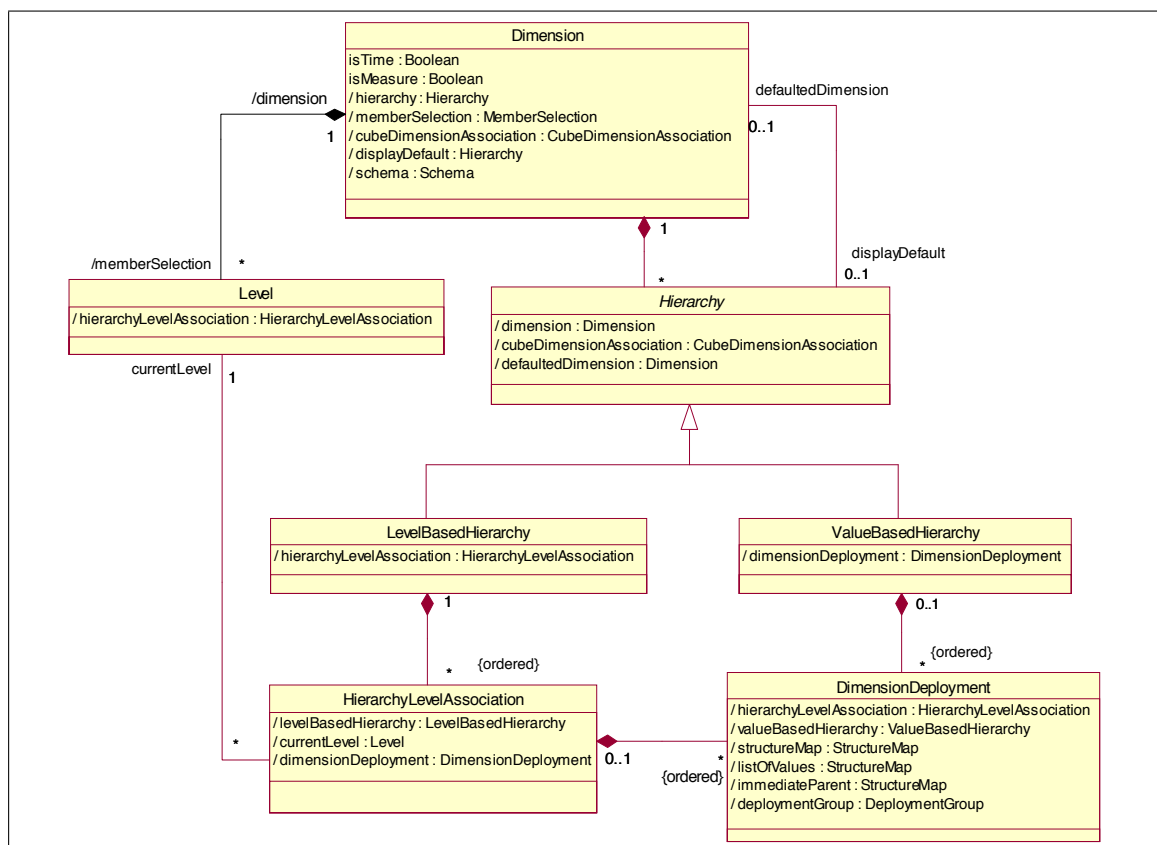


Figure 7.29: OLAP Metamodel of the CWM: Dimension and Hierarchy (adopted from [135])

whenever a Cube stores multiple measures, the measure dimension takes the names of the measure attributes as its member values. Thereby, the cells of a Cube remain single-valued, whereas values in the measure dimension specifies what measure attribute is stored in the respective cell.

There are two subclasses of Hierarchy: LevelBasedHierarchy and ValueBasedHierarchy.

LevelBasedHierarchy represents the classical definition of a dimension hierarchy, in which members are arranged into hierarchical Levels (dimension categories). Level is a subclass of MemberSelection that partitions the Dimension's members into disjoint level-wise subsets. LevelBasedHierarchy is an ordered collection of HierarchyLevelAssociations that defines the hierarchical structure in top-down fashion. A HierarchyLevelAssociation may own any number of DimensionDeployments, whereas the class DimensionDeployment represents an implementation strategy for hierarchically structured Dimensions.

A ValueBasedHierarchy defines a hierarchical ordering with no hierarchy scheme, i.e., no explicit levels. Such hierarchies are typically obtained by classifying or ranking member values according to their distance from the common root. Thereby, each member has some specific "metric" associated with it. A ValueBasedHierarchy is realized as an (ordered) collection of DimensionDeployments.

The classes of the OLAP metamodel inherit from the packages Core and Transformation of the CWM Object Model, as shown in Figure 7.30. The metaclass Measure is defined as a subclass of Attribute that describes the meaning of values stored in the data cells of a Cube. Depending on the adopted definition of the term "measure", measure attributes can be represented as attributes in a Cube or CubeRegion that models the fact table or as values characterized by the members of a Measure Dimension. In the latter representation, a fact table has a single Measure column with values corresponding to the names of the measure attributes (e.g., "sales", "amount") and a single value column of an implicit data dimension storing the actual measure values. Relational OLAP systems adopt the first measure representation option (i.e., as a attribute), whereas multidimensional servers often opt for the second one (i.e., as a dimension).

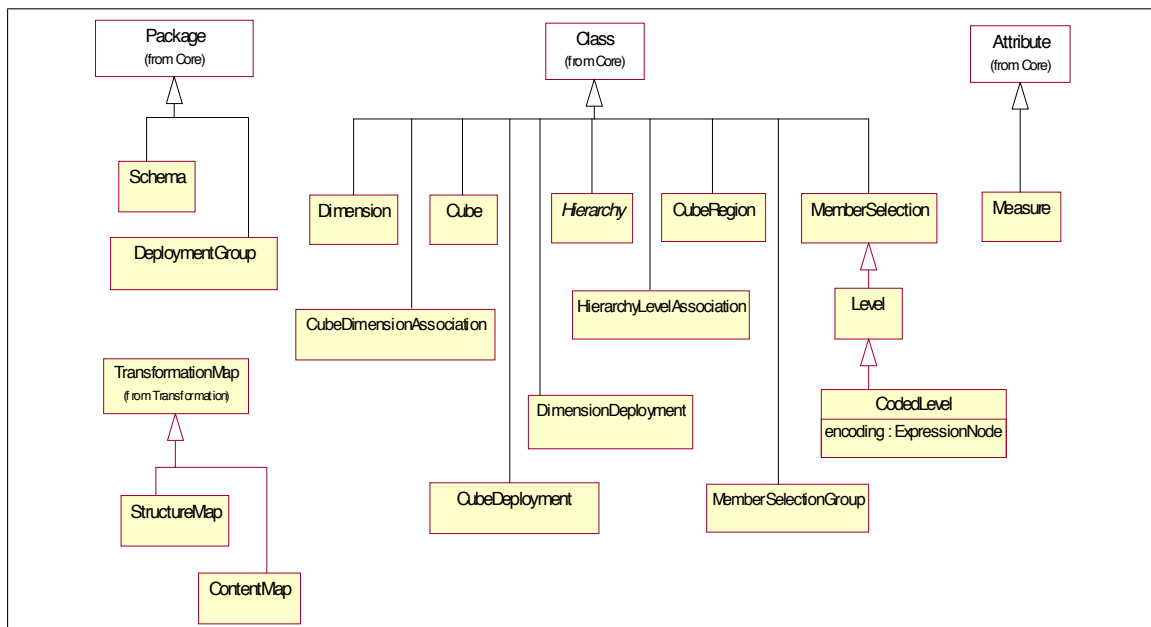


Figure 7.30: OLAP Metamodel of the CWM: Inheritance from the Object Model (adopted from [135])

RELATIONAL DEPLOYMENT OF OLAP MODELS

Note that the OLAP metamodel does not specify how an OLAP system should be implemented. The deployment is modeled by mapping the instances of the OLAP metaclasses to the metaclasses of the relevant package in the Resource layer. In case of a ROLAP implementation, the relevant package is Relational and the target elements are tables, columns, referential constraints, etc.

The CWM Transformation package is used as the primary means of mapping the logical OLAP model to the Resource layer. Figure 7.31 shows a fragment of the Transformation package relevant for OLAP deployment: its major metaclass `TransformationMap` and its subclass hierarchy are used to specify so called “white-box” transformations, which relate data sources and targets to a transformation and to each other at a detailed level. “White-box” transformations are commonly used in data warehousing where it is necessary to specify exactly how a specific data source is related to a specific data target through a specific part of the transformation.

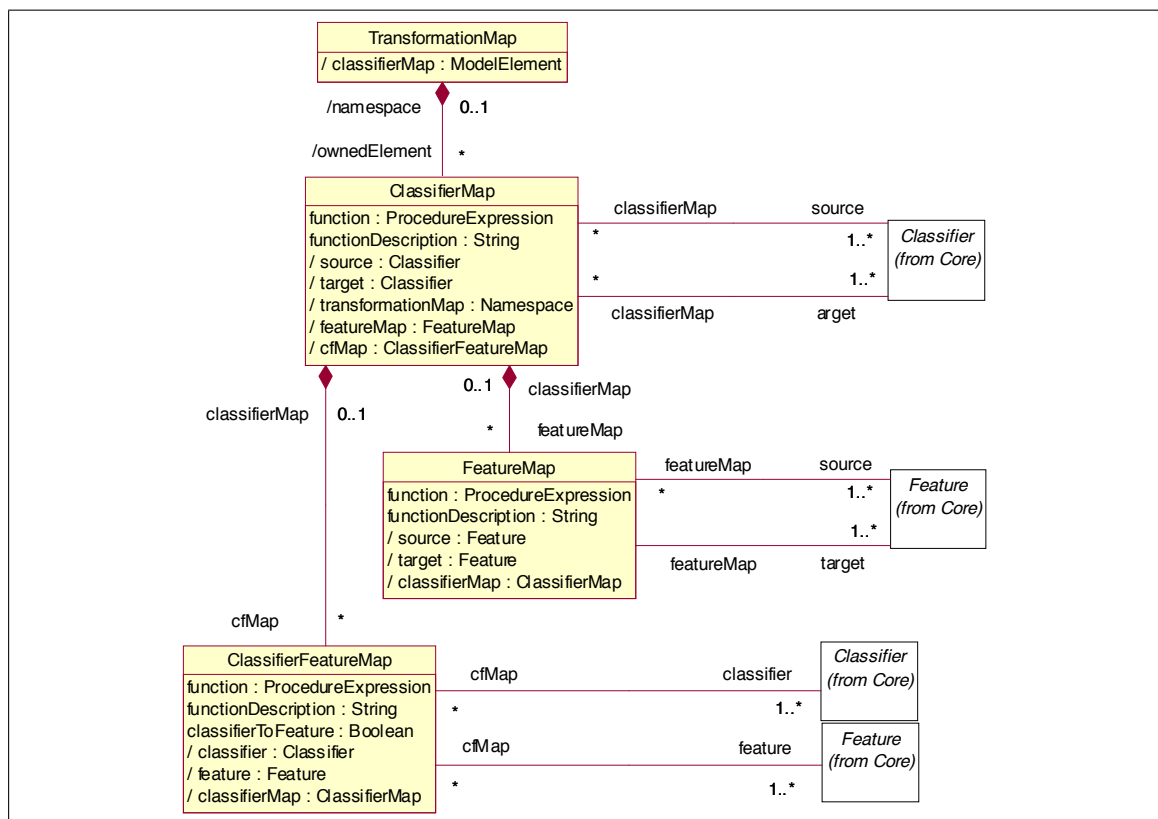


Figure 7.31: Transformation Metamodel (fragment) of the CWM (adopted from [135])

The OLAP package inherits from the Transformation package as shown in Figure 7.30. Deployments of an OLAP model are specified by defining the `TransformationMap` instances to link the logical OLAP objects to one another and to the objects representing the underlying physical data sources.

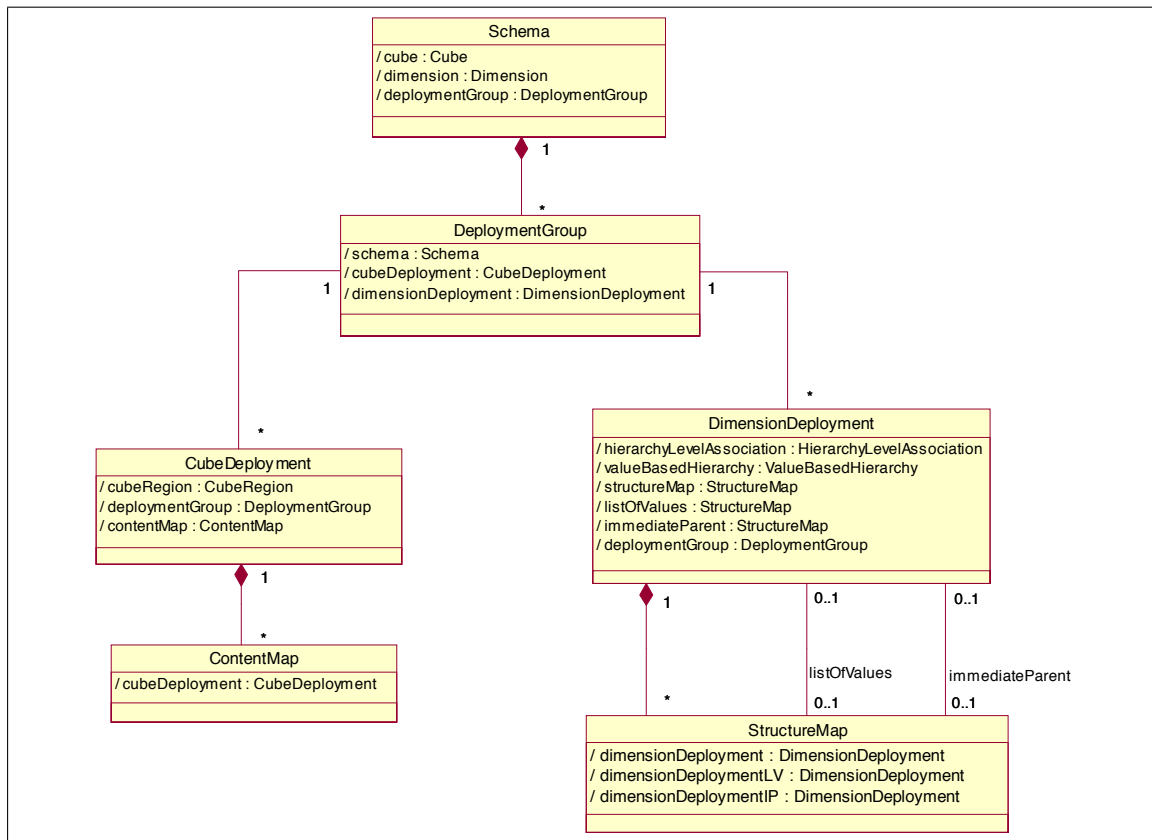


Figure 7.32: Deployment mapping structures in the CWM OLAP Metamodel (adopted from [135])

Figure 7.32 shows metaclasses and associations involved in defining deployment mappings between logical OLAP models and physical resource models. The OLAP package defines *StructureMap* as a subclass of *TransformationMap* for modeling structure-oriented transformation mappings, such as member identity and hierarchy structure. This type of transformation mapping is connected to the OLAP metamodel according to *Level* and *Hierarchy* to make the associations explicit. There are two specific usages of *StructureMap*: *i)* *ListOfValues* maps the attributes that identify the members at a specific *Level* (possibly, within a particular *Hierarchy*), and *ii)* *ImmediateParent* maps the attributes that identify the parent(s) of the members in a *Hierarchy*. Attribute-oriented transformations within an OLAP model, such as mapping dimension attributes to table columns, can be represented by relatively simple *TransformationMaps*. *ContentMap* is a specialization of *TransformationMap* for modeling content-oriented transformation mappings, for example, for mapping each of the *CubeRegion*'s measures to the columns of the underlying fact table.

The *DeploymentGroup* metaclass allows to combine the content (*CubeDeployment*) and the structure (*DimensionDeployment*) of a cube within the context of a single *Schema*. Several possible deployments of the same *Schema* can be specified.

Tools that implement the CWM share their metadata via CWM-compliant XML files. These files are created by following the XML-based format for metadata interchange (XMI) provided by the CWM. Each metamodel has a DTD (Document Type Definition) representation. A tool importing metadata via an XMI

document must validate the latter against the DTDs or otherwise assure the validity of the model.

As a practical demonstration of how the CWM implementation works, let us consider a CWM Enablement Showcase, which was held by the OMG in December 2000 and involved six vendors (IBM, Unisys, Hyperion, Oracle, SAS, and Meta Integration) as well as a large number of different types of warehouse tools [21]. Figure 7.33 contains extracts from a sample XML file, generated from within an Oracle data warehouse system. The file exports both the relational and the OLAP metadata referring to a dimension table *Period*.

```
<CWMRDB:Schema xmi.id="X1231" name="CWMDemo" visibility="public" isSpecification="false" namespace="X1229">
  <CWM:Namespace.ownedElement>
    <CWMRDB:Table xmi.id="X1268" name="Period" visibility="public" isSpecification="false" isRoot="true" isLeaf="true" isAbstract="false"
      isActive="false" isSystem="false" isTemporary="false" namespace="X1231">
      <CWM:Classifier.feature>
        <CWMRDB:Column xmi.id="X1249" name="EndDate" visibility="public" isSpecification="false" ownerScope="instance" precision="0"
          isNullable="columnNullable" owner="X1268" type="X1234"/>
      </CWM:Classifier.feature>
    </CWMRDB:Table>
  </CWM:Namespace.ownedElement>
</CWMRDB:Schema>
```

(a) CWM-compliant relational metadata

```
<CWMOLAP:Schema xmi.id="X1306" name="CWMDemo" visibility="public" isSpecification="false" namespace="X1229">
  <CWM:Namespace.ownedElement>
    <CWMOLAP:Dimension xmi.id="X1323" name="Period" visibility="public" isSpecification="false" namespace="X1306">
      <CWMOLAP:Dimension.cubeDimAssoc>
        <CWMOLAP:CubeDimAssoc xmi.idref="X1429"/>
      </CWMOLAP:Dimension.cubeDimAssoc>
    </CWMOLAP:Dimension>
  </CWM:Namespace.ownedElement>
  <CWM:Namespace.ownedElement>
    <CWMTFM:TransformationMap xmi.id="X1449" visibility="public" isSpecification="false" namespace="X1323">
      <CWM:Namespace.ownedElement>
        <CWMTFM:ClassifierMap xmi.id="X1450" visibility="public" isSpecification="false" namespace="X1449" transformationMap="X1449">
          <CWMTFM:ClassifierMap.source>
            <CWM:Classifier xmi.idref="X1268"/>
          </CWMTFM:ClassifierMap.source>
          <CWMTFM:ClassifierMap.target>
            <CWM:Classifier xmi.idref="X1323"/>
          </CWMTFM:ClassifierMap.target>
        </CWMTFM:ClassifierMap>
      </CWM:Namespace.ownedElement>
    </CWMTFM:TransformationMap>
  </CWM:Namespace.ownedElement>
  <CWMOLAP:Dimension.hierarchy>
    <CWMOLAP:LevelBasedHierarchy xmi.id="X1425" name="Standard" visibility="public" isSpecification="false" dimension="X1323">
      <CWMOLAP:LevelBasedHierarchy.hierarchyLevelAssoc>
        <CWMOLAP:HierarchyLevelAssoc xmi.id="X1404" visibility="public" isSpecification="false" levelBasedHierarchy="X1425" currentLevel="X1349"
          immediateParent="X1406">
          <CWM:Classifier.feature>
            <CWM:Attribute xmi.id="X1405" name="PRNT" visibility="public" isSpecification="false" owner="X1404"/>
          </CWM:Classifier.feature>
        </CWMOLAP:HierarchyLevelAssoc>
      </CWMOLAP:LevelBasedHierarchy.hierarchyLevelAssoc>
    </CWMOLAP:LevelBasedHierarchy>
  </CWMOLAP:Dimension.hierarchy>
</CWMOLAP:Schema>
```

(b) CWM-compliant OLAP metadata

Figure 7.33: A sample XML file for metadata export according to the CWM

Figure 7.33a shows a fragment of the relational metadata demonstrating the definition of relational sources at the example of column `EndDate` of table `Period` in schema `CWMDemo`. Figure 7.33b shows another part of the same XML file, which contains the OLAP metadata describing the hierarchical structure of dimension `Period` associated with the relational table `Period`. The Transformation package is used to map the OLAP metadata to its relational source. For example, the `ClassifierMap` element defines dimension `Period` as the target of the relational table `Period`.

7.4.2 Representing multidimensional properties in the CWM

It remains to investigate whether the OLAP metamodel of the CWM is capable of adequately representing all multidimensional properties defined by our conceptual model. Medina and Trujillo assess the appropriateness of the CWM as a standard for representing multidimensional properties in [122]. This evaluation refers to an object oriented UML-based approach to conceptual multidimensional modeling, proposed by the same research team and presented in [174]. The multidimensional properties assessed in that work represent a subset of the features supported by our extended model, therefore, the results of the CWM evaluation published in [122] appear highly relevant for our work.

Whereas representations of the fundamental constructs in the CWM OLAP metamodel are defined unambiguously, some of the special cases are not explicitly handled by this package. However, an appropriate mapping may be found at a lower layer of the CWM inheritance stack. For instance, there are no guidelines for describing partial roll-ups, fact and dimension degeneration, or non-strict hierarchies in the OLAP package itself, but it is possible to identify an appropriate metamodel construct in other packages, such as Core, Behavioral, Relationships, KeysIndexes, Relational, and Transformation. Especially the Object-Model metamodel at the heart of the CWM provides a solid framework for describing the data in terms of `ModelElements`, `Features`, `Constraints`, `TaggedValues`, `Attributes`, etc.

In Table 7.1 we enumerate the elements of the conceptual multidimensional model including those defined by our extended model and provide the corresponding CWM mechanisms for their mapping. As expected, representations of the major constructs, such as facts, dimensions, measures, hierarchies, and categories in the CWM are rather straightforward: each construct is represented by a designated metaclass, e.g., `Dimension`, `Hierarchy`, and `Level`.

As can be seen from Table 7.1, some of the features are only implicit in the respective metamodel representations. In most cases, such features are considered a pure implementation issue to be captured at a lower metamodel layer or were simply not supported at the time the last CWM specification version was drafted. Consider, for instance, fact degeneration (i.e., satellites, associations, and self-associations) introduced in Section 5.3. The OLAP package does not define any subclasses of `Cube`. However, degeneration can be recognized by inspecting the fact's dimensions as at least one of them should be defined upon a resource used as a fact table within the same `Schema`. Similarly, a non-measurable fact is identifiable via a non-inclusion of `Measure` attributes.

Whether an implied representation of a property is sufficient depends on the requirements of the Analysis layer and the intended purpose of the generated metadata. If intended for frontend tools, the pursued cube navigation approach may require the metadata to be more specific. For example, in our OLAP tool prototype, satellite facts should appear nested in their respective base facts. To support this requirement we had to extend the metadata with an explicit specification of degenerate facts and their associated base facts. Similarly, we needed an explicit association of fact types involved in a fact roll-up to incorporate the containing fact into a dimension of the contained one in the navigation tree.

It is important to note that the concepts of the unified multidimensional space and semantically related dimension categories are not reflected by the CWM OLAP metamodel. However, sharing is expressible at the Resource layer by mapping shared dimensions and/or category types to the same physical resource.

Table 7.1: Representing multidimensional properties in the CWM OLAP metamodel

Multidimensional construct	Representation in the CWM
Fact family	Metaclass Schema
Fact type (cube) <ul style="list-style-type: none"> ► Derived fact type ► Degenerate fact type ► Fact roll-up ► Non-measurable fact type ► Slice (subcube) 	Metaclass Cube Attribute <code>isVirtual</code> of the metaclass Cube Implied by including a dimension defined from another fact type resource Implied by including a fact identifier of another fact type as a dimension level Implied by the absence of associated Measure elements Metaclass CubeRegion
Measure <ul style="list-style-type: none"> ► Aggregability constraints 	Subclass Measure of metaclass Attribute Metaclass ClassifierFeatureMap
Dimension <ul style="list-style-type: none"> ► Time dimension ► Hierarchical dimension ► Multiple hierarchies 	Metaclass Dimension Attribute <code>isTime</code> in metaclass Dimension A single hierarchy reference in metaclass Dimension Multiple hierarchy references in metaclass Dimension
Hierarchy <ul style="list-style-type: none"> ► Hierarchy schema ► Default dimension hierarchy³ ► Derived hierarchy ► Parallel hierarchies ► Alternative hierarchies 	Metaclass Hierarchy Metaclass LevelBasedHierarchy <code>displayDefault</code> reference in Dimension <code>calcHierarchy</code> reference in metaclass CubeDimensionAssociation Assumed by default <i>Not supported!</i>
Dimension category <ul style="list-style-type: none"> ► Abstract category ► Totally ordered category 	Metaclass Level Attribute <code>isAbstract</code> in metaclass Level Metaclass DimensionDeployment
Roll-up relationships <ul style="list-style-type: none"> ► Fact-dimensional roll-up ► Roll-up between categories ► Partial roll-up 	Metaclasses of type Association Metaclass CubeDimensionAssociation Metaclass HierarchyLevelAssociation <i>Not supported!</i>

One obvious deficiency of the OLAP metamodel is that it does not distinguish between multiple alternative and parallel hierarchies: a `Hierarchy` is associated only to its `Dimension`, but there is no association between hierarchies. As a result, all hierarchies in dimension are treated as parallel. As a matter of fact, state-of-the-art OLAP tools also treat all multiple hierarchies as parallel, resulting in admissibility of invalid combinations of grouping criteria in OLAP queries [109]. Besides, the relational mapping of alternative hierarchies is identical to those of parallel ones. However, there is a big difference in how those hierarchies are allowed to be used when performing roll-up operations (see Section 4.5 for the details). Therefore, in our opinion, explicit support of alternative hierarchies is an indispensable extension of the OLAP metamodel due in a forthcoming version of the CWM specification. In our own implementation of a prototypical OLAP frontend, we enabled support for alternative hierarchies in the data navigation by extending the metamodel with the metaclass `HierarchyAssociation` for representing a pair of alternative hierarchies.

³The CWM OLAP specification suggests that a specific hierarchy may be designated as the default one, either for the display purposes or for a default consolidation path within a cube.

Table 7.2: Representing multidimensional properties in the CWM metamodels other than OLAP

Multidimensional construct	Representation in the CWM
Types of facts: ► Fact generalization	Metaclass Generalization of the Relationships package
Types of measures: ► Measure semantics ⁴ ► Measure derivation	Metaclass ClassifierFeatureMap of the Transformation package Metaclass FeatureMap of the Transformation package
Dimension types: ► Degenerated dimension ► Fact identifier ► Derived dimension	Implied by being mapped to the same resource at its fact Metaclass UniqueKey of the KeysIndexes package Implied by the derivation of its bottom-level category
Types of hierarchies: ► Generalized hierarchy	Metaclass Generalization of the Relationships package
Types of dimension categories: ► Derived category ► Conform categories	Metaclass FeatureMap of the Transformation package Implied by mapping the respective Levels to the same resource
Types of roll-up relationships: ► Non-strict roll-up	Metaclasses Association and AssociationEnd of the Relationships package

A partial roll-up relationship, i.e., an optional dimension level, and consequently, a set of partial related roll-ups, is another example of a non-supported concept in the OLAP metamodel, even though the resulting non-covering mappings can be implemented in a relational system by declaring the respective parent-level references as nullable. However, due to unavoidable non-summarizability of such mappings, our model proposes a set of normalization techniques for transforming non-covering schemes into specialization/generalization hierarchies (described in Section 7.2.1), which are expressible by the CWM.

The set of properties in Table 7.1 is not exhaustive as it encompasses only those concepts captured by the OLAP package itself. We now proceed by investigating the cases handled by other layers of the CWM, with the summary of the respective mappings presented in Table 7.2.

Flexibility of the CWM's abstraction mechanism results in a multitude of valid options for representing those multidimensional properties, which lie beyond the scope of the OLAP package. These options can be summarized as follows: *i*) re-interpreting the property to make it representable within the OLAP metamodel, *ii*) extending the OLAP metamodel by adding the missing elements, and *iii*) "pushing" the specification of the property down to a metamodel at a lower abstraction layer that provides the necessary mapping mechanisms. Metadata representations suggested in Table 7.2 refer to the latter category.

As an example of a construct, not documented by the CWM specification, let us consider a degenerated dimension. Since the OLAP metaclass `Dimension` does not have any subclasses, dimension degeneration has to be expressed by some other means. An example of re-defining the notion of a degenerated dimension as to "fit" it into the CWM OLAP metamodel is given in [122], where the authors suggest mapping it to a non-additive measure in the respective fact. However, such a re-interpreted representation may be inadequate for frontend tools, if the latter provide different display and/or navigation options for dimensions and measures.

⁴Measure semantics provides the background for the additivity behavior by distinguishing between transactional and snapshot measure types (see Section 5.2.1).

The option of extending the metamodel is generally less recommendable as it aggravates standard-conform metadata interchange. However, an extension may be feasible, if the CWM standard or the “work-around” options appear too inadequate for representing some concept. An example of a meaningful extension is the *Secure Relational Data Warehouse* (SECRDW) metamodel, proposed by Soler et al. in [165] as an extension of the Relational package of the CWM for representing security and audit measures.

Finally, the option of using lower-level metamodels for representing multidimensional properties in a more implementation-dependent fashion is more in line with the prevailing practices of CWM-compliant vendors. In data warehouse systems, any generated metadata always refers to a particular physical resource, i.e., each multidimensional property has real data behind it. Therefore, it is not the ultimate goal to specify the multidimensional properties in an implementation-independent fashion. Instead, the OLAP package is used primarily for capturing only those properties relevant for the front-end functionality, whereas all other details are “pushed down” to the underlying Resource layer. In our experience, this strategy is rather reasonable as it disburdens the Analysis layer from irrelevant implementation details.

A prominent implementation of the CWM is provided by Pentaho Metadata, which is a feature within a popular open-source BI Platform Pentaho [139]. While most vendors’ systems are based on proprietary metamodels and use the CWM solely as an import/export format for metadata interchange, Pentaho provides direct support for the CWM standard. As an example of capturing the multidimensional semantics at the Resource layer, let us consider how Pentaho represents computed measures. Figures 7.34 and 7.35 show a fragment of sample OLAP metadata describing cube Order Details and its underlying fact table PT_ORDERDETAILS. The XML fragment depicted in Figure 7.34 refers to the description of the cube in terms of measures and dimensions according to the CWM OLAP metamodel. Note that only two measures, namely Quantity Ordered and Price Each are defined at this level. The metadata in Figure 7.35 gives the relational mapping of the same cube in terms of a relational table and its attributes and demonstrates how a computed measure Total is specified: the measure’s table column PC_TOTAL is an aggregated field computed from the two base measures as `sum(QUANTITYORDERED*PRICEEACH)`.

```
<CWMOLAP:Cube xmi.id = 'a384' name = 'Order Details' isAbstract = 'false' isVirtual = 'false'>
  <CWM:ModelElement.taggedValue>
    <CWM:TaggedValue xmi.id = 'a385' tag = 'CUBE_BUSINESS_TABLE' value = 'BT_ORDER_DETAILS'>
    </CWM:ModelElement.taggedValue>
  <CWM:Namespace.ownedElement>
    <CWMOLAP:Measure xmi.id = 'a386' name = 'Quantity Ordered'>
      <CWM:ModelElement.taggedValue>
        <CWM:TaggedValue xmi.id = 'a387' tag = 'MEASURE_BUSINESS_COLUMN' value = 'BC_ORDER_DETAILS_QUANTITYORDERED'>
        </CWM:ModelElement.taggedValue>
      </CWMOLAP:Measure>
    <CWMOLAP:Measure xmi.id = 'a388' name = 'Price Each'>
      <CWM:ModelElement.taggedValue>
        <CWM:TaggedValue xmi.id = 'a389' tag = 'MEASURE_BUSINESS_COLUMN' value = 'BC_ORDER_DETAILS_PRICEEACH'>
        </CWM:ModelElement.taggedValue>
      </CWMOLAP:Measure>
    </CWM:Namespace.ownedElement>
  <CWMOLAP:Cube.cubeDimensionAssociation>
    <CWMOLAP:CubeDimensionAssociation xmi.id = 'a390' name = 'Orders' isAbstract = 'false'>
      <CWMOLAP:CubeDimensionAssociation.dimension>
        <CWMOLAP:Dimension xmi.idref = 'a391'>
        </CWMOLAP:CubeDimensionAssociation.dimension>
      </CWMOLAP:CubeDimensionAssociation>
    <CWMOLAP:CubeDimensionAssociation xmi.id = 'a392' name = 'Products' isAbstract = 'false'>
      <CWMOLAP:CubeDimensionAssociation.dimension>
        <CWMOLAP:Dimension xmi.idref = 'a393'>
        </CWMOLAP:CubeDimensionAssociation.dimension>
      </CWMOLAP:CubeDimensionAssociation>
    </CWMOLAP:Cube.cubeDimensionAssociation>
  </CWMOLAP:Cube>
```

Figure 7.34: Description of a computed measure at the Analysis layer (CWM OLAP)

```

<CWMRDB:Table xmi.id = 'a138' name = 'PT_ORDERDETAILS' isAbstract = 'false' isTemporary = 'false' isSystem = 'false'>
  <CWM:ModelElement.taggedValue>
    <CWM:TaggedValue xmi.id = 'a476' tag = 'TABLE_TARGET_DATABASE_NAME' value = 'SampleData'/>
  </CWM:ModelElement.taggedValue>
  <CWM:Namespace.ownedElement>
    <CWMRDB:Column xmi.id = 'a142' name = 'ORDERNUMBER'/>
    <CWMRDB:Column xmi.id = 'a150' name = 'PRODUCTCODE'/>
    <CWMRDB:Column xmi.id = 'a158' name = 'QUANTITYORDERED'/>
    <CWMRDB:Column xmi.id = 'a167' name = 'PRICEEACH'/>
    <CWMRDB:Column xmi.id = 'a176' name = 'ORDERLINENUMBER'/>
    <CWMRDB:Column xmi.id = 'a184' name = 'PC_TOTAL'/>
  </CWM:Namespace.ownedElement>
</CWMRDB:Table>
<CWM:Description xmi.id = 'a183' name = 'aggregation' body = 'sum' type = 'Aggregation'>
  <CWM:Description.modelElement>
    <CWMRDB:Column xmi.idref = 'a184'/>
  </CWM:Description.modelElement>
</CWM:Description>
<CWM:Description xmi.id = 'a185' name = 'exact' body = 'Y' type = 'Boolean'>
  <CWM:Description.modelElement>
    <CWMRDB:Column xmi.idref = 'a184'/>
  </CWM:Description.modelElement>
</CWM:Description>
<CWM:Description xmi.id = 'a186' name = 'fieldtype' body = 'Fact' type = 'FieldType'>
  <CWM:Description.modelElement>
    <CWMRDB:Column xmi.idref = 'a184'/>
  </CWM:Description.modelElement>
</CWM:Description>
<CWM:Description xmi.id = 'a187' name = 'formula' body = 'sum(QUANTITYORDERED*PRICEEACH)' type = 'String'>
  <CWM:Description.modelElement>
    <CWMRDB:Column xmi.idref = 'a184'/>
  </CWM:Description.modelElement>
</CWM:Description>
<CWM:Description xmi.id = 'a188' name = 'hidden' body = 'N' type = 'Boolean'>
  <CWM:Description.modelElement>
    <CWMRDB:Column xmi.idref = 'a184'/>
  </CWM:Description.modelElement>
</CWM:Description>
<CWM:Description xmi.id = 'a189' name = 'name' body = 'Total' language = 'en_US' type = 'LocString'>
  <CWM:Description.modelElement>
    <CWMRDB:Column xmi.idref = 'a184'/>
  </CWM:Description.modelElement>
</CWM:Description>

```

Figure 7.35: Description of a computed measure at the Resource layer (CWM Relational)

Further examples for using the CWM for describing advanced multidimensional concepts can be found in the work of Medina and Trujillo [122] and in the comprehensive developer's guide to the CWM [147].

Chapter 8

Interactive Exploration of OLAP Aggregates

THE AIM OF THIS CHAPTER is to address the issue of supporting complex data at the analysis and presentation layer, based on the emerging Visual OLAP paradigm. We describe an overall framework for comprehensive exploration of OLAP aggregates in terms of two major components: *i*) data navigation as an interface for interactive query specification and *ii*) a visualization toolkit providing a set of interactive visualization techniques of various complexity, which serve as the output layout for exploring the retrieved query results and gaining a deeper insight into the data. In particular, we propose a novel visualization technique called *Enhanced Decomposition Tree*, which combines a hierarchical layout for arranging aggregate values obtained in a series of successive drill-down steps into a visual decomposition hierarchy. Enhanced Decomposition Tree employs embedded charts at the node level aimed at improving visual comparability of the presented values. The proposed technique offers multiscale visualization while naturally preserving the history of the interaction.

Contents

8.1	Visual Analysis Framework	168
8.1.1	Related Work on Visualization for OLAP	168
8.1.2	Components of a Visual OLAP Tool	172
8.2	Navigating in Multidimensional Data	176
8.2.1	Prevailing Data Navigation Schemes	176
8.2.2	Enhancing Data Navigation through Scheme Awareness	178
8.2.3	Dynamic Properties of the Navigation Scheme	182
8.2.4	OLAP Operators and their Implementation Options	186
8.3	Hierarchical Visualization Techniques for OLAP	188
8.3.1	Salient Characteristics of Visual Interaction Patterns	188
8.3.2	Decomposition Tree	188
8.3.3	Enhanced Decomposition Trees	191
8.3.4	Spatio-temporal Visualization Techniques	199

8.1 Visual Analysis Framework

The multidimensional data model underlying the OLAP technology has proven to be particularly suitable for sophisticated analysis of large multivariate data volumes. With rapid evolution of this technology in the last decade, unprecedented volumes of data have become available for analysis and exploration. To cope with the information overload, analysts use advanced BI frontends that enable purely visual and rather intuitive interaction with the data. Therefore, the ultimate benefit of the data warehousing approach is determined by the functionality, usability, and “intelligence” of end-user interfaces.

The last decade has witnessed an explosion of visual interfaces for OLAP – dashboards, charts, maps and scatterplots – that have impacted modern BI analytics. Interactive features, such as zooming, slicing, brushing, and filtering are becoming a commonplace in analysis software. With ever-growing volumes of accumulated data visualization becomes indispensable for extracting useful knowledge from data by a human expert. Adequate visual presentation helps to rapidly reveal patterns, recognize trends or anomalies. Especially the ad hoc queries, driven by a mere guess or a hypothesis about the knowledge hidden in the “raw” data, benefit from the ability to visually specify the data set of interest and interact with it. Arranging the data into a multidimensional space is especially beneficial for decision support due to the potential of retrieving the data subsets of interest in the form exactly satisfying the user’s information needs.

The term *Visual OLAP*, already introduced in Section 2.3.1, encompasses a new generation of OLAP end-user tools for interactive ad hoc exploration of large multidimensional data volumes. While traditional analysis tools designed primarily to support routine reporting and analysis use visualization merely for expressive presentation of the data, in Visual OLAP it plays a key role as the method of interactive query-driven analysis. Continuous efforts are put into providing new approaches to visual exploration of OLAP cubes, such as hierarchical visualizations (decomposition trees, chart trees, TreeMaps, etc.), multiscale views, interactive and animated scatter-plots, described in the next section.

Analysis tools that abound the market offer a multitude of features and functions that require training and skill to understand and use. Feature overload and usability deficiencies often lead to loss of orientation and discourage users from using novel techniques. The work reported in this chapter is an attempt to enhance the Visual OLAP approach in terms of functionality and user-friendliness. Applicability of a particular visualization technique depends on various criteria, such as the type of the analytical task, data volume and complexity, user preferences and skills. Our approach accounts for a variety of tasks by providing an exploration framework, in which users can experiment with various layouts and techniques to find satisfactory solutions to specific problems.

8.1.1 Related Work on Visualization for OLAP

The work related to the topic of this chapters in one way or another can be subdivided into two major groups, namely, visual analysis systems and advanced visualization techniques for OLAP.

VISUAL ANALYSIS SYSTEMS

First proposals to use visualization for exploring multidimensional data were not tailored towards OLAP applications, but rather addressed the generic problem of visual querying of large datasets stored in a database. Keim and Kriegel [75] proposed VisDB, a visualization system based on a new query paradigm. In VisDB, users are prompted to specify an initial query. Thereafter, guided by visual feedback, they dynamically adjust the query, e.g., by using sliders for specifying range predicates on single attributes. Retrieved records are mapped to the pixels of the rectangular display area colored according the degree of their conformity to the specified set of selection predicates and positioned according to a grouping or ordering directive.

Another example of an early work related to multidimensional data exploration can be found in [44], where an intelligent visual interface CoDecide for cooperative analysis of spreadsheet data is proposed. CoDecide links multiple views of a data cube in a multi-perspective and multi-user mode and uses “tape” representations for visualizing the problem dimensions.

OLAP tools of the current state of the art provide the classical pivot table interface along with a set of popular business visualization techniques, such as charts and time series as well as more sophisticated layouts, such as scatterplots, maps, graphs, cartograms, matrices, grids, etc., and proprietary visualizations (e.g., ProClarity Decomposition Tree [141] and Fractal Map [22]). In the abundance of existing OLAP tools, we limit ourselves to naming a few products, which offer distinguished features.

Tableau Software [169] and other established OLAP vendors deliberately restrict the set of supported visualizations to the popular and proven ones, such as tables, charts, maps, and time series, doubting general utility of exotic visual metaphors [55]. Polaris, a visual tool for multidimensional analysis developed by the research team of Pat Hanrahan at Stanford University [167], is a predecessor of Tableau Software. Polaris inherits the basic idea of the classical pivot table interface that maps aggregates onto a grid defined by dimension categories assigned to the grid’s rows and columns. However, Polaris uses embedded graphical marks rather than textual numbers in the table cells. The types of supported graphics are arranged into a taxonomy, comprising rectangle, circle, glyph, text, Gantt bar, line, polygon, and image layouts.

Advizor system [39] implements a technique that organizes data into three perspectives. A perspective is a set of linked visual components displayed together on the same screen. Each perspective focuses on a particular type of analytical task, such as *i*) single measure view using a 3D multiscape layout, *ii*) multiple measures arranged into a scatterplot, and *iii*) anchored measures presented using techniques from multidimensional visualization (e.g., Box Plots [177] or Parallel Coordinates [69]). A more recent survey [40] investigates common visual metaphors and associated interaction techniques with improved visual scalability, i.e., the capability to effectively display large volumes of multidimensional data, and describes how the proposed techniques were implemented in the Advizor system.

ProClarity Analytics [150] is famous for innovative visualization tools, such as the Decomposition Tree, Perspective View, and Performance Map. In 2006, ProClarity became a Microsoft subsidiary and as of 2007, ProClarity Analytics was released as a part of the Microsoft Office Performance Point Server [34].

Another noteworthy tool is Report Portal – a web client reporting solution for Microsoft Analysis Services released by XMLA Consulting [188]. Report Portal in its current version 2.2 offers interactive OLAP and data mining reports based on visualization techniques, such as GIS maps, chart trees, TreeMaps, dashboards, and animated scatterplots (moving bubbles).

VISUALIZATION TECHNIQUES

A pioneering and fundamental work on automating visualization of relational data was carried out by Jock Mackinlay [105], who proposed to define visual presentations in terms of graphical languages. Graphical languages encode syntactic and semantic properties of graphical presentations in form of sentences, similar to other formal languages. Expressiveness and effectiveness criteria are used to assess the quality of a graphical encoding and to compare various presentation alternatives with one another. By formalizing graphical presentation as a collection of graphical languages, Mackinlay’s approach provides an abstraction for automatic synthesis of effective visual designs for a variety of data sets, focusing on two-dimensional static representations, such as bar-charts, scatterplots, and connected graphs.

Besides the classical visualization techniques, such as the pivot table and 2-dimensional plots and charts familiar to any data analyst, a wealth of more comprehensive visual frameworks for incremental exploration of and navigation in large multidimensional data volumes have emerged. Visualization techniques applicable in the OLAP context can be roughly grouped into the following categories (see [140] for further details):

- ◆ *Geometric* (Scatterplots, Landscapes, Hyperslices, Parallel Coordinates)
- ◆ *Icon-based* (Chernoff Faces, Stick Figures, Color Icons, TileBars)
- ◆ *Pixel-oriented* (Recursive Pattern, Circle Segments)
- ◆ *Hierarchical* (Dimensional Stacking, Worlds-within-Worlds, TreeMap, Cone Trees, InfoCube)
- ◆ *Graph-Based* (Straight-, Poly- and Curved-Line, DAG, Symmetric, Cluster)
- ◆ *Hybrid* techniques which arbitrarily combine any of the above.

Applicability of a particular technique or a meaningful combination of techniques depends largely on the analysis needs and the level of user expertise.

Figure 8.1 presents a structured overview of visualization techniques for OLAP arranged into four quadrants according to the layout (simple vs. hybrid) and granularity (uniform vs. mixed). The techniques in each quadrant are sorted upwards in the increasing order of the maximum number of dimensions they can support. Visual metaphors capable of displaying multiple measure fields are shown with orange background. This enumeration is by no means exhaustive and contains only the major techniques provided by existing OLAP tools or proposed in the research literature. Descriptions of the most of the enumerated techniques may be found in standard literature on information visualization [20, 176], as well as in industrial and research publications [119, 168, 172, 178].

Any OLAP tool implements just a small subset of the visualizations listed in Figure 8.1, mostly from the upper-left quadrant of simple layouts. However, there is a trend towards adopting novel and more complex layouts to support a wider spectrum of analysis tasks. This trend raises the issue of assisting the user in choosing a “good” visualization. In data warehouse systems, the issue of assessing the aptitude of a particular visualization approach for solving different types of analysis tasks is rather neglected. Typically, the user has to find an appropriate solution manually by experimenting with different layout options. As a result, users often come up with inefficient and even misleading visualizations. Apparently, a successful visual OLAP framework needs to be based on a comprehensive taxonomy of domains, tasks, and visualizations.

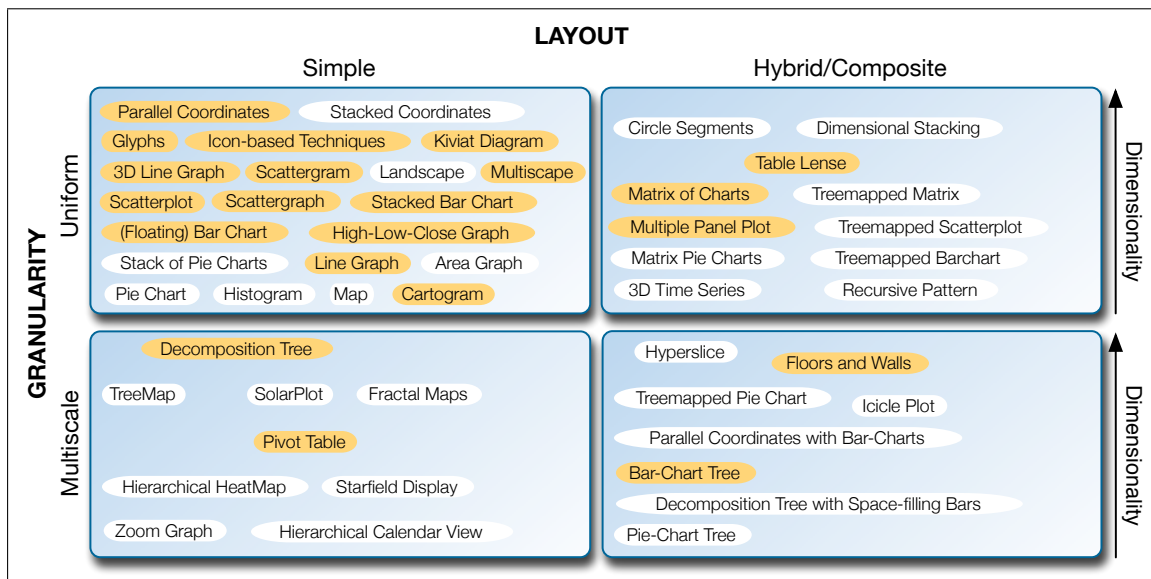


Figure 8.1: Segmentation of visualization techniques for OLAP by layout and granularity

To support a large set of diverse visualization techniques and to enable dynamic switching from one technique to another, an abstraction layer has to be defined for specifying the relationships between the data and its visual presentation. Maniatis et al. [111] propose an abstraction layer solution, called the *Cube Presentation Model* (CPM), which distinguishes between two layers: the logical layer deals with data modeling and retrieval whereas the presentation layer provides a generic model for representing the data visually (normally, on a 2D screen). The entities of the presentation layer include points, axes, multicubes, slices, tapes, cross-joins, and content functions. The authors demonstrate how CPM constructs can be mapped to advanced visual layouts at the example of the Table Lens – a technique based on a cross-tabular paradigm with support for multiple zoomable windows of focus.

While many OLAP vendors restrict the set of supported visualizations to the popular and proven ones and doubt general utility of exotic visual metaphors, some research works suggest enriching the visual OLAP framework by extending basic charting techniques or employing novel and less known visualization techniques to take full advantage of multidimensional and hierarchical properties of the data [95, 164, 170, 172].

Tegarden [172] formulates the general requirements of business information visualization and gives an overview of advanced visual metaphors for multivariate data, such as *Kiviat diagrams* and *Parallel Coordinates* for visualizing data sets of high dimensionality, as well as 3D techniques, such as *3D Scattergrams*, *3D line graphs*, *floors and walls*, and *3D map-based bar-charts*.

Another branch of visualization research for OLAP concentrates on developing *multiscale* visualization techniques capable of presenting the data at different levels of aggregation. Stolte et al. describe their implementation of multiscale visualizations within the framework of the Polaris system [168]. The underlying visual abstraction is that of a *zoom graph* that supports multiple zooming paths, where zooming actions may be tied to dimension axes or triggered by a different type of interaction.

Lee and Ong propose a multidimensional visualization technique that adopts and modifies the Parallel Coordinates method for knowledge discovery in OLAP [95]. The main advantage of this technique is its scalability to virtually any number of dimensions. Each dimension is represented by a vertical axis and the aggregates are aligned along each axis in form of a bar-chart. The other side of the axis may be used for generating a bar-chart at a higher level of detail. Polygon lines adopted from the original Parallel Coordinates technique are used for indicating relationships among the aggregates computed along various dimensions (a relationship exists if the underlying sets of fact entries overlap in both aggregates).

Sifer [164] presents a multiscale visualization technique for OLAP based on *coordinated views of dimension hierarchies*. Each dimension hierarchy with qualifying fact entries attached as the bottom-level nodes is presented using a space-filling nested tree layout. Drilling-down and rolling-up is performed implicitly by zooming within each dimension view. Filtering is realized by (de-)selecting the values of interest at any level of dimension hierarchies, resulting either in highlighting the qualifying fact entries in all dimension views (*global context coordination*) or in eliminating the disqualified entries from the display (*result only coordination*). A similar interactive visualization technique, called the *Hierarchical Dynamic Dimensional Visualization* (HDDV), is proposed in [170]. Dimension hierarchies are shown as hierarchically aligned barsticks. A barstick is partitioned into rectangles that represent portions of the aggregated measure value associated with the respective member of the dimension. Color intensity is used to mark the density of the number of records satisfying a specified range condition. Unlike in [164], dimension level bars are not explicitly linked to each other, allowing to split the same aggregate along multiple dimensions and, thus, to preserve the execution order of the disaggregation steps.

One of the major visualization challenges for OLAP is the ability to present a large number of dimensions on a display. An additional visual attribute for mapping a dimension could be *animation*, as found in the Gapminder software for interactive data exploration using animated scatterplots in which animation is used to show the evolution of values along the timeline [157]. A well-structured classification of visualization and interaction techniques with respect to the type and the dimensionality of the data is produced in [77].

A technique for finding an appropriate ordering of the aggregates along dimensional axes, proposed by Wei Choong et al. in [26], may help to improve the analytical quality of any visualization. By default, the ordering of the measures is imposed by the lexical ordering of dimension members. To make patterns more obvious, the user has to rearrange the ordering manually. The proposed algorithm automates the ordering of measures in a representation as to best reveal the patterns (e.g., trends, similarity) in a data set.

Whenever a data cube contains spatio-temporal characteristics, the analysis may benefit from specialized exploration techniques for space-time patterns. Rivest et al. [155] propose SOLAP (spatial OLAP) as a visual platform for spatio-temporal analysis using cartographic and general displays. The authors also define different types of spatial dimensions and measures as well as a set of specialized geometry-aware OLAP operators. A synopsis of techniques for spatio-temporal exploration arranged according to the data and task types is produced in [6]. Kuchar et al. [88] point out that time dimension is not an ordinary data attribute and that, therefore, to ensure satisfactory analysis, interaction and visualization techniques have to incorporate explicit awareness of temporal characteristics.

8.1.2 Components of a Visual OLAP Tool

Comprehensive analysis includes a variety of tasks such as examining the data from multiple perspectives, extracting useful information, verifying hypotheses, recognizing trends, revealing patterns, gaining insight, and discovering new knowledge from arbitrarily large and/or complex data volumes. In addition to conventional operations of analytical processing, i.e., drill-down, roll-up, slice-and-dice, pivoting, and ranking, OLAP frontends support further interactive data manipulation techniques, such as zooming and panning, filtering, brushing, collapsing, distorting, etc.

OLAP tools account for a diversity of potential analytical tasks by providing a comprehensive framework for interactive generation of desired visual presentations. The overall query specification cycle evolves by *i*) selecting a data source of interest, *ii*) choosing a desired visual layout (e.g., a scatterplot or a pivot table), and *iii*) mapping various data attributes to these structural elements of the chosen layout (e.g., the horizontal and the vertical axis of a plot) as well as to other visual attributes, such as color, shape, and size.

The entire exploration framework can be considered as composed of an input and an output area for specifying queries and presenting query results, respectively. The input component has the form of a navigation interface for visual querying of data sources by presenting data cubes as browsable structures. The output area presents the results of user interactions in a selected visual format and enables interactive exploration by providing a taxonomy of available visual layouts and attributes along with a toolkit of interaction techniques for dynamic refinement of the queried data subset and its visual representation. A unified framework is obtained by designing an abstraction layer for each element and providing mapping routines (e.g., metadata to a navigation hierarchy, navigation events to database queries, and query results to a visual layout) that implement the interaction between different layers.

VISUAL QUERY SPECIFICATION

Visual OLAP disburdens the end-user from composing queries in the “raw” database syntax (e.g., SQL or MDX). Instead, queries are specified visually. Multidimensional data is represented as a browsable structure whose elements can be queried by “pointing-and-clicking” and “dragging-and-dropping”. The visual interface does not trade advanced functionality off for simplicity, it rather facilitates the process of specifying ad hoc queries of arbitrary complexity.

While analytical queries aggregate over detailed data, visual exploration evolves in the inverse direction, i.e., “descending” from coarsely grained views towards more detailed ones via a stepwise decomposition into subaggregates along selected dimensions. This prevailing drill-down direction is reflected in the structure

of a typical OLAP data navigation: each data cube is presented as a hierarchy of its dimensions and each dimension is a recursive top-down nesting of granularity levels, i.e., with the coarsest granularity at the top and the finest at the bottom. Users proceed by specifying the measure(s) (both the data field and the aggregate function), choosing the dimensions to be used as decomposition axes, filtering the selected data subset, and manipulating the visual representation of the result. These query steps are performed irrespective of the query type and the chosen visualization technique. Therefore, we see a great potential for improving the usability in designing a common uniform navigation framework for satisfying any type of analytical query.

Various navigation events, such as dragging and clicking, are translated into valid queries and executed instantaneously. Therefore, from the user's point of view, querying is done implicitly by populating the visualization with data and incrementally refining the data view. The first step is to instantiate an empty visualization template with data, performed by dragging the elements (measures, dimensions) of interest into the respective layout areas. Figure 8.2 shows an example of instantiating a visualization in the Tableau Software: an empty pivot table template prompts the user to drop data fields from the navigation (left) into the column, the row or the cell area.

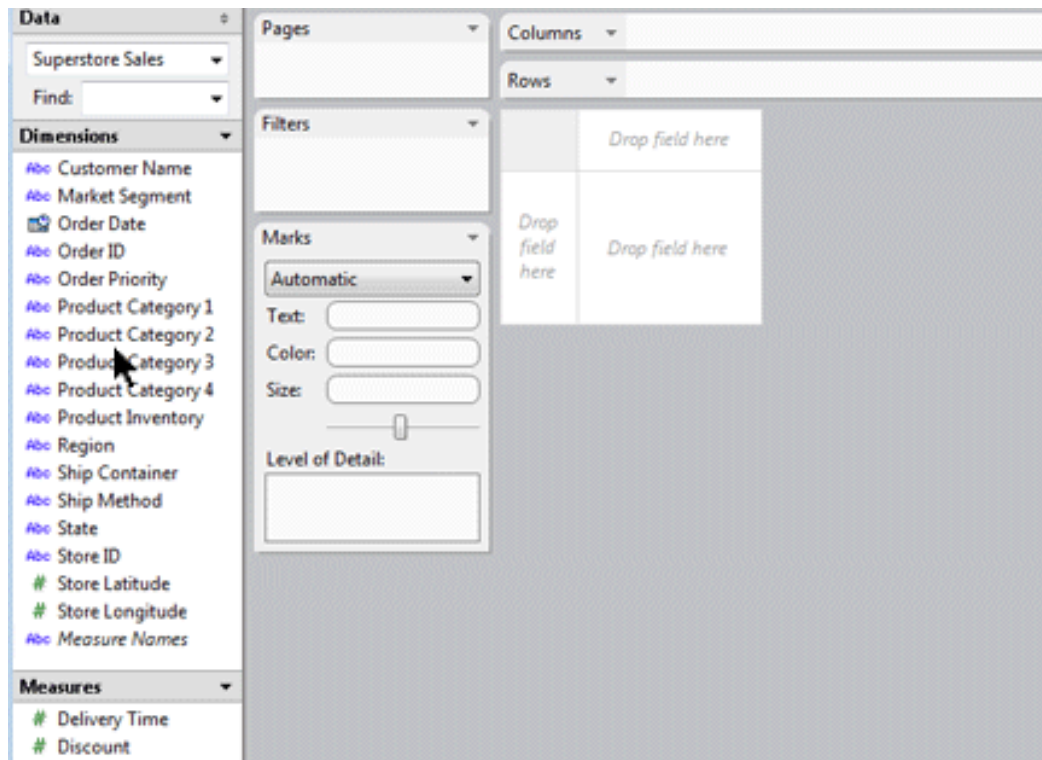


Figure 8.2: Mapping data fields to a visual layout in Tableau

Any OLAP query follows the same scheme, i.e., consists of the same query clauses, some of which are optional. In ROLAP systems, database queries are expressed in SQL and structured into the following sequence of clauses (optional clauses and elements are placed in square brackets):

```

SELECT [ dimension_list, ] measure_list
FROM table_list
[ WHERE predicate_list ]
[ GROUP BY [ ROLLUP | CUBE ] dimension_list ]
[ HAVING measure_predicate_list ]
[ ORDER BY attribute_list [sort_direction] ]

```

Elements `measure_list` and `table_list` are obligatory and have to be populated with at least one measure and one fact table, respectively. Thereby, the simplest possible query for instantiating a visualization with a grand total value is generated by picking a data cube and some measure field in it. For example, picking the field `Discount` from the cube `Superstore Sales` in Figure 8.2 would generate the following SQL query:

```

SELECT SUM(discount) FROM superstore_sales

```

Further clauses serve for refining the initial query: *i)* **WHERE** and **HAVING** clauses allow to specify selection conditions on any attributes and aggregated measure fields, respectively, *ii)* **GROUP BY** contains dimension categories to aggregate along, and *iii)* **ORDER BY** sorts the output. These clauses are populated with data by invoking corresponding OLAP operations described in the next section.

VISUALIZATION OF QUERY RESULTS

In the context of OLAP, visualization refers to the mapping of the data returned by a query or a series of queries to a visual layout. The output of any OLAP query is a data cube. Visual presentation is generated by assigning the cube's elements – measures and dimensions – to visual variables of the display. A visualization technique is defined by its graphical primitives, such as line or circle segments, points, curves, etc., which in combination determine the layout template. Further visual variables, such as color, position, length, and area, are used for encoding various properties of the data set into its visual presentation.

Users analyze the visual presentation by extracting the quantitative information encoded into the graphics in the form of perceptual tasks. Visual analysis tasks are quite different from those encountered in classical data analysis. The former include recognizing shapes, discerning colour, judging sizes and distances, tracing motion, etc. Obviously, various tasks differ in their accuracy and ease of interpretation. Cleveland and McGill [28] propose the notion of *elementary perceptual tasks*, or *elementary graphical encodings*, to describe the basic way of encoding data into a visualization. The authors also provide an empirically verified ranking of those tasks according to the accuracy of quantitative perception. Mackinlay [105] extended the set of considered tasks by addressing the issue of encoding non-quantitative information and provided a ranking of perceptual tasks according to the data type (quantitative, qualitative, nominal).

The commonly recognized perceptual tasks in descending order of accuracy for quantitative data domains according to [28, 105] (except for animation, which was not evaluated in those studies) are the following ones:

- ◆ *Position* (e.g., a coordinate of a point along an axis),
- ◆ *Length (distance)* (e.g., length of a bar in a bar-chart),
- ◆ *Angle* (e.g., angle of a segment in a pie-chart),
- ◆ *Slope* (e.g., slope of a line in a line-chart),
- ◆ *Direction (orientation)* (e.g., direction of an edge in a graph),
- ◆ *Area (size)* (e.g., a rectangular area of a node in a TreeMap),
- ◆ *Volume* (e.g., volume of a 3-D shape),
- ◆ *Curvature* (e.g., curve-difference charts),
- ◆ *Density (darkness)* (e.g., greyscale colormap),
- ◆ *Color saturation (brightness)* (e.g., fading effect in the animation),

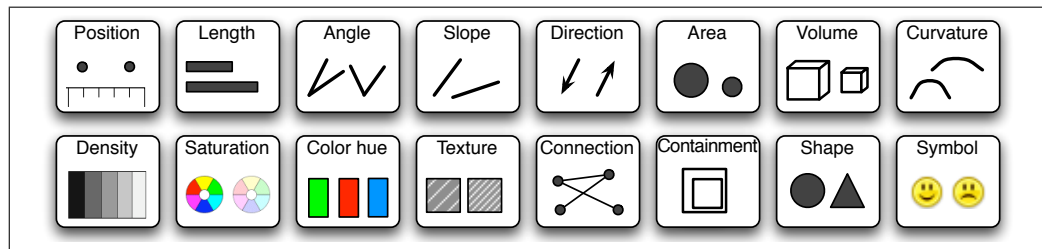


Figure 8.3: Elementary perceptual tasks in visual data analysis

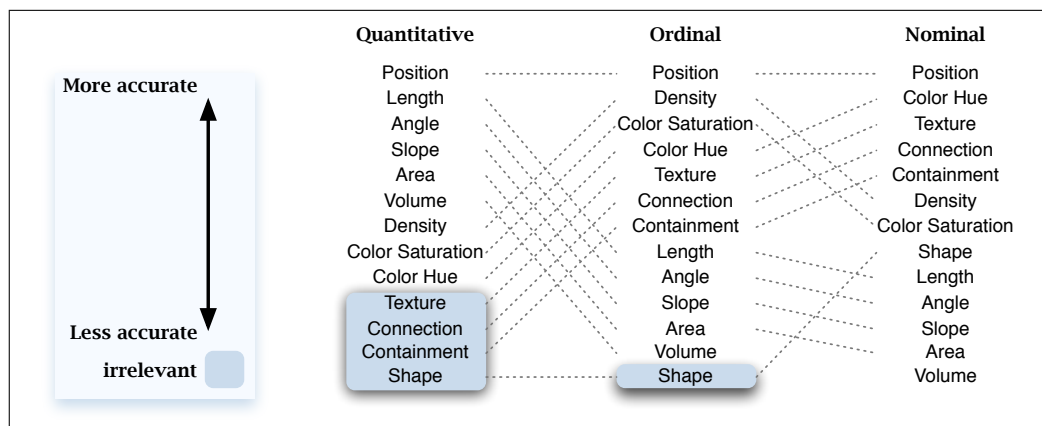


Figure 8.4: Ranking of perceptual tasks with respect to the data type

- ◆ *Color hue* (e.g., assigning distinct colors to the segments of a pie-chart),
- ◆ *Texture (shading)* (e.g., assigning different filling patterns to shapes),
- ◆ *Connection* (e.g., connecting related nodes in a graph by an edge),
- ◆ *Containment* (e.g., nesting child nodes within a parent node in a TreeMap),
- ◆ *Shape or symbol* (e.g., using marks with different shapes or borders in a scatterplot),
- ◆ *Animation (motion)* (e.g., animating the evolution of a value along the time axis in a scatterplot).

Figure 8.3 shows simple pictorial symbols that describe the main idea of each elementary perception task. A visual layout typically employs a combination of different tasks to encode multiple characteristics of the data items. For example, bar-charts use position and length, whereas stacked bar-charts additionally make use of containment and color.

Since OLAP is concerned with the analysis of quantitative information, the accuracy of the utilized visual elements is a crucial requirement. However, the adequacy of the perceptual tasks for non-quantitative data types is also an important issue since dimensional characteristics may be of different data types – numeric, ordinal, or nominal. Ranking of perceptual tasks according to the encoded data type, proposed by Mackinlay [105], is shown in Figure 8.4. This ranking is related to relational data in general. Data sets retrieved by OLAP queries can be considered a special case of relational data consisting of two types of attributes, namely, numeric measures and descriptive dimensions. Therefore, ranking of quantitative tasks is especially relevant for encoding measures whereas ordinal and nominal ranks should be considered for mapping dimensions.

Prevailing visual layouts and default visualization metaphors in the area of OLAP generally adhere to the proposed rankings: popular presentations are business charts and scatterplots, which map measures to length, angle, volume, area, or color, encoding their dimensional characteristics into position, density, or color hue.

Intuitively, the more perceptual tasks a particular visualization combines, the more characteristics of a data set can be presented. Simple bar-charts and pie-charts are capable of showing just a single measure grouped along a single dimension category, scatterplots support two dimensions and pivot tables allow to display multiple measures and to nest multiple dimensions in its rows and/or columns. Comprehensive analysis tasks may require more expressive visualization techniques. Popular approaches to increasing the dimensionality are to extend a 2-dimensional layout to 3-D (e.g., 3D charts and maps in Miner3D [124]), to use hierarchical layouts (e.g., Chart Trees in Report Portal [188]), to split the view into multiple perspectives (e.g., perspectives in Advizor [39]), or to arrange it into a grid of small multiples (e.g., visual tables in Tableau [169]). Another emerging trend is to adopt specialized multidimensional visualization techniques, such as Parallel Coordinates [69], which scale to a higher number of dimensions.

8.2 Navigating in Multidimensional Data

A common interface for accessing the data in a cube is a browser-like navigation hierarchy, which represents each cube as a hierarchical node consisting of dimensions and measures. Dimensions are also represented in a hierarchical fashion, i.e., as a recursive nesting of hierarchy levels. Comprehensive OLAP tools allow users to access the data residing in different data sources. Therefore, the navigation hierarchy in such tools is rooted at the node of type *Database*. Relational databases are partitioned into schemes (not to be confused with conceptual or logical schemes!), where a schema specifies a collection of table definitions. Therefore, the next-level node in the navigation is of type *Schema*, which, in its turn, is a parent node of *Cube*. In simplified configurations, the navigation may be limited to single data cubes and, thus, consist solely of dimension and measure attributes of a selected cube.

8.2.1 Prevailing Data Navigation Schemes

The cube navigation is typically subdivided into *Dimensions* and *Measures* sections, as can be found in nearly any popular commercial BI tools, e.g., IBM Cognos PowerPlay [31], SAP BusinessObjects Intelligence Platform [14], and MicroStrategy OLAP Services [123]. Most of the tools also agree that dimensions are to be represented in a straightforward fashion, i.e., by their hierarchy instances, where each member is a node containing its child members. We denote this kind of navigation *instance-based*. Figure 8.5 shows a screenshot of a data navigation in Cognos PowerPlay. Consider, for instance, the navigation hierarchy of Product dimension: the top-level node ALL PRODUCTS contains single product categories, which, in their turn, contain the respective product groups. The hierarchy scheme $\text{Product} \sqsubseteq \text{Product category} \sqsubseteq \top_{\text{Product group}}$ is not explicit in such a presentation but is expected to be guessed by the user.

In an instance-based navigation, each dimension is accessed in a top-down fashion, so that lower-level elements are reached exclusively via their containing upper-level elements. For example, in the time hierarchy with the scheme $\text{date} \sqsubseteq \text{month} \sqsubseteq \text{quarter} \sqsubseteq \text{year}$, date value ‘February, 16 2004’ is reached by successively expanding the nodes ‘2004’, ‘Q1’, and ‘February’. Thereby, instance-based hierarchy presentation has some obvious disadvantages, such as tedious navigation to lower hierarchy levels, having to expand multiple paths to access multiple members at the same level, and the premise that the user knows the path leading to the element of interest (e.g., a user interested in software products must know that this product group is assigned to the product category ENTERTAINMENT MEDIA).

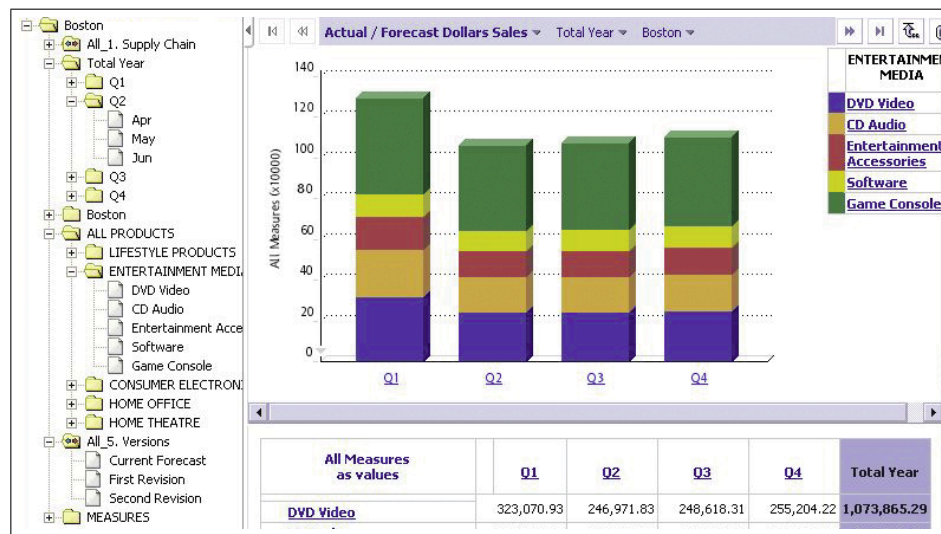


Figure 8.5: Instance-based hierarchy navigation in Cognos PowerPlay



Figure 8.6: Explicit enumeration of hierarchy levels in OracleBI Discoverer

Some OLAP vendors tried to compensate for the above disadvantages by enhancing the navigation with some kind of scheme awareness. For example, OracleBI Discoverer [32] allows to “jump” to the desired hierarchy level in a dimension by providing a drop-down list of all available levels, as shown in Figure 8.6. Other vendors treat hierarchy levels as distinct dimensions in the navigation, as can be seen in the screenshot of MicroStrategy OLAP Services depicted in Figure 8.7. In this example, Product dimension is resolved into hierarchy levels Category and Subcategory and Time dimension is shown as Year and Month. We denote the latter navigation approach *category-based*.

Category-based navigation is superior into the instance-based one as it provides a more detailed overview of the available granularities and facilitates the navigation to the desired level of detail and the desired subset of member values. More importantly, it becomes possible to support multiple and heterogeneous hierarchies since each category can be presented as a dimension in its own right. However, many issues remain unsolved or aggravated: belonging of a category to a particular dimension and hierarchical relationships between

The screenshot shows the MicroStrategy OLAP Services interface. On the left is a navigation pane with a tree structure of dimensions: Category, Month, Region, Subcategory, Year, Profit, Profit Margin, Revenue, and Units Sold. The main area displays a pivot table. The dimensions are Category, Year (set to 2004), and Region. The measures are Revenue and Profit. The table shows data for Books, Electronics, Movies, and Music. A red 'A' icon is visible in the Region column for the Electronics row.

Category	Year	Revenue	Profit
Books	2004	\$20,152.00	\$5,360
Electronics	2004	\$3,447,445.00	\$939,754
Movies	2004	\$858,893.00	\$212,731
Music	2004	\$861,154.00	\$94,746

Figure 8.7: Treating hierarchy levels as dimensions in MicroStrategy OLAP Services

categories are not shown, there is no distinction between parallel and alternative hierarchies and no means of expressing generalization/specialization relationships, partial and non-strict roll-ups. Besides, nesting dimensions and measures within their cubes disables parallel exploration of multiple cubes (drill-across) and asymmetric treatment of dimensions and measures disallows using a dimension attribute as a measure and vice versa (push and pull). The above mentioned limitations of the existing OLAP tools motivated us to develop a novel OLAP cube navigation paradigm, capable of adequately handling the entire set of static and dynamic features of the extended multidimensional model, proposed in Chapters 3 to 5.

8.2.2 Enhancing Data Navigation through Scheme Awareness

We have come to realize that the weaknesses of the instance-based and the category-based navigation are primarily due to trading semantic richness off for simplicity: decomposition of a multidimensional scheme into a set of unrelated categories results in a loss of all types of relationships between categories and, consequently, in the inability to support semantic properties provided by those relationships. We propose a *scheme-based* navigation approach, which preserves the multidimensional semantics by accurately reflecting the conceptual data scheme in the navigation hierarchy. In a nutshell, we pursue a clear distinction between the scheme and the instance of a dimension: a dimension is represented by its scheme, i.e., its category nodes nested according to parent-child relationships between them. Expanding a category node reveals solely its contained child categories, whereas the members of a particular category are displayed on-demand.

Figure 8.8 demonstrates the main idea of moving from the instance-based navigation (a) to the scheme-based one (b) at the example of a hierarchical dimension Nationality. Expansion of the top-level node in Figure 8.8b reveals the entire hierarchy scheme, thus enabling the user to drill through to any desired granularity level. The instance is retrieved by clicking on the category's data-view button as shown in Figure 8.8c for category Subcontinents. By switching to the instance view, the user can explore the data hierarchy in the same way as in the instance-based navigation.

HANDLING HIERARCHIES

The navigation structure of a dimension is further refined by distinguishing between the nodes of type *Dimension*, *Hierarchy*, *Level*, and *Attribute*, whose meanings correspond to their counterparts in the conceptual and the metadata model:

- ◆ *Dimension* node represents the abstract top category of the dimension and “wraps” the entire dimension scheme into a single navigation node at the top level.
- ◆ *Hierarchy* node “wraps” each of the multiple hierarchies into a single containing node (dimension in a dimension), thus allowing to specify alternative and specialization hierarchies.
- ◆ *Level* node represents a single non-abstract category in a hierarchy.
- ◆ *Attribute* node corresponds to a single attribute in a category with multiple attributes.

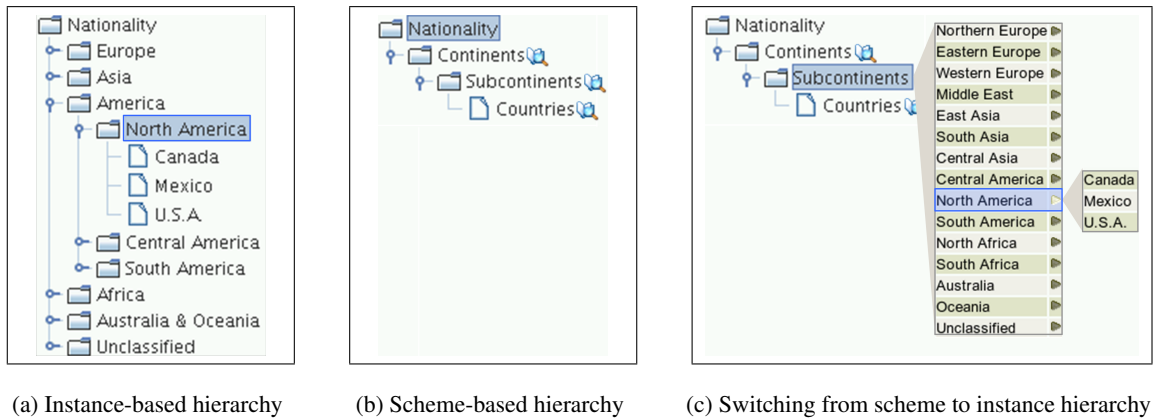


Figure 8.8: Instance-based vs. scheme-based navigation for a hierarchical dimension

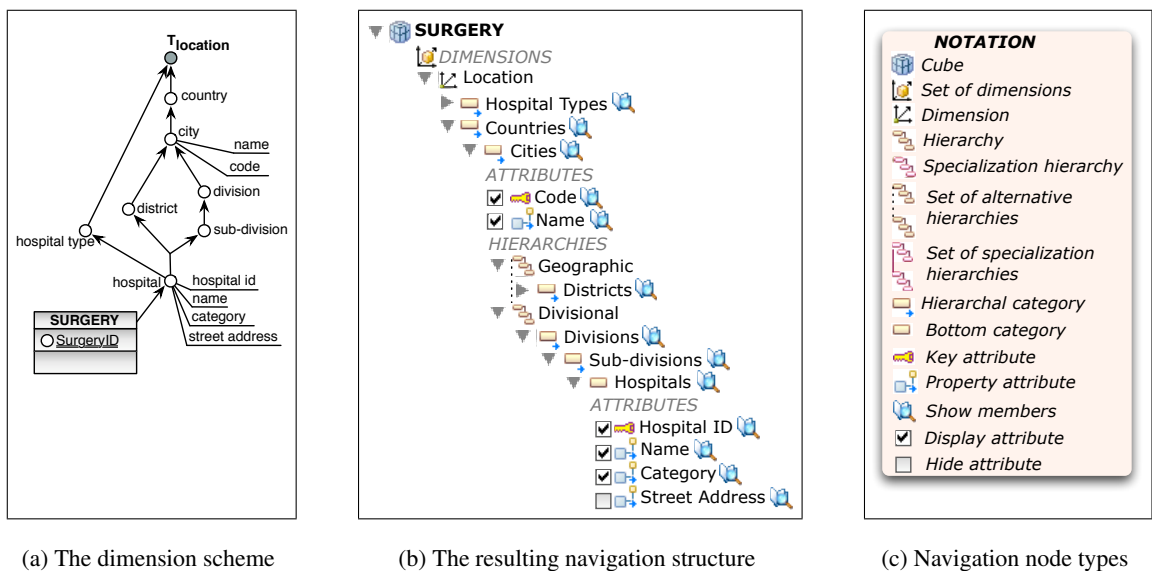


Figure 8.9: Obtaining a scheme-based navigation hierarchy of a dimension

To understand the roles of different node types in the navigation, let us consider a sample dimension scheme location in Figure 8.9a and its resulting navigation scheme in Figure 8.9b. In addition to the nodes mentioned above, the navigation also uses structural nodes of type *HIERARCHIES* and *ATTRIBUTES* to separate the set of the category's hierarchical relationships from the set of its properties (as in category city). The *ATTRIBUTES* node can be omitted in a category consisting of a single attribute and *HIERARCHIES* node is only necessary in the existence of the *ATTRIBUTES* node. Figure 8.9c provides an overview of all node types that may occur in a dimensional navigation structure.

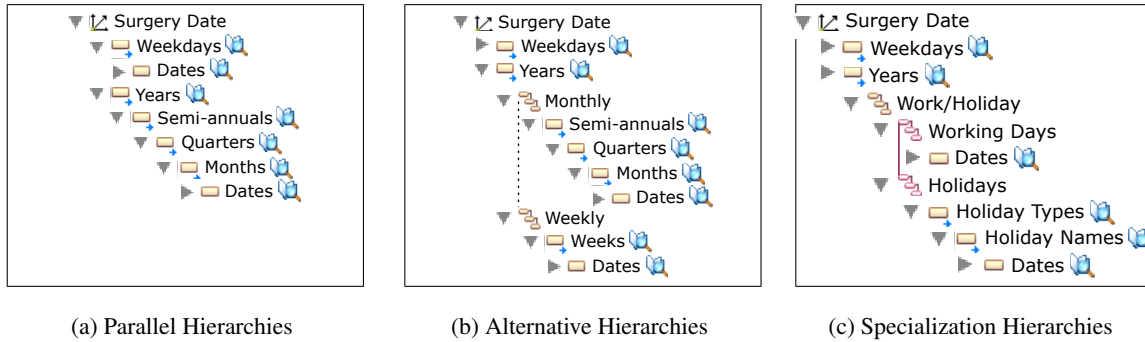


Figure 8.10: Examples of navigating in multiple hierarchies

Navigation nodes of type *Hierarchy* are employed to enable adequate handling of multiple alternative and specialization hierarchies. Diverging hierarchy paths underneath a category node indicate parallel, alternative, or specialization hierarchies, as exemplified by the respective variants of dimension Surgery Date depicted in Figure 8.10. Each type leads to specific aggregation constraints and, therefore, has to be treated accordingly in the navigation hierarchy:

- ◆ *Parallel hierarchies* are independent non-exclusive aggregation paths that may be explored in combination. Therefore, these hierarchies can be represented as simple nestings of category nodes, not topped by a *Hierarchy* node and not linked to one another in any way. As an example, consider the variant of Surgery Date navigation in Figure 8.10a: the bottom category Dates rolls up to Weekdays, on the one hand, and to Months, Quarters, Semi-annuals, and Years, on the other hand. Both paths are represented by their respective top levels Weekdays and Years, nested in the dimension node.
- ◆ *Alternative hierarchies* are incompatible aggregation paths: once a category of a hierarchy has been used as a grouping criterion, all hierarchies alternative to the former should be disabled. The variant of Surgery Date dimension in Figure 8.10b contains alternative aggregation paths of Dates: grouping by Months, Quarters, and Semi-annuals may not be combined with grouping by Weeks. Both paths are represented by the respective *Hierarchy* nodes, linked with a dashed line, within Years, where both paths converge. Once a category from such a hierarchy is selected as a grouping condition, all hierarchies alternative to the former become “undraggable” and can only be used for filtering.
- ◆ *Specialization hierarchies* are exclusive (each member of the superclass belong to just one of the specializations) but compatible (specializations may be explored in parallel) for parallel exploration. In the navigation, a set specialization paths is represented by a set of linked *Specialization Hierarchy* nodes, as can be seen in Figure 8.10c at the example of the generalized hierarchy Work/Holiday and its specializations Working Days and Holidays. The resulting representation is rather similar to that of alternative hierarchies, however, there are significant differences in query options. First, specialization paths do not “lock” one another allowing the user to drill into different subclass hierarchies within the same query. Besides, the navigation element of type *Specialization Hierarchy* is not just an abstract “wrapper”, like the nodes of type *Cube*, *Dimension*, or *Hierarchy*, but it actually represents the local root category and its implied member “all” of the underlying subclass hierarchy. Therefore, we make such nodes selectable as grouping criteria. For example, dropping the hierarchy nodes Working Days and Holidays into the visualization would split the parent aggregate value into two subaggregates, one for all date values of type working day and the other one for all date values of type holiday.

HANDLING MEASURES

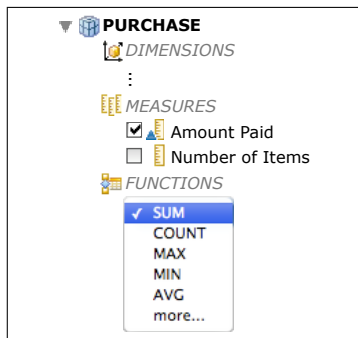
Representation of measures in the navigation is generally less challenging as the former are non-hierarchical each consisting of a single data field. So, basically, a set of cube's measures can be represented as a plain enumeration. Issues relevant in this context are whether the navigation should distinguish between basic and computed measures, how it should go about the aggregability constraints and how it could implement definition of new measures and enable interchangeability of measure and dimension roles in a query.

In the current practice of OLAP interfaces, there exist two major approaches to representing measures, namely as an attribute of the fact table or as computation formula. As an example, consider fact scheme *PURCHASE* with measure attributes Number of Items and Amount Paid. The simplest navigation alternative would be to enumerate the above two attributes in the *MEASURES* area of the cube and to prompt the user select an aggregate function from the *FUNCTIONS* area of the navigation. By default, the SUM function is assumed. Figure 8.11a shows the resulting navigation fragment (a measure icon with a blue triangle marks the default measure of the cube). In such a setting, the user is free to apply any function to any measure field, i.e., there is no support for aggregation semantics.

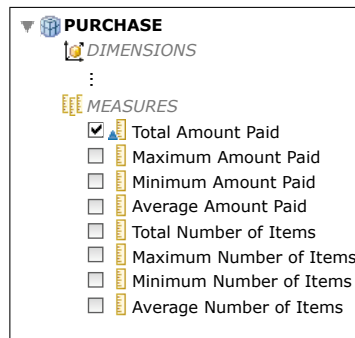
Another extreme is to explicitly display each valid aggregation option specified by the metadata. This is done by representing each allowed computation formula, i.e., a combination of a measure attribute and an aggregate function, as a measure in its own right, as shown in Figure 8.11b. Both measure attributes Number of Items and Amount Paid are allowed to be aggregated using SUM, MAX, MIN, and AVG, resulting in eight valid measure entries in the navigation. With such a setting, the user's choice is restricted to the aggregation options specified through the metadata.

Apparently, both approaches have their pros and cons. The first option produces a very compact view, but it is unable to assign various aggregate functions to various measure attributes within the same query. The second approach prevents invalid aggregation semantics, however, at the expense of generating very long lists of measures and inflexibility to support user-defined measures. We propose an alternative representation, aimed at overcoming the disadvantages of the above two approaches and extending the functionality by the following means (see Figure 8.11c for the corresponding example):

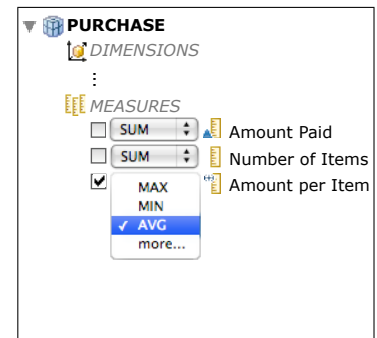
- ◆ Measures are represented as attributes prepended by a drop-down selection of applicable aggregate functions. Thereby, the navigation remains compact allowing multiple measures to be selected each with its own function.



(a) Separating measures attributes from aggregate functions



(b) Measures with built-in aggregate functions



(c) Measures with built-in aggregation semantics

Figure 8.11: Different approaches to representing measures

- ◆ Aggregability constraints are preserved by adjusting each measure's function selection. As an example, consider the derived measure field Amount per Item of type *value-per-unit* and its functions set reflecting the measure's non-additivity in Figure 8.11c.
- ◆ The option “more . . .” in the function list enables the user to select another aggregate function, besides the explicitly allowed one.
- ◆ Derived measure fields are shown by marking their measure icons with ^(x). Amount per Item is an example of a derived field, computed as Amount Paid / Number of Items.
- ◆ In addition to the measure fields (primary or derived ones), specified through the metadata, the user should be able to derive further measures from the existing ones ad hoc. This is done via the measure definition wizard – a feature provided by any advanced OLAP tool.
- ◆ Each cube specifies a measure attribute to be used by default and each measure specifies the default aggregate function to guarantee proper query instantiation. In case of a non-measurable scheme, a default measure entry COUNT(*) (i.e., mere counting of qualifying facts) is added to the navigation.

With the proposed semantically rich scheme-based navigation framework, we are able to support advanced scheme-transforming OLAP operators and other dynamic properties as described in the next section.

8.2.3 Dynamic Properties of the Navigation Scheme

One of the desirable advanced properties of the navigation scheme as the query specification interface is supporting ad hoc manipulation of multidimensional schemes in order to obtain new views of the available data. Such manipulations are concerned with changing the hierarchical organization of dimension, interchanging the roles of dimensions and measures in a cube, and creating new facts by joining multiple related cubes.

The initial state of the navigation corresponds to the metadata description of the displayed sources. In the course of interaction, the users may manipulate the original multidimensional schemes to adjust them to specific analysis needs. We distinguish between *persistent* and *ad hoc* data manipulation. Persistent changes to the data (e.g., materialization of derived elements or definition of new data views) are done via cube designer tools, which propagate the required operations to the backend and to the metadata layer, whereas ad hoc changes refer to temporary transformations aimed at supporting particular tasks or queries. We propose that the latter type of operations should be supported by the navigation interface of OLAP frontends: the state of the navigation may undergo a series of transformations during the session but gets reset once the session is closed.

The first kind of ad hoc transformations is concerned with allowing the user to change the aggregation hierarchy of a specific dimension by adding or removing hierarchy levels or adjusting the existing levels and relationships between the elements. Since this functionality is readily supported by the leading OLAP vendors by means of providing corresponding dimension redesign wizards, we skip its further detailing and rather focus on insufficiently supported features, such as PUSH, PULL, and DRILL-ACROSS operations.

ENABLING PUSH AND PULL

OLAP operators PUSH and PULL allow the user to define a measure from a dimension category and to use a measure field as a dimension, respectively. A combination of these operators providing a foundation for interchanging measure and dimension roles in a scheme. Since the navigation scheme of a cube is subdivided into *DIMENSIONS* and *MEASURES* sections, an intuitive way of supporting these operators is to allow a drag&drop of measure and category nodes from one section to the other.

Dropping a measure attribute into the *DIMENSIONS* section is interpreted as a PULL operation and triggers a dimension specification wizard. Semantic implication of “pulling” a measure is to treat its values as dimensional characteristics of other measure attributes in the fact scheme. For example, converting measure Number of Items in fact scheme PURCHASE into a dimension allows to use the former as a filtering condition and to aggregate the other two measures Amount Paid and Amount per Item by Number of Items.

Considering that measure fields are typically of a numeric type, possibly with high precision and a huge number of distinct values, their instances need to be adjusted to be usable as dimensions. This problem, known as “continuous dimensions” in data warehousing research, is solved by employing various discretization techniques to reduce the size of the dimension’s instance [162]. Discretization is performed by lowering the precision (e.g., rounding) of the values or replacing each set of values by a single (e.g., median) value or a range specification. The dimension specification wizard prompts the user to specify the desired discretization method in the same similar fashion as the existing cube designer tools treat continuous dimensions. The resulting ad hoc dimension is non-hierarchical. If desired, hierarchy levels (INSERT LEVEL operator) can be added using the dimension redesign wizard. The navigation scheme of the affected cube reflects the changes by removing the converted attribute from *MEASURES* and making it appear in *DIMENSIONS*.

To ensure the validity of the transformed fact scheme, we reduce the set of PULL-able measures to the simple ones, i.e., consisting of a single data field. Members of the new dimension correspond to those of the underlying measure attribute, thus, ensuring the validity of the original fact entries. Conversion of a measure with no data field in the fact table (e.g., COUNT(*)) or with a complex computation formula is not trivial and remains a subject for further investigation.

The reverse act of dropping a dimension category into the *MEASURES* section is interpreted PUSH and triggers the measure specification wizard. The semantics of “pushing” is to enable aggregation of the category’s values along other dimensions of the fact scheme. This operation is indispensable in non-measurable fact schemes and was used in sample queries presented in Section 6.3. The key attribute of the selected category turns into the input field of the new measure, prompting the user to specify the set of applicable aggregate functions. A non-numeric input attribute results in a non-aggregable measure. Therefore, the set of selectable functions is automatically reduced to COUNT and COUNT DISTINCT (duplicate-free counting).

The major challenge of performing a PUSH is to adjust the granularity of the affected cube. Especially when converting a non-bottom category into a measure, the fact entries must be consolidated accordingly. For example, if category city in dimension Location of SURGERY (see Figure 8.9a) is turned into a measure (e.g., to query the number of cities in which certain surgery types took place), all levels below city should be removed from the aggregation path and hospital values in fact entries should be replaced by the respective city values. Figure 8.12 shows a view of the measure specification wizard used for transforming category Cities into a measure. The navigation scheme of the affected cube reflects the changes by removing the converted category as well as the path below it from *DIMENSIONS* and making the new measure appear in *MEASURES*.

New Measure	
Measure name	Cities
Input field	Code (Cities)
Allow functions	SUM COUNT COUNT DISTINCT MAX MIN AVG more...
Cancel	OK

Figure 8.12: Performing a PUSH operation

Let us consider a complete example of swapping a measure with a dimension in a variant of fact scheme SURGERY depicted in Figure 8.13a. The transformation is aimed at enabling the analysis of surgery duration in each cost category. In the first step, category Full Hours in Duration is converted into a measure, as shown in Figure 8.13b, resulting in disappearance of the entire dimension Duration since Full Hours was its coarsest level. In the subsequent PULL step, measure attribute Costs is converted into a dimension. To enable aggregation by cost categories, hierarchy level Cost Ranges is added on top of the bottom category Exact Costs. The resulting cube navigation is shown in Figure 8.13c.

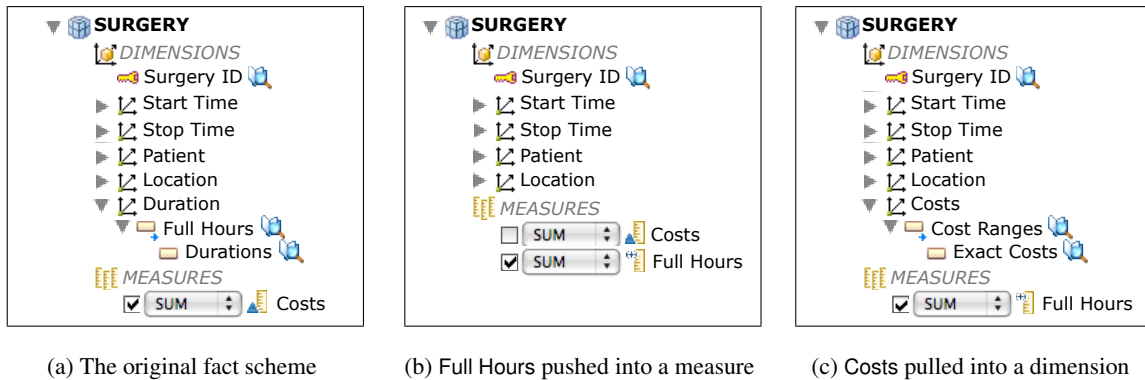


Figure 8.13: Example of a combined usage of PUSH and PULL

ENABLING DRILL-ACROSS

Drilling across, also known as a multicube join, is an advanced OLAP operation for parallel exploration of multiple related cubes in order to compare their measures or even to combine the latter into a new measure. Intuitively, a pair of measures are comparable, if they have compatible sets of dimensional characteristics. Therefore, to qualify for a DRILL-ACROSS, participating fact schemes must be related, i.e., must share at least one category. Each of the input cubes is made compatible for multicube join by being rolled-up to the subset of dimensional characteristics, common to all participating cubes. Some of the existing OLAP tools support drilling across by providing a cube definition wizard for a step-by-step construction of the desired multicube view. To avoid a tedious cube definition procedure, we suggest that the drill-across functionality should be integrated into the navigation scheme in the form of a multicube navigation hierarchy.

A rather simple solution is to “unnest” the dimensions and the measures from their containing cube nodes into a common navigation hierarchy, as illustrated in Figure 8.14. The standard view of the navigation scheme in Figure 8.14a reveals the structure of cubes SURGERY and HOSPITALIZATION. Both fact schemes are strongly related since their dimension sets overlap for the most part. The unnested (“galaxy”) navigation view is obtained by placing the dimensions and the measures of both cubes into common *DIMENSIONS* and *MEASURES* sections, respectively, as shown in Figures 8.14b and 8.14c, with conform dimensions represented in a non-redundant fashion. The resulting navigation structure can be used for querying the measures both within any of the cubes and across cubes. Once a measure is selected, the *DIMENSIONS* section adjusts itself by disabling (fading out) the dimensions invalid in that measure’s context, as shown in Figure 8.14b. Similarly, selecting measures from both cubes results in displaying only the subset of dimensions valid for all selected measures. Figure 8.14c presents a drill-across view for comparing average surgery costs with the respective hospitalization expenses.

The above galaxy view provides a simple drill-across interface, however, it is restricted to admitting only fully conforming dimensions as multicube join axes. Obviously, a more intelligent solution is needed for drilling across along partially related dimension schemes. Let us suppose that temporal dimensions in HOSPITALIZATION are called Check-In and Check-Out and have Date category as the bottom grain. The corresponding dimensions Start Time and Stop Time in SURGERY are fine-grained into Timestamp. A DRILL-ACROSS is still possible, if HOSPITALIZATION facts are rolled up to the date level. Similarly, surgery location is Operating Theater, whereas hospitalization location is Room, but both hierarchies converge in Hospital.

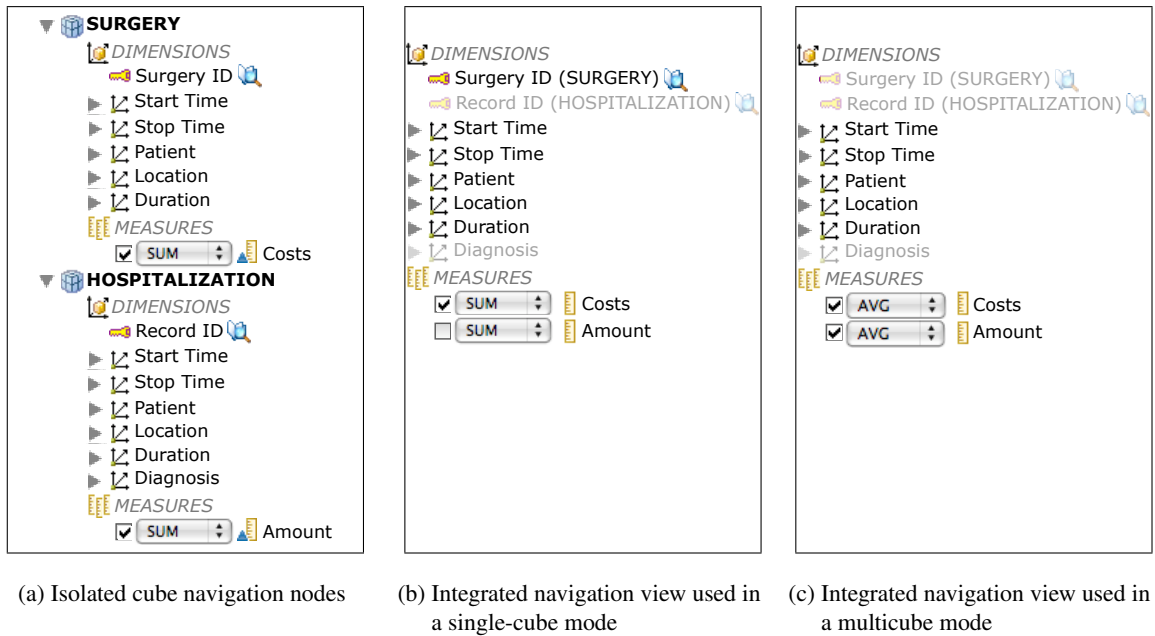


Figure 8.14: “Galaxy” view of the navigation for a pair of related cubes

We propose to enhance the galaxy view of the navigation by prompting the user to specify, what dimensions should be matched, and by merging each pair of matching dimensions into a common *super-dimension*. Figure 8.15 visualizes the essence of our approach based on the above example, with isolated navigation schemes of the two cubes in (a) and the resulting joined navigation scheme in (b). Notice that the unified view of dimension hierarchy schemes is an accurate reflection of the underlying semantic galaxy scheme.

The proposed enhanced multicube navigation scheme is obtained as follows:

1. The user is prompted to specify the set of cubes for drilling across. The system analyzes the validity of the selection, generates the definition of the virtual cube for performing the requested multicube join, and constructs the corresponding joined navigation scheme. Basically, there is no limitation on the number of cubes to be explored in parallel, but in the practice, this number rarely exceeds two.
2. In case of ambiguities (e.g., differing names of related elements or multiple possibilities of dimension matching), the user must specify which dimensions should be merged and how. Back to our example, the user chose to merge Start Time with Check-In and Stop Time with Check-Out and confirmed that dimensions Location and Duration in both cubes should be matched in spite of the differing bottom grain.
3. Each of the selected cubes is assigned a unique color icon (see the top of Figure 8.15b). Belonging of a dimension, a hierarchy level, an attribute, or a measure to any of the cubes is then specified by marking it with the respective color mark. Dimensional nodes at the top display the marks of all contained elements. Similarly, computed measures display the marks of all cubes, whose measures are involved into the computation. As an example, consider the derived measure Total Charge, defined as Costs + Amount. We denote such measures *multicube measures*.

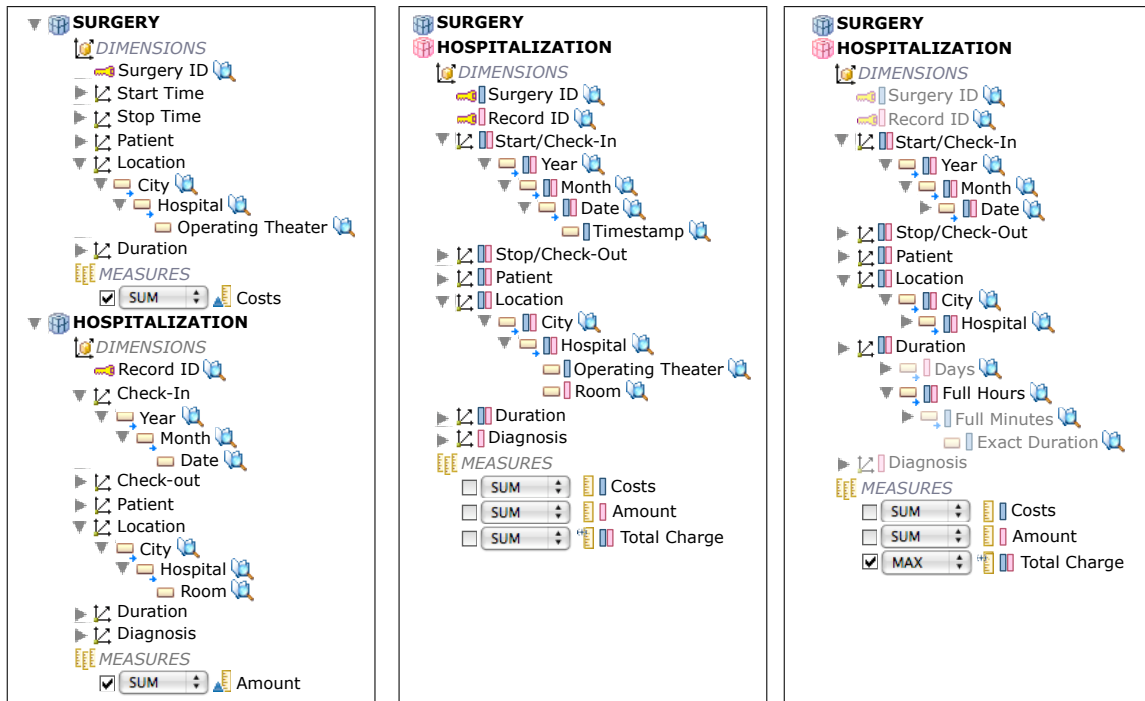


Figure 8.15: Multicube navigation scheme with enhanced semantics

The “drill-across” navigation mode is triggered by selecting either a set of measures belonging to different cubes or a multicube measure. The latter case is shown in Figure 8.15c. The use of color marks facilitates visual recognition of valid drill-across paths: only those hierarchy levels containing each of the marks present in the selected measure (or set of measures) may be used as drill-down axes, as shown in Figure 8.15c by a “faded” display of all disqualified paths. Notice, however, that disqualified categories are prohibited only for drilling down, but they still may be used for filtering. For example, category Days in Duration can be used to exclude HOSPITALIZATION facts of too short duration from the computation of Total Charge.

Implementation of scheme-transforming operators is done by generating a temporary metadata description for the transformed cube, rebuilding the navigation according to the new metadata, and defining an appropriate virtual cube at the backend. In case of a complex multicube join, constructing a virtual cube “on-the-fly” may cause unacceptable performance degradation of the backend system. One solution to solving this problem is to materialize the defined multicube, i.e., the correspondingly preaggregated input facts and their join. In that case, however, the functionality of filtering along disqualified levels becomes unavailable as the respective characteristics are not present in the preaggregated view.

8.2.4 OLAP Operators and their Implementation Options

Analytical queries manipulate the input data by applying of OLAP operators, such as drill-down, roll-up, slice&dice, ranking, and pivoting, to name the major ones. These operators enable the user to abstract “raw”

numbers into a desired perspective by taking a cube or a set of cubes as an input and outputting a new cube. In visual frontends, in which queries are specified implicitly, OLAP operators are “encoded” into interaction events performed either on the navigation scheme or on the visual presentation of the retrieved data subset.

To account for various user preferences, most of the operators are provided redundantly, e.g., via the navigation, as a menu option, an icon, or a widget, and via direct interaction with the visualization. In the previous subsection we already touched upon a number of operations, such as drill-down, filter, push, pull, and drill-across, and described their implementation in the navigation interface. Table 8.1 enumerates the set of supported OLAP operations (for detailed descriptions of the provided functionality refer to Section 2.2.2) and describes possibilities of their implementation as navigation events or other interaction techniques. The next section will provide further insights into interactive visualization techniques for OLAP.

Table 8.1: OLAP operations and their implementation in a visual frontend

Operation	Navigation	Interaction
ROLL-UP	<i>Undo</i> of a previously performed DRILL-DOWN	<i>Zoom-out</i> within a dimension axis, <i>collapse</i> the element(s) of interest, or <i>remove</i> the category from its visual mapping
PROJECT	See ROLL-UP	<i>Remove</i> the whole dimension from its visual mapping
DRILL-DOWN	<i>Dropping</i> the category into a visual mapping	<i>Zoom-in</i> within a dimension axis or <i>expand</i> the element(s) of interest
DRILL-THROUGH		<i>Menu option</i> , <i>icon</i> , or a <i>popup menu</i>
DRILL-WITHIN	See DRILL-DOWN	A <i>popup menu</i>
SLICE	<i>Selecting</i> a value in a category to serve as a filter	<i>Trim</i> the view to the area of interest
DICE	<i>Selecting</i> a set of values in a category to be filtered out	<i>Deleting</i> or <i>collapsing</i> corresponding areas in the visualization
SELECT	<i>Selecting</i> a set of values in a category to serve as a filter	<i>Trim</i> the view to the area of interest, use of <i>sliders</i> or <i>panning windows</i> for range selections
FILTER	See DICE	Configuration via a <i>filter</i> menu, use of <i>sliders</i> for range selections
CONDITIONAL HIGHLIGHTING		See FILTER
RANKING	Including a ranking function into the measure’s definition	Layout-specific options, a <i>filter</i> menu or a <i>slider</i> on the measure field
SWITCH		<i>Sort</i> menu or <i>drag&drop</i> of visual elements into desired positions
PIVOT		<i>Drag&drop</i> of data fields to new visual mappings
DRILL-ACROSS	<i>Multicube</i> navigation scheme	
PUSH	<i>Dropping</i> the category into the measure section	
PULL	<i>Dropping</i> the measure into the dimension section	
INSERT LEVEL	Dimension redesign wizard	
DELETE LEVEL	Dimension redesign wizard	

Apart from the intuitive and purely visual query specification, OLAP frontends enable incremental generation of the desired view, i.e., without the loss of context. For example, one can drill down into just a subset of the retrieved aggregates, resulting in the display of multiple granularities in the same view. Thereby, the complexity of the underlying backend operations remains transparent to the user.

8.3 Hierarchical Visualization Techniques for OLAP

8.3.1 Salient Characteristics of Visual Interaction Patterns

Confronted with having to evaluate thousands of values returned by an OLAP query, users tend to focus on few outstanding values and discard the remaining ones or run another query to calculate the next-level aggregates (drill-down or roll-up) in order to obtain a mental image of the data. While this strategy can be successfully applied for reporting tasks, exploratory analysis is driven forward by finding information hidden in the data via ad hoc queries. In such scenarios, abstract visual representations (i.e., overviews) can be of significant help in supporting this process of comprehending characteristics of large data sets.

As data sets for analysis are huge, coarsely grained aggregates play the role of orientation hints, or “direction signs”, in exploratory queries. The analyst wants to see measures of interest at different aggregation levels and compare the aggregates within the same level as well as across levels. As an example, one might want to compare expenditures of departments with each other, but a comparison of a small department with a project team of approximately the same size would also make sense.

Multiscale visualizations are effective techniques for facilitating the exploration process because they change the visual representation to show the data at different levels of abstraction. At a high level, it is heavily aggregated in order to display a large amount of data in a compact way. As the user zooms in, data density decreases allowing to show more detailed representations of individual data points [168]. Preservation of the overview throughout the course of interaction (zooming and panning) becomes very important as otherwise the user might easily lose the context in the process of drilling down.

Visual scalability, i.e., the ability of a visualization to meaningfully display a large number of data items, is another challenge. Display space becomes a scarce resource, which in turn leads to the development of space-filling visualization techniques that efficiently utilize the available display area.

While analytical queries aggregate over detailed facts, visual analysis evolves in the inverse direction, i.e., proceeding from a few coarsely grained aggregates towards more detail by drilling down along dimensions of interest. This top-down direction makes it easier for the analysis to retain an overview and to get hints on how to proceed, e.g., whether to apply a filtering or a drill-down step. The top-down navigation hierarchy, described in the previous section, accords nicely with this query pattern by nesting deeper hierarchy levels inside their coarser-grained predecessor levels.

Considering the prevailing *iterative disaggregation* pattern in visual exploration and the fact that aggregates obtained at earlier stages often remain useful for comparisons with their subaggregates or other values, hierarchical visualization techniques suggest themselves as perfect candidates for exploring OLAP data.

8.3.2 Decomposition Tree

A structure that captures the aggregates produced by a series of drill-down or roll-up operations is denoted a *decomposition tree*: the grand total value of the measure is placed into the root node and the constituent sub-aggregates, obtained via a drill-down step, are represented as child nodes of the containing aggregate.

As an example, let us consider iterative decomposition of student numbers, measured as the number of

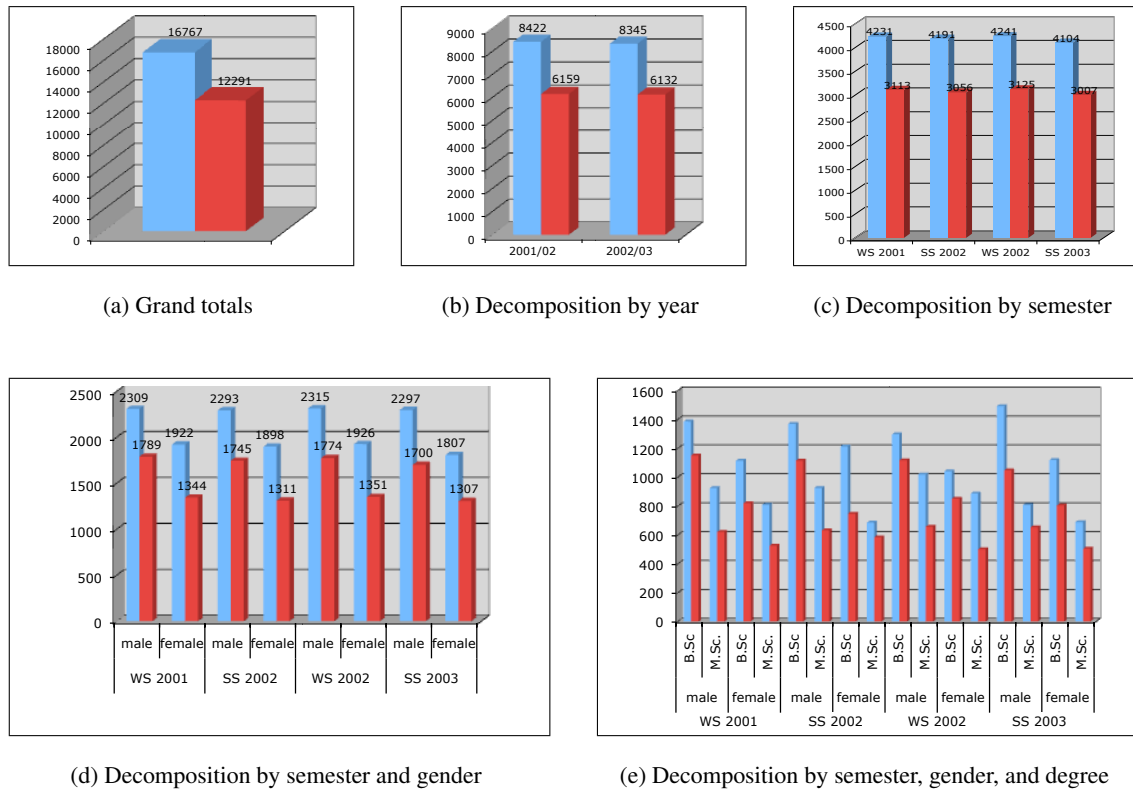


Figure 8.16: Mapping a sequence of decomposition steps to a set of bar-charts

enrollment cases and the number of persons¹ by year, semester, gender, and degree. Figure 8.16 shows the output of each decomposition step organized into a bar-chart with number of enrollment cases and number of persons mapped to blue and red bars, respectively. Since the bar-chart layout provides a one-dimensional view, each decompositions results in splitting the charts along the horizontal axis.

With every decomposition step, the chart view gets updated resulting in the loss of previously computed aggregates. Replacing a sequence of charts with a decomposition tree view connects the aggregates computed at each step into a hierarchy as shown in Figure 8.17. Such a hierarchical presentation by its very nature has an advantage of supporting any number of drill-down categories. Since there are as many disaggregation operations possible within a cube as the total number of mutually non-exclusive hierarchy levels in all its dimensions, and since the order of combining orthogonal grouping criteria can be arbitrary, OLAP cubes can be seen as containers of a multitude of decomposition hierarchies.

The first commercial implementation of the decomposition tree as an interactive visualization technique for OLAP was provided by ProClarity [141]. Figure 8.18a shows the results of decomposing the total sales in Category “Computer Hardware” by Sub Category, Region, and Name. Child nodes display the next level of detail as absolute value and percentage of the total. The tree can be expanded only one node at a time, is limited to displaying a single measure attribute and has no visual formatting of the node values. Besides, it is

¹The number of enrollment cases tends to be slightly higher than the number of persons since a person may be enrolled into multiple study programs.

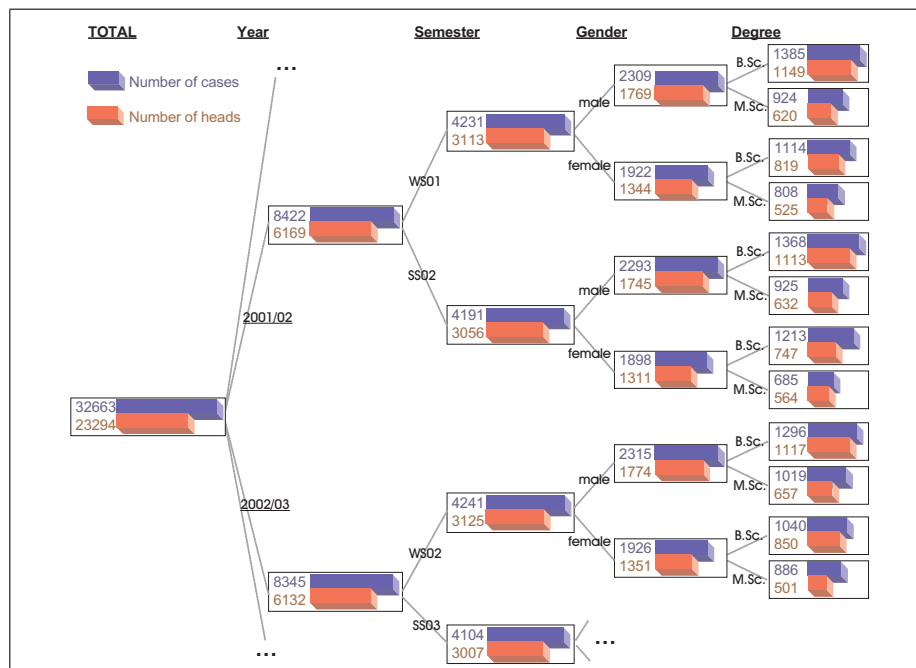
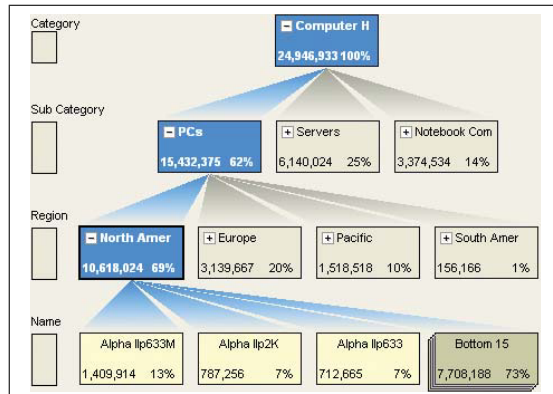
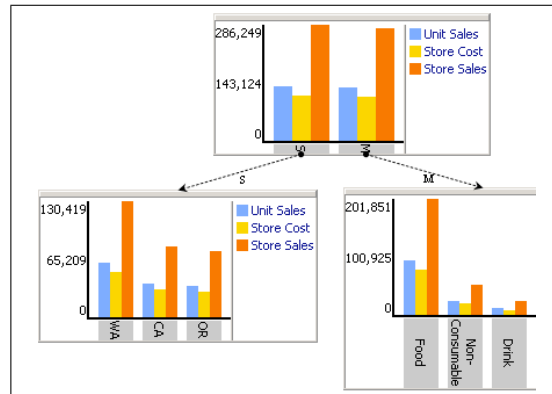


Figure 8.17: Arranging a sequence of decomposition steps into a decomposition tree



(a) ProClarity's Decomposition Tree



(b) Bar-Chart Tree in Report Portal

Figure 8.18: Commercial decomposition tree techniques

rather wasteful in terms of display utilization and is thus infeasible for exploring large data volumes.

Further enhancements of hierarchical decomposition techniques for OLAP are found in the OLAP web client Report Portal 2.1 released by XMLA Consulting in 2005 [188]. Report Portal offers graphical decomposition tree techniques, such as Bar-Chart Tree and Pie-Chart Tree, which enhance the presentation by

arrange the values inside a node into a chart. The bar-chart variant even supports multiple measures and negative value domains. The interaction approach is similar to that of ProClarity, i.e., a single aggregate can be expanded at a time. However, drill-down steps need not be aligned into categories, thus allowing to expand each of the “sibling” aggregates along a different path. Figure 8.18b shows an example of such a heterogeneous Bar-Chart Tree, in which the the left-hand set of measure values is decomposed by state and the right one by product category.

8.3.3 Enhanced Decomposition Trees

Aware of the limited functionality and poor presentation options of the existing decomposition tree interfaces, we developed a class of *Enhanced Decomposition Tree*. The introduced enhancements are manifold and target various optimization criteria, such as simplicity, type of measure and aggregate function, the size and the dimensionality of the data set, etc. The entire set of the proposed enhancements can be subdivided in *i*) generation/interaction and *ii*) layout options, discussed in the following subsections.

DECOMPOSITION STRATEGIES

Existing decomposition tree techniques enable drilling down into single aggregates to see their subaggregates, as shown in the tree instances depicted in Figure 8.18a, in which just a single node per level is expanded. To change the grain of the whole tree, the user has to expand every single node at the bottom level – obviously a too tedious way of performing a single drill-down query. However, this node-per-node decomposition strategy is pursued by the vendors deliberately since the provided tree layouts are of very limited scalability and are thus inadequate for visualizing complete decomposition hierarchies. In our framework, this exploration deficiency is overcome by providing layouts of better scalability (described in the next subsection) and enabling various drill-down modes, such as *i*) the *value-wise*, *ii*) the *node-wise*², and *iii*) the *level-wise* one.

As an example of a hierarchical decomposition, let us consider that of student enrollments along section, faculty, and department, mapped to a bar-chart tree shown in Figure 8.19: (a) shows the results of two value-wise drill-down steps, (b) shows a variant of decomposing all values within one node at each level, and (c) captures two complete (i.e., level-wise) drill-down operations. The user is free to choose the desired mode and can arbitrarily switch from one mode to another during the exploration.

Value-wise and the node-wise decomposition steps require specification of the context element, i.e., a value or a node to be split. The context can be specified either explicitly by marking the elements of interest with mouse clicks prior to performing the respective drill-down step or implicitly by dropping the drill-down category’s node into the element’s area.

Contrary to the chart tree techniques of Report Portal, Enhanced Decomposition Trees preserve the concept of uniform tree levels, i.e., all aggregates within the same level are obtained via the same sequence of drill-down steps. This property is crucial for enabling the above multi-mode decomposition within the same visualization as well as interactive switching between different layouts.

Another empowerment of the decomposition tree technique is achieved by introducing different types of drill-down steps, triggerable by the same drag&drop action:

- ◆ **Nested split (split-into)** refers to the specification of the drill-down step inside the nodes and is valid only in hybrid tree layouts that arrange multiple values into a single node by means of nested visualizations. The resulting intra-node granularity in the tree can be refined by a subsequent nested split along any category, which lies deeper in the hierarchy path of the current split.

²Differentiation between the value-wise and the node-wise modes is only made in those layouts, in which a node contains a set of aggregates rather than a single one.

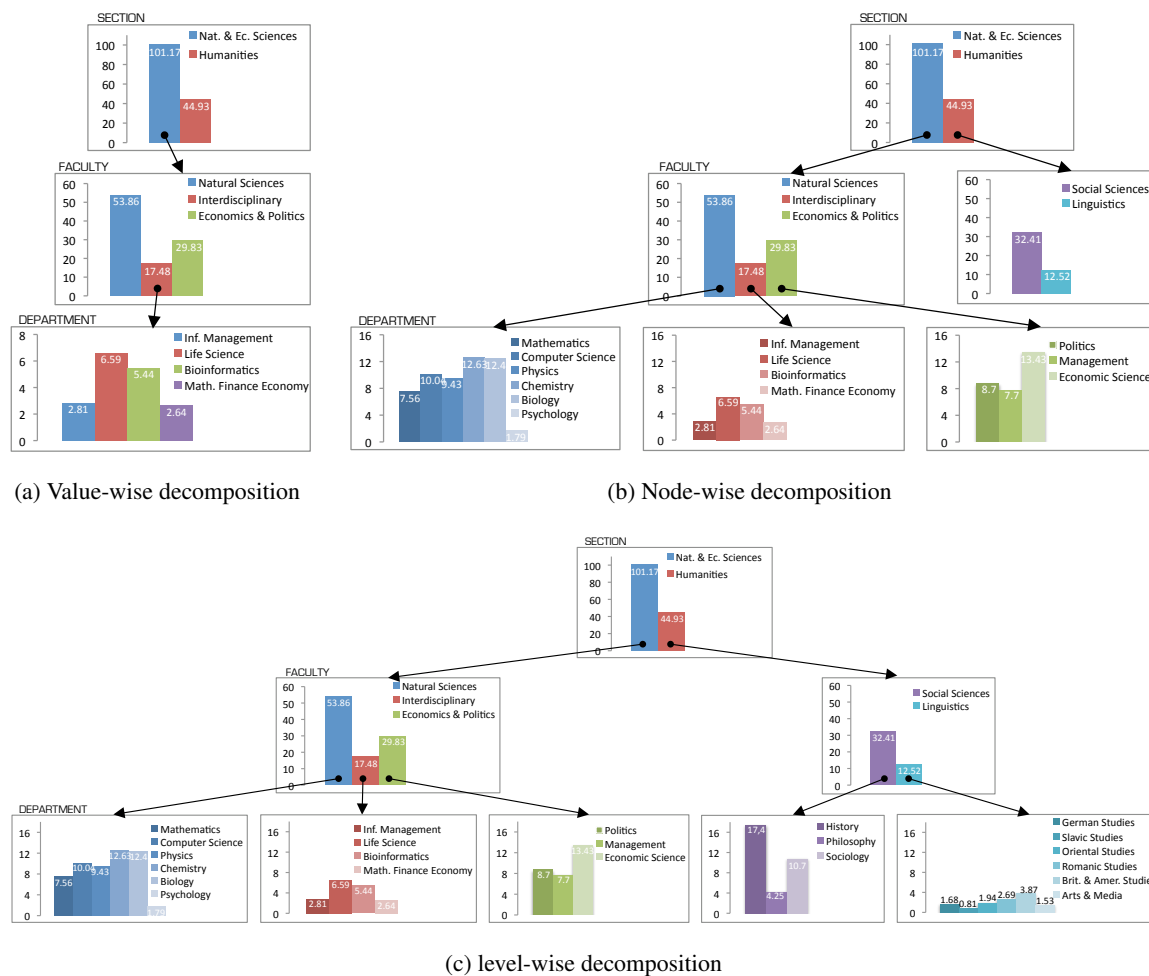


Figure 8.19: Using different decomposition modes with a bar-chart tree

- ◆ **Inner split (split-across)** decomposes each of the aggregate values in the parent node into subaggregates along a specified split category. As a result, each of the child nodes has the same number of aggregate values as the parent node, however, at a finer grain.
- ◆ **Outer split (split-along)** maps each of the the aggregate values in the parent node to a separate child node and uses the split category to perform a nested-split in those new nodes. As a result, the number of aggregates is different from that in the parent node.

Figure 8.20 illustrates the above split types. A split-into by faculty generates a set of aggregates for the bar-chart presentation in the root node. A subsequent decomposition by department is performed as a split-along: each child nodes corresponds to an aggregate in the parent node broken down by department. A split-across along the category cost class, which consists of just two member values *External* and *Internal*, partitions each parent-node set into two subsets, one for each member value: each aggregate in a child node's refers to a different aggregate value in the parent node.

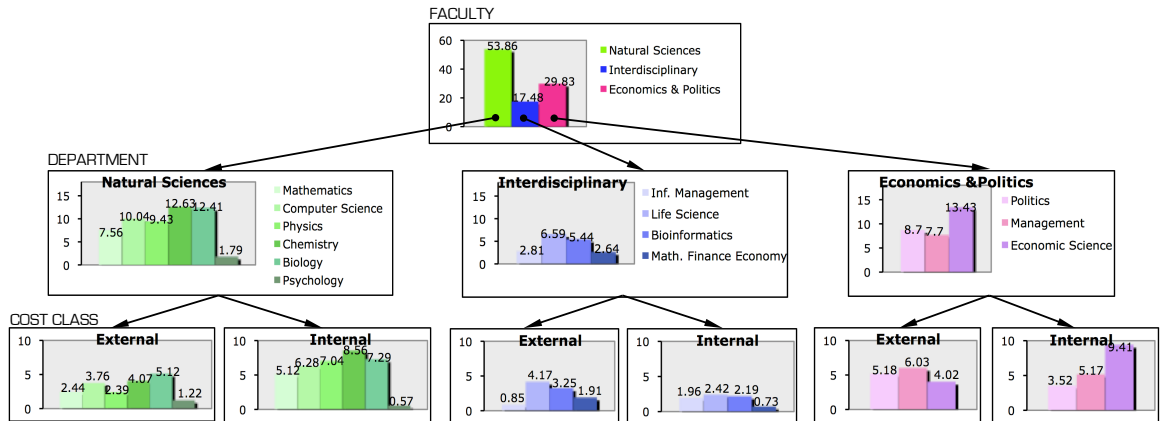


Figure 8.20: A decomposition tree of the measure student enrollments produced via a nested split by faculty, an outer split by department, and an inner split by cost class

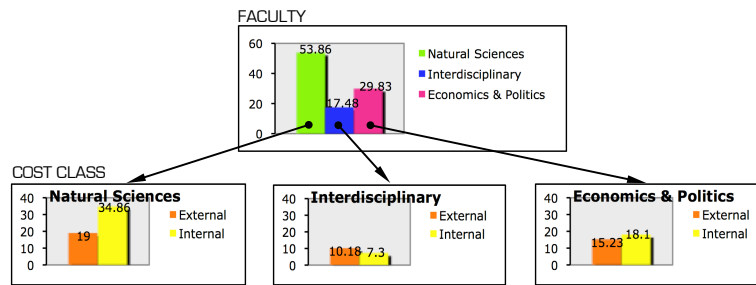


Figure 8.21: A decomposition tree with an orthogonal outer split (from faculty to cost class)

In addition to the above split types, it is important to distinguish between two semantic subtypes of a drill-down: *i*) drilling within a dimension occurs when the current split category \mathcal{C}_j represents a refinement of the previous split category \mathcal{C}_i ($\mathcal{C}_j \subseteq^* \mathcal{C}_i$), and *ii*) drilling out of a dimension occurs when the current split category is orthogonal to the previous split category. Both types are applicable in the split-along mode, however, with different implications. The split by department in Figure 8.20 represents the case of remaining within the teaching unit hierarchy. As a result, a child-level set of department values is individual for each faculty value of the parent level. Figure 8.21 depicts the results of an outer split for the same set of faculty aggregates along cost class. In this case, each child-level node is broken down by the identical set of member values, reflecting the orthogonality of cost class to faculty.

As exemplified by the above decomposition trees, an outer split can be performed for drilling both within and out of a dimension. The other two split modes, however, are more restrictive. Thereby, a non-initial nested split is admissible only in case of drilling within, since the granularity of the inner representation in the existing tree may only be refined but not replaced. An inner split, on the contrary, is valid only in case of an orthogonal drill-down as only in this case each parent set can be broken down by the entire member set of the split category. Checking the validity of attempted decomposition steps to prevent invalid user interactions is an important task of the visualization framework.

LAYOUT STRATEGIES

The ultimate success of applying any particular visualization technique depends on the type of task to be solved, the kind of data to deal with, as well as on the user's expertise and preferences. Therefore, our aim is to define a flexible framework rather than to provide a set of pre-configured layouts. Enhanced Decomposition Trees improve the quality of visualization by enabling hybrid layouts and providing various layout options at two levels: *i*) the hierarchical layout itself *ii*) embedded visual presentations.

An abundance of hierarchical visualization techniques proposed in the research literature fall into one of two major classes, namely, *connection* and *enclosure* layouts.

Connection layouts, also known as node-link diagrams in a classical aesthetic view [154] and in a variety of more compact layouts (*hyperbolic*, *balloon*, *radial*, etc., presented in [58]) are known to be easy to grasp and intuitive to interpret. These layouts can be used to increase the user's awareness of the hierarchical relationships within the data or to allow users to build their own hierarchies, such as decomposition trees.

In the aesthetic layout, nodes are aligned both vertically and horizontally, thereby enabling visual comparison of aggregates within the same level as well as in a top-down fashion: all nodes containing the subaggregates of the same granularity appear aligned and so are descendant subtrees with respect to their roots. The tree layout can be set either to vertical (placing child nodes underneath their parents) or to horizontal (left-to-right direction of parent-child relationships). Bar-chart nodes in the decomposition hierarchy depicted in Figure 8.20 are arranged into a vertical layout. Directing embedded bar-charts in the same direction appears optimal for perceiving the whole level as a single chart, provided that all of them are scaled uniformly.

The simplest implementation of a decomposition tree is a node-link diagram with a plain subaggregate value placed inside each node. This straightforward layout may be sufficient for the task of tracing the composition of an aggregate. Figure 8.22 shows an example of exploring a drill-across measure costs-per-student ratio by drilling down into semester and, subsequently, into faculty. Compared to the layout of the ProClarity's tree, our variant is more compact as it minimizes the node area by placing the associated dimensional characteristic outside the node over the respective edge.

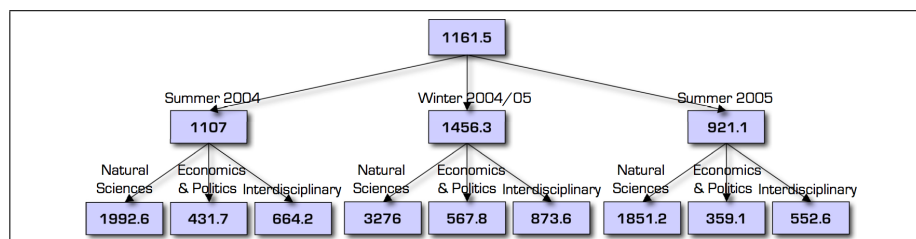


Figure 8.22: Exploring a drill-across measure costs-per-student ratio with a plain-text decomposition tree

Aesthetic node-link layouts are heavily criticized for extremely wasteful display utilization: sparsely populated upper levels consume as much area as the bottom ones. Nodes at lower levels quickly run out of space for displaying the labeling information, while a large portion of the display is wasted as background. Radial and balloon node-link trees scale somewhat better. *Enclosure*, also known as “space-filling” or “value-by-area”, offer a space-optimized solution. The compactness is achieved by scaling down the node size and allotting progressively less space for nodes deeper in the hierarchy. The most commonly used enclosure techniques are TreeMaps [163] and their variations, based on the rectangular node shape, and radial layouts, such as the SolarPlot [27] and InterRing [189], in which the aggregates are laid out radially with the coarsest aggregates in the center and finer grained levels farther away from the center.

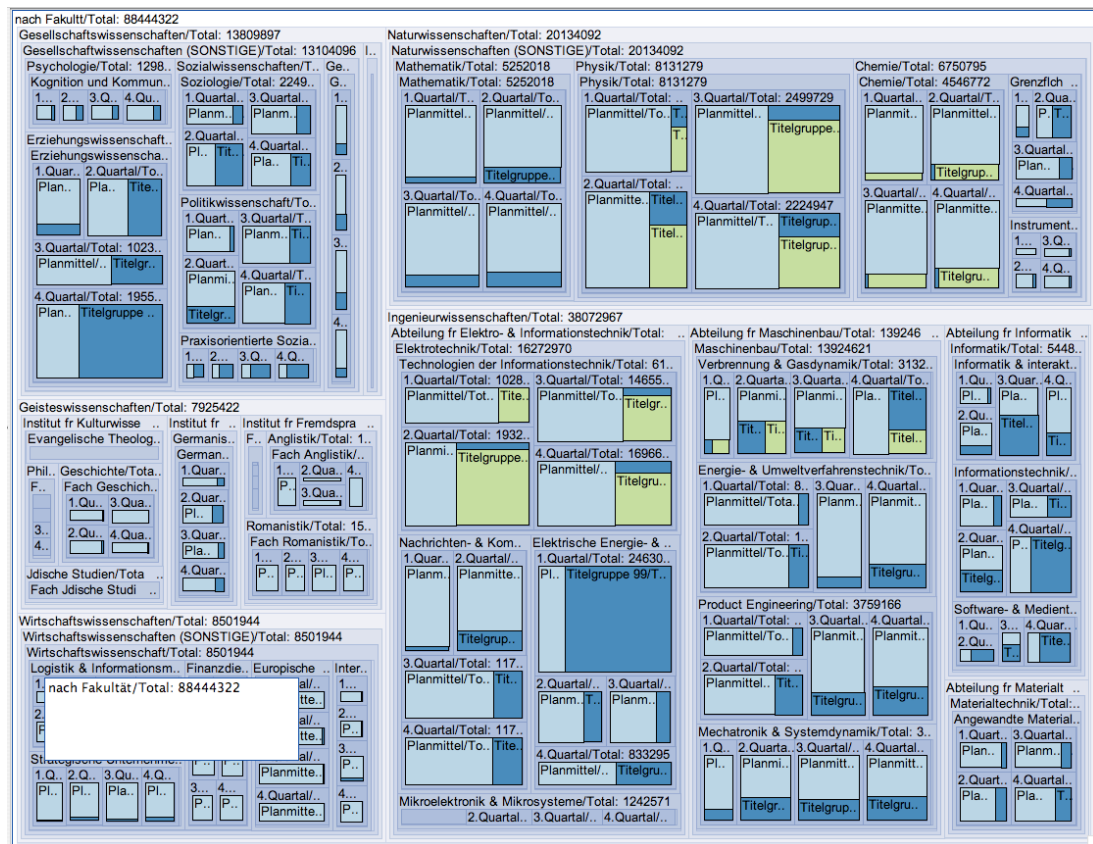


Figure 8.23: A rectangular space-filling decomposition tree (TreeMap)

Figure 8.23 demonstrates the use of a TreeMap layout for capturing hierarchical decomposition of total expenditures, broken down by faculty, department, unit, chair, quarter, and cost class. Each aggregate's value is mapped to the size of the respective rectangular node, enclosed in the parent node's area. Thereby, the display area is not only fully utilized, but is re-utilized by rendering child nodes inside parent nodes, resulting in the ability to visualize rather large data sets in a compact fashion. However, small values are hardly visible and there is no space to label them.

Figure 8.24 shows the results of querying student enrollment numbers using a SolarPlot visualization. The measure's aggregates are mapped to the size of the circle segments in each ring. For examining the distribution of foreign students over departments, the analyst filters out *Germany* in the dimension country (German "Land"), retrieves the subaggregates at subcontinent (German "Subkontinent") and, subsequently, at country level. In the final step, a drill-down by department (German "Institut") is mapped to the outermost ring. SolarPlot technique overcomes the problem of nested rectangle layouts (TreeMaps), which quickly run out of space for lower-level nodes. This is due to the fact that outer rings intrinsically occupy more space than the inner ones. However, radial layouts have a number of drawbacks, such as leaving large portions of the rectangular screen area unused, inability to accurately compare the nodes at different levels by area, and difficulty of labeling the node areas due to their differing orientation.

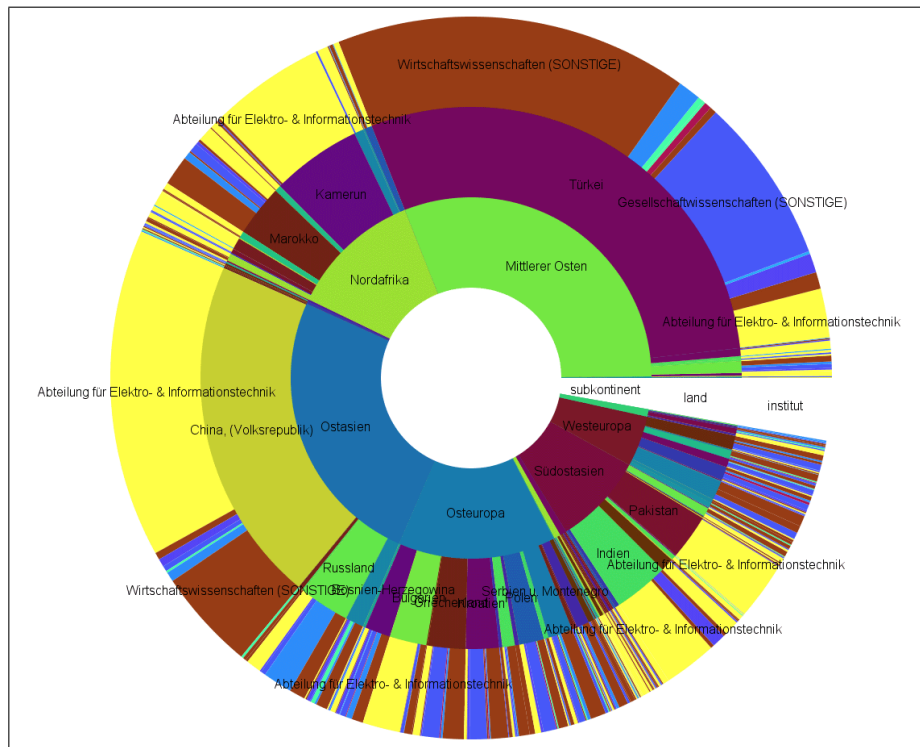


Figure 8.24: A radial space-filling decomposition tree (SolarPlot)

In space-filling techniques, a hierarchically decomposed aggregate is mapped to node size (area), whereas node color is used for encoding some relevant dimensional characteristic, as in Figure 8.24. Such “value-by-area” mapping is appropriate for exploring the aggregates computed using the SUM function where each subtotal represents a fraction of the total. However, non-cumulative aggregate functions, such as MIN, MAX, or AVG, produce values that remain within comparable ranges across the entire decomposition hierarchy and, therefore, may not be represented as subareas of the parent node. On the other hand, a common value range opens up an opportunity to map the measure to *color* using a common colormap for the entire value set.

To support non-cumulative disaggregation, we propose a visualization technique called *Hierarchical HeatMap*, demonstrated in Figure 8.25. The layout is inspired by the pivot table technique: bottom-level nodes are represented as a column of equally-sized cells and each higher-level nodes are shaped as cells spanning the width of their subtrees. The value of an aggregate is represented as plain text as well as the node’s background color. The color value is drawn using a variant of a “heat” colormap (colder or lighter tones for lower values and warmer or darker tones for higher values).

Hierarchical HeatMap instances depicted in Figure 8.25 are obtained by decomposing the measure *expenditure amount* along the hierarchy of teaching institution: aggregates in (a) are computed using AVG and those in (b) are obtained with MAX. The range and the distribution of the resulting set of aggregates is sensitive to the applied aggregate function: while MIN and MAX simply propagate extreme values to upper levels, AVG has a “smoothing” effect on the values at upper levels. Our technique compensates for different aggregation patterns by employing different colormap settings. In case of average values, a linear scale and a unipolar colormap based on the increasing color intensity provide an intuitive view of the measure’s overall behavior

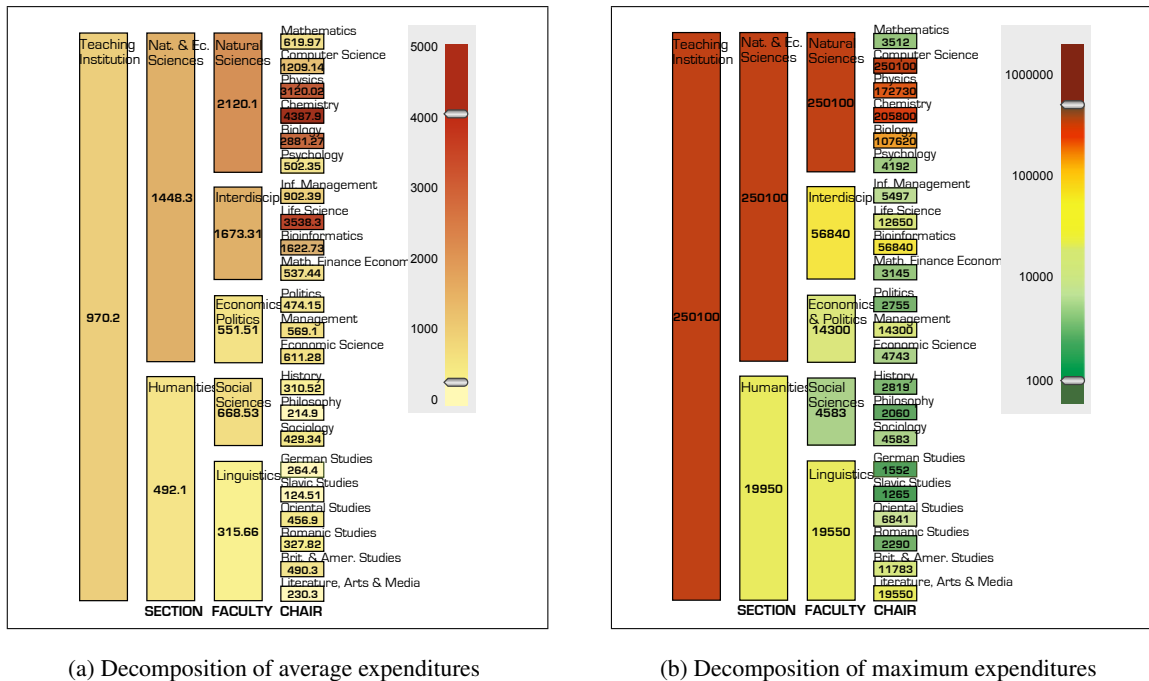


Figure 8.25: Hierarchical HeatMaps for non-cumulative disaggregation

and enable immediate recognition of outliers. In case of maximum/minimum values, a logarithmic scale and a multipolar colormap can successfully cope with the skewness of the distribution. Sliders at the poles of the colormap are useful for dynamically adjusting the visualization's sensitivity to outliers.

So far, we only considered bar-charts as an embedded representation in decomposition trees. Business charts enjoy great user acceptance for their simplicity and ease of interpretation. As data subsets within single nodes tend to be rather small and 2-dimensional (measure axis and the nested split dimensional axis), there is no urge to employ more complex metaphors at this level. Basically, there is no limitation to what layouts can be embedded in the nodes. However, bar-charts are known to outperform other chart types (e.g., line-charts or area-charts) in terms of accuracy for presenting quantitative information, robustness to outliers (by adjusting the range or the method of the scale), ability to present negative numbers and multiple measures.

Used in the context of hierarchical decomposition and embedded in an aesthetic node-link view, standard bar-charts lead to rather wasteful display utilization, as can be observed in Figure 8.19c. We propose two improved bar-chart variants, namely, *i*) space-filling bars and *ii*) area-preserving bars.

The first variant takes advantage of space-filling methodology for arranging the bars in a chart, as depicted in Figure 8.26: the compactness of bar-charts is improved by replacing equal-width bars with equal-height bars, proposed by Keim et al. in [76]. Thus, the whole chart is transformed into a partitioned rectangle, with no display area wasted as background. The tree view in Figure 8.26 shows the results of disaggregating the measure expenditure amount along a heterogeneous hierarchy purchaser. With space-filling bar-chart decomposition trees, all bars can be visually compared with one another by considering their areas or widths.

Another similar option is based on area-preserving bars. An example of a horizontal decomposition tree with such bars is depicted in Figure 8.27 and refers to the same scenario as the bar-chart tree in Figure 8.19c. The idea is rather simple: charts within the same level use the same scale and the same bar width, whereas at

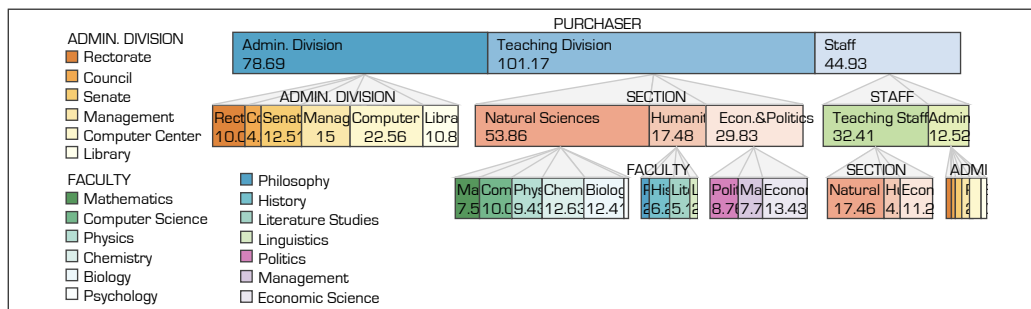


Figure 8.26: A decomposition tree with space-filling bars

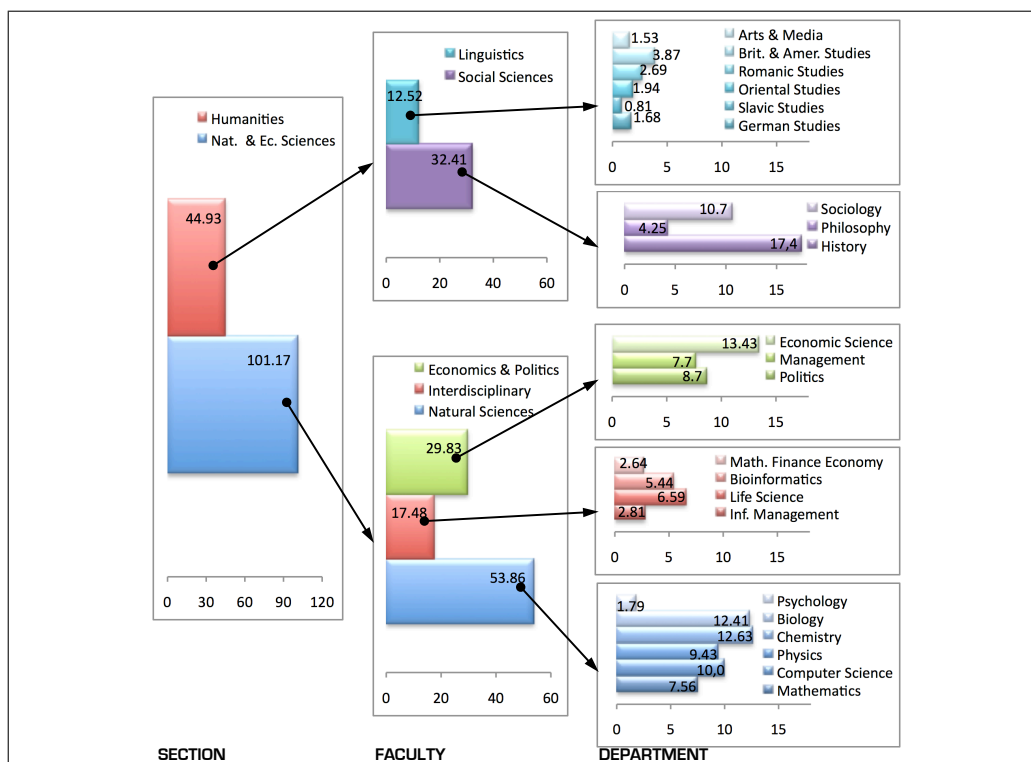


Figure 8.27: A decomposition tree with area-preserving bars

each child level the width of the bars scales down proportionally to the decrease of the scale with respect to the parent level. As a result, all bars remain comparable across levels by area, as in TreeMaps. Area-preserving bars are not as scalable as space-filling bars, but, embedded in an aesthetic tree layout, they naturally achieve a better utilization of the display than standard bar-charts while preserving the familiar chart view.

8.3.4 Spatio-temporal Visualization Techniques

In the multidimensional data model, spatial and temporal components of the data are treated just like any other dimensions. However, to fully exploit the potential and the richness of spatio-temporal data attributes, the awareness of their specific characteristics has to be incorporated into the data warehouse and the analysis tools on top of it. The outstanding role of the temporal dimension in OLAP is reflected in the data warehouse definition: “...a subject-oriented, integrated, time-variant, non-volatile collection of data in support of management’s decision making process” [68]. Rivest et al. [155] point out that space is the other of the two analytical components needed to take full advantage of a data warehouse, and that spatial dimensions, just like temporal ones, should receive a special treatment in any data warehouse implementation. Indeed, there is an estimation that about 80% of all data stored in corporate databases has a spatial component [43].

The requirements towards handling spatio-temporal data in a visual interface are twofold: *i)* to communicate spatial or temporal patterns hidden within the data and *ii)* to facilitate query specification by building up a mental image of the data distribution, which helps in restricting too generally formulated queries to data regions of interest. We enhance the decomposition trees with temporal and spacial components by adopting existing specialized visual presentations, such as *calendar views* and *maps*. In what follows, we present two examples of spatio-temporal exploration tasks and a variant of a temporal and a spacial decomposition tree, employed for solving the respective task.

Scenario 1: Temporal decomposition. To visualize the temporal evolution of a certain measure within a data cube, a Recursive Pattern visualization technique [78] can be adapted to fit into a calendar view, similar to the calendar-based visualization proposed in [186]. Figure 8.28 demonstrates the results of analyzing the volumes of email communication along the time axis using the above approach. As the information can be displayed at several granularities without changing the positions of the underlying data items, this technique has proven to be very scalable. Recursive pattern calendar view can be used as an exploration technique in its

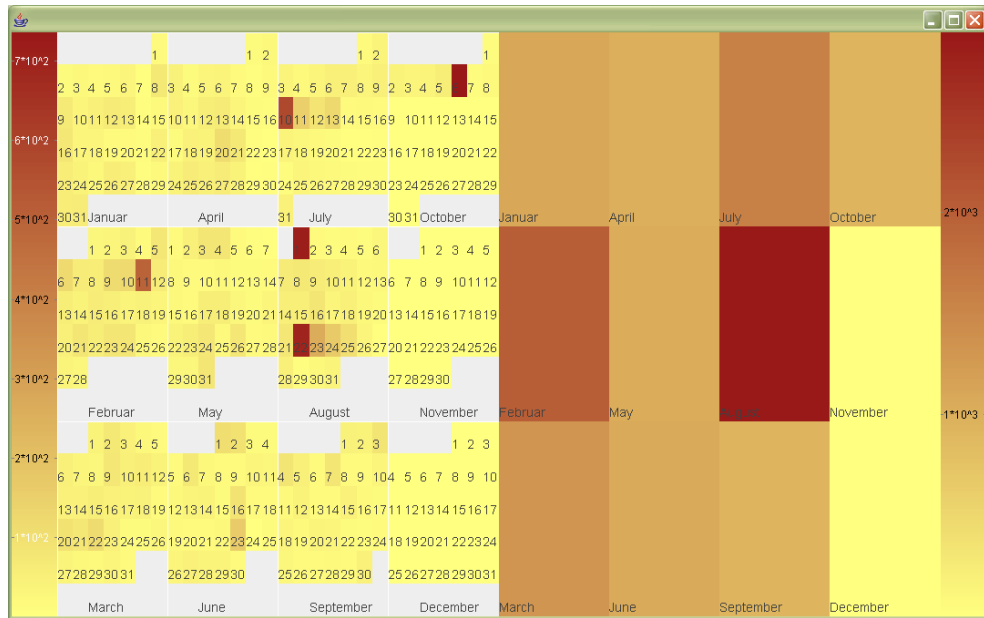


Figure 8.28: Multiscale *Recursive Pattern* visualization for drilling down along the time dimension

own right or as a node-level view in a decomposition tree. The latter variant enables combination of temporal exploration with disaggregation along other dimensions.

Scenario 2: Spatial decomposition. Spatial attributes, such as address, location coordinates, position, orientation, or size, are frequently encountered dimensional characteristics in OLAP cubes. Geography is often used as an exploration axis due to the fact that other data characteristics (i.e., consumer behavior) may strongly vary in space and display interesting distribution behavior. Straightforward map visualization techniques proceed by colorizing regions of a map or a cartogram according to the measure's value. Using geographical maps often leads to a disadvantage that larger regions (e.g., a state or a country) – which do not necessarily encompass more data elements than small countries – dominate the visual impression and a lot of display space remains unused.

Distortion approaches are employed to overcome the disadvantages of classical maps. The measure of the analysis is used as a distortion parameter as demonstrated in [79]. In case of a gap-free rectangular distortion, the resulting map degrades to a TreeMap, as in the *Hierarchical Network Map* (HNMap) technique [112]. Figure 8.29 shows a HNMap instance, which reveals geographical distribution of the network traffic at a network gateway. The approach simultaneously displays two measures: one attribute is mapped to the size of rectangular areas and the other is represented by color. In this example, the size of the country's or the internet backbone system's node is proportional to the number of IP addresses it encompasses and the background color marks the actual traffic volumes registered at each node.

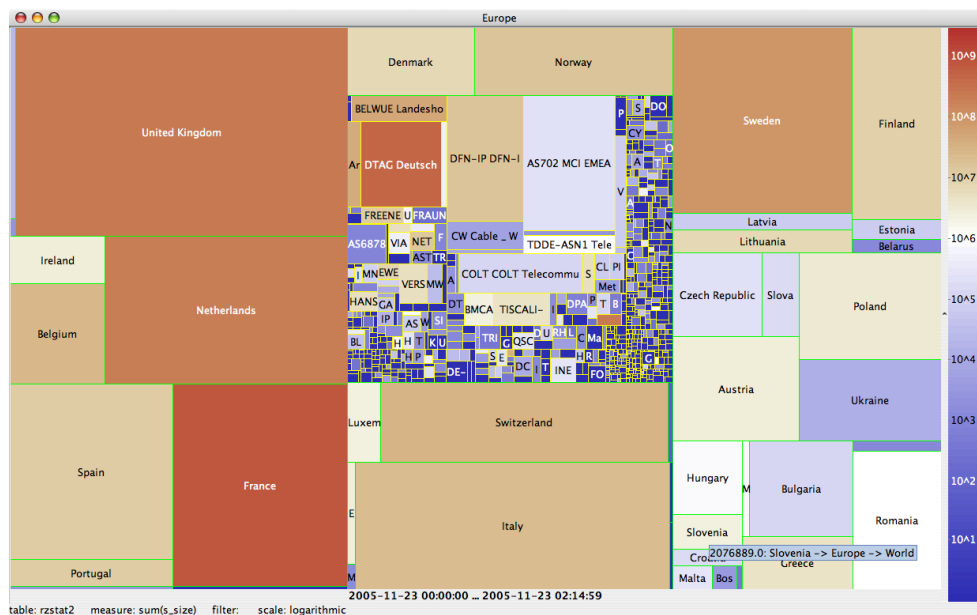


Figure 8.29: *Hierarchical Network Map* with geographical decomposition of the network traffic load by country and a drill-down into internet backbone system for *Germany's* node region

In the future, we plan to investigate other established and novel approaches to spatio-temporal visualization and incorporate them as further specializations of the Enhanced Decomposition Tree technique.

Chapter 9

Thesis conclusions and Future Work

THIS DISSERTATION has explored the issue of extending the OLAP technology to support complex data and non-conventional application scenarios. In this concluding chapter, we summarize and evaluate the main findings of our research, discuss the issues that remain open, and make suggestions for possible directions of further research.

Contents

9.1 Summary of Supported Multidimensional Properties	201
9.2 Conclusions	204
9.3 Future Work	208

9.1 Summary of Supported Multidimensional Properties

The main contribution of this work consists in extending the scope of analysis tasks and usage scenarios supported by the OLAP technology. We have come to realize that the extensions are due primarily in the conceptual model, which captures the relationships in the application domain in an intuitive and implementation-independent fashion. The capacity of a given multidimensional model can be presented in terms of the provided multidimensional constructs and their properties. In this section, we summarize the concepts proposed in different parts of this thesis and evaluate them against an extensive set of requirements found in the state-of-the-art literature on multidimensional modeling.

To enable a more systematic evaluation, we group the considered set of multidimensional properties into five categories: *i*) unification of the multidimensional space, *ii*) facts and measures, *iii*) dimensions and hierarchies, *iv*) hierarchy levels and roll-up relationships, and *v*) dynamic features. The properties chosen for the evaluation essentially correspond to the requirements formulated in Sections 3.1 and 3.2 of the thesis.

Table 9.1 enumerates the properties related to the unification of the multidimensional space as a foundation for modeling semantically related elements. In the original OLAP setting, each fact scheme is modeled in its own isolated space, resulting in the inability to capture semantic relationships across multiple fact schemes. Our approach overcomes this deficiency by distinguishing between the terms *category* and *category type* and presenting related categories as different roles of the respective category type.

A classification of measure and fact types as well as the resulting palette of multi-fact relationships defined primarily in Chapter 6 are given in Table 9.2. Many-to-many relationships between facts and dimensions are

Table 9.1: Multidimensional properties for mapping semantically related elements

Property	Support	Source
<i>Common multidimensional space</i>	+	Sections 3.4.1, 3.4
▶ <i>Unified multidimensional space</i>	+	
▶ <i>Conformed multidimensional space</i>	+	
<i>Semantically related dimensions</i>	+	Sections 3.4, 5.4.2
▶ <i>Conform dimensions</i>	+	
▶ <i>Partially shared dimensions</i>	+	
<i>Dimension sharing</i>	+	Section 5.4
▶ <i>Conformance</i>	+	
▶ <i>Overlap</i>	+	
▶▶ <i>Fact-as-dimension</i>	+	
▶ <i>Inclusion</i>	+	
▶▶ <i>Convergence</i>	+	
<i>Semantically related categories</i>	+	Section 3.4
▶ <i>Conform categories</i>	+	
▶ <i>Compatible categories</i>	+	
<i>Related fact schemes</i>	+	Section 3.4
▶ <i>Fact family (galaxy)</i>	+	
▶ <i>Fact cluster</i>	+	

Table 9.2: Multidimensional properties related to measures and facts

Property	Support	Source
<i>Measure aggregation semantics</i>	+	Section 5.2.1
▶ <i>Aggregability</i>	+	
▶ <i>Additivity</i>	+	
▶▶ <i>Measure value type</i>	+	
<i>Derived (computed) measures</i>	+	Sections 3.3.1, 7.1.2
<i>Non-measurable facts</i>	+	Section 5.2.2
▶ <i>Event-tracking facts</i>	+	
▶ <i>Coverage facts</i>	+	
<i>Non-strict fact-dimensional roll-up</i>	–	
<i>Degenerate facts</i>	+	Section 5.3.1
▶ <i>Satellite fact</i>	+	
▶ <i>Association fact</i>	+	
▶▶ <i>Self-association fact</i>	+	
<i>Fact roll-up (composition)</i>	+	Section 5.3.2
<i>Heterogeneous facts</i>	+	Section 5.3.3
▶ <i>Optional dimensions</i>	+	Section 4.3.2
▶ <i>Fact generalization / specialization</i>	+	

marked as unsupported since our model forces such relationships to be resolved into satellite facts for the sake of correct aggregation.

Table 9.3 summarizes the contributions of Chapter 4 in the form of dimension and hierarchy types. Incomplete and overlapping specializations are marked with ‘+/-’ as those hierarchy types can be expressed in the conceptual scheme “as they are”, however, our model forces their transformation into summarizable structures as a preparation step for providing their logical mapping.

Table 9.3: Multidimensional properties related to dimensions and hierarchies

Property	Support	Source
<i>Derived dimension</i>	+	Sections 3.3.1, 4.3.2, 7.1.2
<i>Degenerated dimension</i>	+	Sections 4.3.2, 5.2.2
▶ <i>Fact identifier</i>	+	
<i>Optional dimension</i>	+	
<i>Hierarchically structured dimension</i>	+	Sections 3.4.2, 4.3.2
▶ <i>Multiple hierarchies</i>	+	Sections 3.3.1, 4.5
▶▶ <i>Alternative hierarchies</i>	+	
▶▶ <i>Parallel hierarchies</i>	+	
▶▶ <i>Dependent hierarchies</i>	+	
<i>Totally ordered dimension</i>	+	Section 3.4.2
<i>Non-strict hierarchy</i>	+	Sections 4.4.1, 7.3.2
▶ <i>Weighted non-strict hierarchy</i>	+	
▶ <i>Fuzzy hierarchy</i>	–	
<i>Complete hierarchy</i>	+	Section 4.4.2
<i>Asymmetric (non-onto) hierarchy</i>	+	Sections 4.4.2, 7.3.3
<i>Heterogeneous hierarchy</i>	+	Sections 4.4.3, 7.2
▶ <i>Non-covering hierarchy</i>	+	Section 7.2.1
▶ <i>Mixed-grain hierarchy</i>	+	Section 7.2.2
▶ <i>Generalization/specialization</i>	+	Section 7.2.2
▶▶ <i>Incomplete specialization</i>	+/-	
▶▶ <i>Overlapping specialization</i>	+/-	
<i>Imprecise/incomplete hierarchy</i>	–	

Table 9.4: Multidimensional properties related to categories and roll-up relationships

Property	Support	Source
<i>Derived category</i>	+	Sections 3.3.1, 3.4.2, 7.1.2
<i>Totally ordered category</i>	+	Sections 3.3.1, 3.4.2
<i>Abstract (top) category</i>	+	Sections 3.3.1, 3.4.2, 4.4.3
▶ <i>Local root category</i>	+	Section 7.2.2
<i>Category type</i>	+	Sections 3.4.2, 5.4.1
<i>Property attribute</i>	+	Sections 3.3.1, 3.4.2
<i>Mutually exclusive roll-up relationships</i>	+	Sections 3.3.1, 3.4.2
▶ <i>Alternative roll-up relationships</i>	+	
▶ <i>Partial related roll-up relationships</i>	+	
▶▶ <i>Non-covering roll-up</i>	+	
<i>Degenerated roll-up relationship</i>	+	Section 4.3.2
<i>Optional (partial) roll-up relationship</i>	+	Sections 3.3.1, 3.4.2
<i>Non-strict roll-up relationship</i>	+	Sections 3.3.1, 3.4.2
▶ <i>Weighted non-strict roll-up relationship</i>	+	Section 7.3.2
▶ <i>Fuzzy roll-up relationship</i>	–	
<i>Complete roll-up relationship</i>	+	Sections 3.3.1, 4.4.2
<i>Generalization/specialization relationship</i>	+	Sections 3.3.1, 4.4.3, 5.3.3
▶ <i>Mixed-grain generalization</i>	+	Section 7.2.2
▶ <i>Set of overlapping specializations</i>	+	Section 7.2.2
▶ <i>Incomplete set of specializations</i>	+	

Table 9.5: Dynamic multidimensional properties

Property	Support	Source
<i>Ad hoc measure specification</i>	+	Section 5.2.1
▶ <i>Derived measure</i>	+	
<i>Scheme-transforming operators</i>	+	Sections 5.3.4, 8.2.4
▶ “PUSH” (<i>measure from a dimension category</i>)	+	Section 8.2.3
▶ “PULL” (<i>dimension from a measure</i>)	+	Section 8.2.3
▶ “DRILL-ACROSS” (<i>fact overlap</i>)	+	Section 8.2.3
<i>Ad hoc hierarchy</i>	+	Section 8.2.3
▶ <i>Ad hoc dimension category</i>	+	
▶▶ <i>Derived dimension category</i>	+	
<i>Ad hoc cube specification</i>	+	Section 8.2.3
<i>DRILL-ASIDE (resolution of a non-strict roll-up)</i>	+	Section 8.2.4

Fuzzy hierarchies are stated as unsupported, even though the *X*-DFM provides a fuzzy roll-up edge construct. We have deliberately abstained from including fuzzy features into the formal model in order not to explode the scope of this dissertation. Imprecision and incompleteness are further currently unsupported features identified as direction for future work.

Categorization of dimension category types and the kinds of roll-up relationships between the them is shown in Table 9.4.

Finally, Table 9.5 summarizes the dynamic properties of the multidimensional model, presented predominantly in Chapter 8. The latter are not “hard-coded” into the conceptual scheme but are added by a user at query time. User-defined elements are specified directly in frontend tools and can be stored for future use as additional metadata within the respective user’s profile.

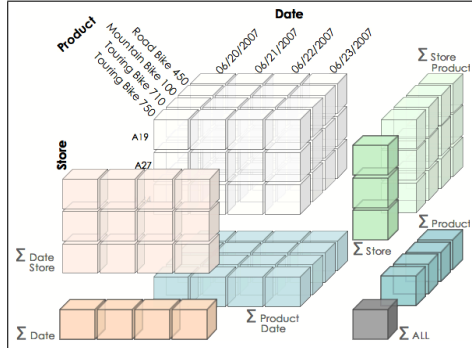
Not surprisingly, our proposed multidimensional model and its graphical notation cover most of the defined requirements. After all, the objective of this work was to systematize the wealth of existing concepts related to the conceptual data warehouse design and enrich them with the results of our research in order to obtain a powerful modeling framework. To the best of our knowledge, our proposed formalization, classification, and graphical notation are the most comprehensive and coherent in the state of the art.

9.2 Conclusions

The research represented in this thesis was motivated by two observations. On the one hand, data warehousing and OLAP are gaining momentum as the core technology for decision support in the business world and beyond. On the other hand, disclosure of novel application domains pushes this technology to its limits and invalidates many of its established concepts. This work presents an attempt to reduce the gap between the capacities of the state-of-the-art OLAP systems and the challenges imposed by comprehensive data analysis and emerging decision-support applications.

In the introductory Chapter 1 we explained the general complexity of extending OLAP functionality by pointing out that OLAP is an integral component of a multilayer data warehouse system architecture. The overall power of such a complex system is determined by the interplay of the involved components and therefore, any improvement introduced at a lower layer must be propagated upwards in the system in order to reach the end-user. We also stated that many deficiencies of the standard technology are due to the rigidity of the underlying multidimensional model. This insight coined the structure of this thesis.

Background and Related Work (Chapter 2)

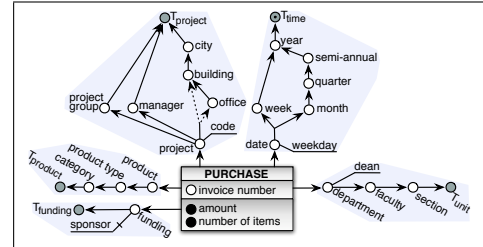


This chapter set the stage for presenting our research by providing the necessary background in terms of the relevant terminology and concepts as well as of the related work starting from the fundamentals and components of Business Intelligence, followed by an overview of the data warehousing technology and its outstanding characteristics, OLAP operations and implementation alternatives, the multidimensional data model, and the data warehouse design methodology. We also highlighted the fundamentals of visual OLAP as an emerging paradigm for exploring multidimensional aggregates.

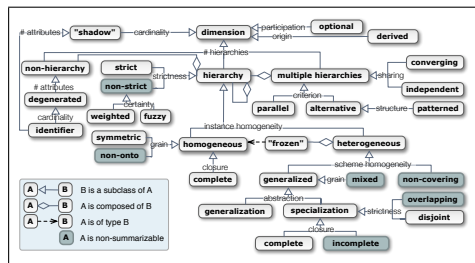
Extending the Multidimensional Data Model (Chapter 3)

The central contribution of this work has been to develop a conceptual multidimensional data model capable of handling complex and non-conventional data and analytical tasks. This chapter laid the foundation for the new model by investigating the challenges of data warehouse design for advanced applications, revising the state of the art, and identifying major “bottle-necks” in the existing systems. The results of this survey were organized into a set of requirements and properties to be satisfied by the extended model.

The proposed conceptual design framework comes in the form of a formal model and an accompanying graphical modeling notation. Since none of the existing graphical models appeared to be fully compliant with the formulated requirements, we developed our own model, denoted *X-DFM*, which extends and modifies the popular Dimensional Fact Model of Golfarelli et al. [47]. Both the formalization and the graphical toolkit were defined at three levels of abstraction – the lower, the intermediate, and the upper level – to support consecutive layering of the conceptual scheme at different design stages. The concept of the unified multidimensional space was introduced as a cornerstone for capturing semantic relationships between the elements. In this chapter, we define only the core elements of the conceptual model, such as fact, dimension, hierarchy, and roll-up relationship, to be particularized in the next two chapters.



Dimensions and Hierarchies in the Multidimensional Data Model (Chapter 4)

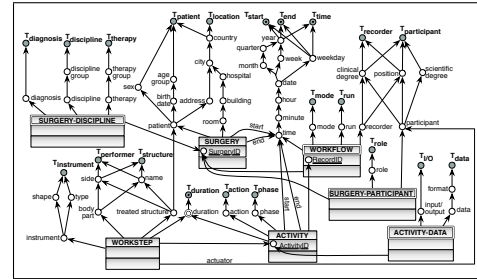


This chapter was dedicated to extending the multidimensional model as to support complex dimension and hierarchy types. This work was inspired by the challenges encountered in a case study from the domain of academic administration. We demonstrated that the established constraints of the conventional model, such as homogeneity, strictness, and regularity, appear too restrictive for many real-world applications and, therefore, have to be overcome at the conceptual and, subsequently, at the logical level in order to provide adequate OLAP support to such data domains.

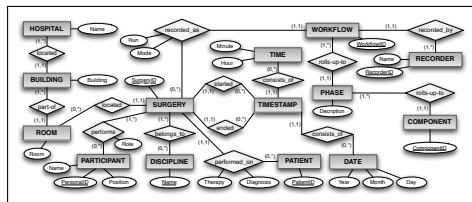
We undertook a systematic categorization of dimension and hierarchy types in the form of a metamodel of dimensional modeling and provided formal definitions accompanied by illustrating examples for each of the identified classes. Strong emphasis was placed on examining non-summarizable hierarchy types and identifying inadmissible relationships in the data, which must be remodeled in the conceptual scheme prior to proceeding to the logical design phase.

Measures, Facts, and Galaxies in the Multidimensional Data Model (Chapter 5)

While Chapter 4 focused on classifying complex dimensions, here we provided a similar categorization of facts and measures. We used surgical workflow analysis as a motivating non-conventional usage scenario. Fact schemes were subdivided into measurable and non-measurable ones as a foundation for determining the aggregation semantics. By employing the concept of the unified multidimensional space, we were also able to capture multi-fact relationships, such as degeneration, composition, and generalization as well as to classify dimension sharing patterns within and across fact schemes. We also demonstrated the interchangeability of fact and dimension roles when dealing with semantically related fact schemes.



Data Warehouse Design for Non-Conventional Applications (Chapter 6)

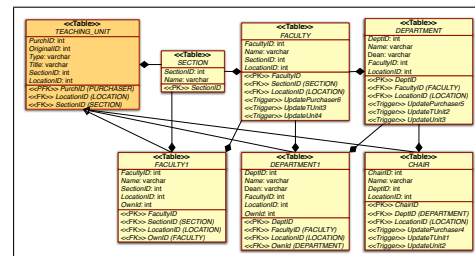


This chapter raised the issue of reconsidering the classical data warehouse design methodology for dealing with non-conventional applications, such as warehousing operational data. Considering the fact that data warehouses are typically built on top of existing information systems and data sources, we developed a methodology for semi-automatic acquisition of multidimensional schemes from existing data models. The proposed solution evolves as a the verification

and refinement of the existing conceptual scheme, followed by its cardinality-based transformation into a multidimensional one. The capabilities of the proposed modeling framework were demonstrated by solving sample analysis tasks from the field of surgical workflow analysis.

Relational Implementation of the Multidimensional Data Model (Chapter 7)

The aim of this chapter was to demonstrate the implementability of the proposed conceptual model in a ROLAP system with a reasonable overhead. The relational implementation was favored over the multidimensional one for its superiority in handling complex data structures. A well-established requirement of summarizability, i.e., correct aggregate navigation, served as a benchmark for determining the elements of the conceptual model, which had to undergo scheme and/or instance transformation prior to obtaining their logical mappings.



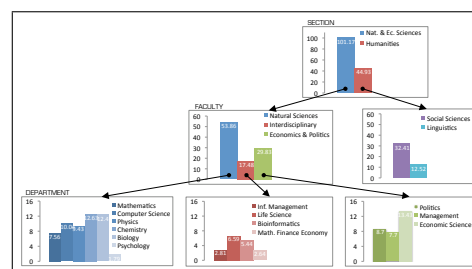
We arrived at a conclusion that irregular dimension hierarchies represent the major challenges for relational implementation and proposed a two-phase hierarchy normalization approach for enforcing summarizability in dimension hierarchies. The first phase is concerned with the normalization of heterogeneous schemes by eliminating incomplete and overlapping specialization, mixed-grain, and non-covering roll-up relationships. In the next phase, multiple and generalized hierarchies are decomposed into constituent homogeneous subtrees in order to identify non-onto and non-strict mappings to be normalized at the instance level.

Finally, we paid a tribute to the outstanding role of the metadata in data warehousing systems, where the former acts as an intermediary between various components and, in particular, between the physical data model and the application layer residing on top it. We showed how the Common Warehouse Metamodel, which is an established industrial standard for metadata interchange in data warehousing environments, can be employed for capturing the rich multidimensional semantics behind the relational implementation.

Interactive Exploration of OLAP Aggregates (Chapter 8)

This chapter addressed the issue of supporting complex data at the frontend layer based on the emerging Visual OLAP paradigm. Aware of the deficiencies of standard business visualization techniques and employed data navigation approaches in OLAP tools, we proposed an enhanced visual framework for comprehensive exploration of multidimensional aggregates. Our framework incorporates the awareness of the conceptual extensions in the form of enriched metadata along with a powerful data interface for interactive specification of ad hoc queries. We also provided an exhaustive set of OLAP operators and elaborated on their possible implementations (e.g., as navigation events or interaction techniques) in a visual interface.

Finally, we investigated the typical patterns of interacting with OLAP cubes and proposed to use hierarchical layouts for mapping the iterative nature of navigating to a dataset of interest. In particular, we designed a class of hierarchical visualization techniques called *Enhanced Decomposition Tree*, in which hierarchical layouts are employed for arranging aggregate values into a decomposition hierarchy, combined with simple and intuitive chart representations used for displaying the sets of values inside single nodes. We experimented with different hierarchical and chart layouts, colormap and scaling options in order to identify visualizations that best match a specified criterion (e.g., compactness, precision, robustness against outliers).



9.3 Future Work

The results presented in this thesis improve the claimed universality of the data warehousing technology by bringing it some steps closer to the requirements of comprehensive data analysis. However, the set of features enabled by our framework is by no means exhaustive. Some of the useful properties were omitted due to the limited timeframe of our work whereas others were excluded deliberately as they did not fit or would have exploded the scope of the resulting model. Our work can be continued by connecting it to other related research areas, such as spatio-temporal analysis, real-time data warehousing, intergration of OLAP and data mining, or by deepening the issues central to this thesis. In the latter category, a number of promising directions for future research can be identified.

Conceptual design

We have not considered the outstanding role of the time dimension and temporal characteristics in OLAP. It would be useful to provide formalisms for capturing time-related semantics and to handle evolution of dimension instances over time. In the literature, the term “*slowly chanding dimensions*” [81] is used to denote dimensions, prone to modifications over time. Whereas various solutions to handling such dimensions exist at the level of logical design, conceptual models tend to neglect this property.

Formalization of OLAP operators, especially of the scheme-transforming ones, would be beneficial for modeling the outcomes of those operators at the conceptual level as well as for semantic query optimization.

The set of OLAP operations can be extended to include novel operators and aggregate functions, tailored towards specific data types and analysis tasks. For instance, multidimensional analysis of text data could benefit from retrieval operators, such as substring matching and similarity search, whereas business process analysis requires operators for process decomposition and consolidation, subprocess matching, and identifying similar process fragments.

Logical design

Implementation of the conceptual scheme is an important issue. So far, we only considered the relational representation and identified a set of normalization techniques necessary in the relational context. It would be useful to also provide a MOLAP and an object-relational DBMS implementation alternative and become aware of the respective mapping challenges. We also intend to develop a CASE tool that would integrate our proposed conceptual design methodology based on the *X*-DFM notation and the transformation techniques for obtaining logical representations from the multidimensional schemes.

Visual OLAP

At present, visual OLAP is lacking a unified formal model and query specification standard. Existing frameworks are based on proprietary formalisms and models. To be universally adopted, a new standard has to be open, flexible, and extendible to account for a wide range of visualization approaches. Furthermore, advanced OLAP tools claim to turn visualization from the presentation layout into the method of data exploration, raising the necessity of re-defining visualization as an instrument in terms of its structural components and interaction functions.

Another direction for future work is to search for novel visual querying paradigms as well as on adopting and extending visualization and interaction techniques from the area of Information Visualization. Especially those layouts capable of elegantly presenting large data volumes and/or a large number of dimensions are demanded. Spatial and temporal OLAP support can be enabled by providing visualization techniques with spatio-temporal “intelligence”, such as maps, cartograms, calendar views, and animated scatterplots.

Bibliography

- [1] A. Abelló, “YAM² : a multidimensional conceptual model,” Ph.D. dissertation, Universitat Politècnica de Catalunya, Apr. 2003.
- [2] A. Abelló, J. Samos, and F. Saltor, “A framework for the classification and description of multidimensional data models,” in *DEXA 2001: Proceedings of the 12th International Conference on Database and Expert Systems Applications*. Springer-Verlag, 2001, pp. 668–677.
- [3] A. Abelló, J. Samos, and F. Saltor, “Understanding facts in a multidimensional object-oriented model,” in *DOLAP '01: Proceedings of the 4th ACM International Workshop on Data Warehousing and OLAP*, 2001, pp. 32–39.
- [4] A. Abelló, J. Samos, and F. Saltor, “On relationships offering new drill-across possibilities,” in *DOLAP '02: Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP*, 2002, pp. 7–13.
- [5] R. Agrawal, A. Gupta, and S. Sarawagi, “Modeling multidimensional databases,” in *ICDE '97: Proceedings of the 13th International Conference on Data Engineering*. Washington DC, USA: IEEE Computer Society, 1997, pp. 232–243.
- [6] N. Andrienko, G. Andrienko, and P. Gatalsky, “Exploratory spatio-temporal visualization: an analytical review,” *Journal of Visual Languages & Computing*, vol. 14, no. 6, pp. 503–541, 2003.
- [7] C. Ballard, D. M. Farrell, A. Gupta, C. Mazuela, and S. Vohnik, *Dimensional Modeling: In a Business Intelligence Environment*, ser. IBM Redbooks. Vervante, 2006.
- [8] A. Bauer and H. Günzel, Eds., *Data Warehouse Systeme - Architektur, Entwicklung, Anwendung*, 2nd ed. Heidelberg: dpunkt-Verlag, 2004.
- [9] A. Bauer, W. Hümmer, and W. Lehner, “An alternative relational OLAP modeling approach,” in *DaWaK 2000: Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery*. London, UK: Springer-Verlag, 2000, pp. 189–198.
- [10] D. Baum, “Interview with Gartner Group’s Howard Dresner,” *Information Builders Magazine*, vol. 11, no. 2, pp. 26–28, 2001.
- [11] D. Bulos, “OLAP database design: A new dimension,” *Database Programming & Design*, vol. 9, no. 6, pp. 33–37, 1996.

- [12] O. Burgert, T. Neumuth, M. Gessat, S. Jacobs, and H. U. Lemke, "Deriving DICOM surgical extensions from surgical workflows," *Medical Imaging 2007: PACS and Imaging Informatics*, vol. 8, no. 35, p. 651604, 2007.
- [13] O. Burgert, T. Neumuth, F. Lempp, R. Mudunuri, J. Meixensberger, G. Strauß, A. Dietz, P. Jannin, and H. U. Lemke, "Linking top-level ontologies and surgical workflows," *International Journal of Computer Assisted Radiology and Surgery*, vol. 1, no. 1, pp. 437–438, 2006.
- [14] BusinessObjects, an SAP Company, "Intelligence Platform." [Online]. Available: <http://www.businessobjects.com/product/platforms/>
- [15] J. W. Buzydlowski, I.-Y. Song, and L. Hassell, "A framework for object-oriented on-line analytic processing," in *DOLAP '98: Proceedings of the 1st ACM International Workshop on Data Warehousing and OLAP*. New York, NY, USA: ACM Press, 1998, pp. 10–15.
- [16] L. Cabibbo and R. Torlone, "Querying multidimensional databases," in *DBLP-6: Proceedings of the 6th International Workshop on Database Programming Languages*. London, UK: Springer, 1997, pp. 319–335.
- [17] L. Cabibbo and R. Torlone, "From a procedural to a visual query language for OLAP," in *SSDBM '98: Proceedings of the 10th International Conference on Scientific and Statistical Database Management*. Washington DC, USA: IEEE Computer Society, 1998, pp. 74–83.
- [18] L. Cabibbo and R. Torlone, "A logical approach to multidimensional databases," in *EDBT'98: Proceedings of the 6th International Conference on Extending Database Technology*, ser. LNCS, vol. 1377. Springer, 1998, pp. 183–197.
- [19] L. Cabibbo and R. Torlone, "The design and development of a logical system for OLAP," in *DaWaK 2000: Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery*, ser. LNCS, vol. 1874. Springer, 2000, pp. 1–10.
- [20] S. K. Card, J. D. Mackinlay, and B. Shneiderman, Eds., *Readings in information visualization: using vision to think*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.
- [21] D. T. Chang, "CWM enablement showcase: Warehouse meta data interchange made easy using CWM," *Data Warehousing: What Works?*, vol. 11, May 2001. [Online]. Available: <http://www.tdwi.org/research/display.aspx?ID=5003>
- [22] I. Charlesworth, "Fractal Edge Visualization Summary," 2007, White Paper. [Online]. Available: <http://extranet.fractaledge.com/Materials/FractalEdgeVisualization.pdf>
- [23] S. Chaudhuri and U. Dayal, "An overview of data warehousing and OLAP technology," *ACM SIGMOD Record*, vol. 26, no. 1, pp. 65–74, 1997.
- [24] S. Chaudhuri, U. Dayal, and V. Ganti, "Database technology for decision support systems," *Computer*, vol. 34, no. 12, pp. 48–55, 2001.
- [25] P. P.-S. Chen, "The entity-relationship model – toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9–36, 1976.
- [26] Y. W. Choong, D. Laurent, and P. Marcel, "Computing appropriate representations for multidimensional data," *Data & Knowledge Engineering*, vol. 45, no. 2, pp. 181–203, 2003.

- [27] M. C. Chuah, "Dynamic aggregation with circular visual designs," in *InfoVis '98: Proceedings of the 1998 IEEE Symposium on Information Visualization*. IEEE Computer Society, 1998, pp. 35–43.
- [28] W. S. Cleveland and R. McGill, "Graphical perception: Theory, experimentation, and application to the development of graphical methods," *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 531–554, 1984.
- [29] E. F. Codd, "Extending the database relational model to capture more meaning," *ACM Transactions on Database Systems*, vol. 4, no. 4, pp. 397–434, 1979.
- [30] E. F. Codd, S. B. Codd, and C. T. Salley, "Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate," E.F.Codd & Associates, Tech. Rep., 1993.
- [31] Cognos ULC, an IBM Company, "IBM cognos PowerPlay: Overview–OLAP Software." [Online]. Available: http://www.cognos.com/products/business_intelligence/analysis/
- [32] O. T. I. Company, "Oracle Business Intelligence Discoverer." [Online]. Available: <http://www.oracle.com/technology/products/discoverer/>
- [33] T. M. Connolly and C. E. Begg, *Database Systems: A Practical Approach to Design, Implementation and Management (4th Edition)*. Pearson Addison Wesley, 2004.
- [34] M. Corporation, "Microsoft Office PerformancePoint Server),” 2007. [Online]. Available: <http://www.microsoft.com/business/performancepoint/>
- [35] L. Data Mining & Analysis, "MOLAP or ROLAP,” 1998. [Online]. Available: <http://www.donmeyer.com/art3.html>
- [36] A. Datta and H. Thomas, "The cube data model: a conceptual model and algebra for on-line analytical processing in data warehouses," *Decision Support Systems (Special Issue on WITS'97)*, vol. 27, no. 3, pp. 289–301, 1999.
- [37] D. Dodds, H. Hasan, and E. Gould, "Relational versus multidimensional databases as a foundation for online analytical processing," in *IRIS 22: Proceedings of the 22nd Information Systems Research Seminar in Scandinavia*, 1999, pp. 281–288.
- [38] R. J. Earle, "Method and apparatus for storing and retrieving multi-dimensional data in computer memory," Oct. 1995, U.S. Patent No. 5,359,724.
- [39] S. G. Eick, "Visualizing multidimensional data," *SIGGRAPH Comput. Graph.*, vol. 34, no. 1, pp. 61–67, 2000.
- [40] S. G. Eick and A. F. Karr, "Visual scalability," *Journal of Computational & Graphical Statistics*, vol. 11, pp. 22–43, 2002.
- [41] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 5th ed. Addison-Wesley, 2007.
- [42] E. Franconi and U. Sattler, "A data warehouse conceptual data model for multidimensional aggregation," in *DMDW'99: Proceedings of the International Workshop on Design and Management of Data Warehouses*, ser. CEUR Workshop Proceedings, vol. 19. CEUR-WS.org, 1999, pp. 13.1 – 13.10.
- [43] C. Franklin, "An introduction to geographic information systems: linking maps to databases," *Database*, vol. 15, no. 2, pp. 12–21, 1992.

- [44] M. Gebhardt, M. Jarke, and S. Jacobs, "A toolkit for negotiation support interfaces to multi-dimensional data," in *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 1997, pp. 348–356.
- [45] W. A. Giovinazzo, *Object-oriented data warehouse design: building a star schema*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.
- [46] P. Gluchowski, "Business Intelligence: Konzepte, Technologien und Einsatzbereiche," *HMD - Praxis der Wirtschaftsinformatik*, vol. 222, Dec. 2001.
- [47] M. Golfarelli, D. Maio, and S. Rizzi, "The Dimensional Fact Model: A conceptual model for data warehouses," *International Journal of Cooperative Information Systems*, vol. 7, no. 2-3, pp. 215–247, 1998.
- [48] M. Golfarelli and S. Rizzi, "A methodological framework for data warehouse design," in *DOLAP'98: Proceedings of the 1st ACM International Workshop on Data Warehousing and OLAP*. New York, NY, USA: ACM Press, 1998, pp. 3–9.
- [49] V. Gopalkrishnan, Q. Li, and K. Karlapalem, "Star/snow-flake schema driven object-relational data warehouse - design and query processing strategies," in *DaWaK '99: Proceedings of the 1st International Conference on Data Warehousing and Knowledge Discovery*. London, UK: Springer, 1999, pp. 11–22.
- [50] D. Gornik, "UML data modeling profile," Rational Software Corporation, IBM, Tech. Rep., May 2002, TP162.
- [51] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 29–53, 1997.
- [52] M. Gyssens and L. V. S. Lakshmanan, "A foundation for multi-dimensional databases," in *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 106–115.
- [53] K. Hahn, C. Sapia, and M. Blaschka, "Automatically generating OLAP schemata from conceptual graphical models," in *DOLAP '00: Proceedings of the 3rd ACM International Workshop on Data Warehousing and OLAP*. New York, NY, USA: ACM, 2000, pp. 9–16.
- [54] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed., ser. The Morgan Kaufmann Series in Data Management Systems. San Francisco, CA, USA: Morgan Kaufmann, 2006.
- [55] P. Hanrahan, C. Stolte, and J. Mackinlay, "Visual analysis for everyone: Understanding data exploration and visualization," *Tableau Software Inc.*, 2007, tableau White Paper. [Online]. Available: http://www.tableausoftware.com/docs/Tableau_Whitepaper.pdf
- [56] G. Henderson, "OLAP best practices - what you need to consider when building and deploying an OLAP application," in *SUGI 26: Proceedings of the 26th SAS Users Group International Conference*, 2001, paper 37.
- [57] O. Herden, "A design methodology for data warehouses," in *Proceedings of the 7th Doctoral Consortium on Advanced Information Systems Engineering, co-located with CAiSE*00*, 2000, pp. 13–23.

- [58] I. Herman, G. Melançon, and M. S. Marshall, “Graph visualization and navigation in information visualization: A survey,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 24–43, 2000.
- [59] C. A. Hurtado, C. Gutierrez, and A. O. Mendelzon, “Capturing summarizability with integrity constraints in OLAP,” *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 3, pp. 854–886, 2005.
- [60] C. A. Hurtado and A. O. Mendelzon, “Reasoning about summarizability in heterogeneous multidimensional schemas,” in *ICDT 2001: Proceedings of the 8th International Conference on Database Theory*, 2001, pp. 375–389.
- [61] C. A. Hurtado and A. O. Mendelzon, “OLAP dimension constraints,” in *PODS ’02: Proceedings of the 21st ACM Symposium on Principles of Database Systems*, 2002, pp. 169–179.
- [62] T. N. Huynh, O. Mangisengi, and A. M. Tjoa, “Metadata for object-relational data warehouse,” in *DMDW’2000: Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses at CAiSE’00*, ser. CEUR Workshop Proceedings, vol. 28. CEUR-WS.org, 2000, pp. 3.1–3.9.
- [63] W. Hümmer, W. Lehner, A. Bauer, and L. Schlesinger, “A decathlon in multidimensional modeling: Open issues and some solutions,” in *DaWaK 2002: Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, London, UK, 2002, pp. 275–285.
- [64] B. Husemann, J. Lechtenböcker, and G. Vossen, “Conceptual data warehouse design,” in *DMDW’2000: Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses at CAiSE’00*, ser. CEUR Workshop Proceedings, vol. 28. CEUR-WS.org, 2000, pp. 6.1–6.11.
- [65] M. Inc., “Relational OLAP: An enterprise-wide data delivery architecture,” 1994, White Paper.
- [66] M. Inc., “The case for relational OLAP,” 1995, White Paper.
- [67] B. Inmon, “Data mart does not equal data warehouse,” *DM Direct*, Nov. 1999. [Online]. Available: http://www.dmreview.com/article_sub.cfm?articleId=1675
- [68] W. H. Inmon, *Building the Data Warehouse*, 3rd ed. New York, NY, USA: John Wiley & Sons, Inc., 2002.
- [69] A. Inselberg, “The plane with parallel coordinates,” *The Visual Computer (Special issue on Computational Geometry)*, vol. 1, no. 2, pp. 69–91, 1985.
- [70] H. V. Jagadish, L. V. S. Lakshmanan, and D. Srivastava, “What can hierarchies do for data warehouses?” in *VLDB ’99: Proceedings of 25th International Conference on Very Large Data Bases*, 1999, pp. 530–541.
- [71] E. Jank, A. Rose, S. Huth, C. Trantakis, W. Korb, G. Strauss, J. Meixensberger, and J. Krüger, “A new fluoroscopy based navigation system for milling procedures in spine surgery,” *International Journal for Computer Assisted Radiology and Surgery*, vol. 1, no. Supplement 1, pp. 196–198, 2006.
- [72] M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis, *Fundamentals of Data Warehouses*, 2nd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.

- [73] C. S. Jensen, A. Kligys, T. B. Pedersen, and I. Timko, "Multidimensional data modeling for location-based services," in *GIS '02: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*. New York, NY, USA: ACM, 2002, pp. 55–61.
- [74] M. E. Jones and I.-Y. Song, "Dimensional modeling: identifying, classifying & applying patterns," in *DOLAP '05: Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP*. New York, NY, USA: ACM, 2005, pp. 29–38.
- [75] D. A. Keim and H.-P. Krigel, "VisDB: Database exploration using multidimensional visualization," *IEEE Computer Graphics and Applications*, vol. 14, no. 5, pp. 40–49, 1994.
- [76] D. Keim, M. Hao, U. Dayal, M. Hsu, and J. Ladisch, "Pixel bar charts: A new technique for visualizing large multi-attribute data sets without aggregation," in *INFOVIS '01: Proceedings of the 2001 IEEE Symposium on Information Visualization*. Los Alamitos, CA, USA: IEEE Computer Society, 2001, pp. 113–122.
- [77] D. A. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1–8, 2002.
- [78] D. A. Keim, M. Ankerst, and H.-P. Kriegel, "Recursive pattern: A technique for visualizing very large amounts of data," in *VIS '95: Proceedings of the 6th Conference on Visualization*. Washington, DC, USA: IEEE Computer Society, 1995, p. 279.
- [79] D. A. Keim, F. Mansmann, C. Panse, J. Schneidewind, and M. Sips, "Mail explorer - spatial and temporal exploration of electronic mail," in *EuroVis05: Proceedings of Joint Eurographics - IEEE VGTC Symposium on Visualization*. Eurographics Association, 2005, pp. 247–254.
- [80] H.-G. Kemper, W. Mehanna, and C. Unger, *Business Intelligence - Grundlagen und praktische Anwendungen*, 2nd ed. Wiesbaden, Germany: Vieweg, 2006.
- [81] R. Kimball, *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. New York, NY, USA: John Wiley & Sons, Inc., 1996.
- [82] R. Kimball, "Drilling down, up, and across," *DBMS*, vol. 9, no. 3, 1996.
- [83] R. Kimball, "Factless fact tables," *DBMS*, vol. 9, no. 10, 1996.
- [84] R. Kimball, "Relocating the ODS: moving the operational data store will solve a number of problems," *DBMS*, vol. 10, no. 13, 1997.
- [85] R. Kimball, "Dimensional vs. relational olap: The final deployment conundrum," *Intelligent Enterprise*, Apr. 2007.
- [86] R. Kimball, L. Reeves, M. Ross, and W. Thornwaite, *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses*. New York, NY, USA: John Wiley & Sons, Inc., 1998.
- [87] R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. New York, NY, USA: John Wiley & Sons, Inc., 2002.
- [88] O. A. Kuchar, T. J. Hoeft, S. Havre, and K. A. Perrine, "Isn't it about time?" *IEEE Computer Graphics and Applications*, vol. 26, no. 3, pp. 80–83, 2006.

- [89] A. Kurz, *Data Warehousing Enabling Technology*. Bonn, Germany: MITP-Verlag, 1999.
- [90] W. J. Labio, D. Quass, and B. Adelberg, "Physical database design for data warehouses," in *ICDE '97: Proceedings of the 13th International Conference on Data Engineering*. Los Alamitos, CA, USA: IEEE Computer Society, 1997, pp. 277–288.
- [91] A. Laurent, "Generating fuzzy summaries from fuzzy multidimensional databases," in *IDA '01: Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, 2001, pp. 24–33.
- [92] J. Lechtenbörger, "Data Warehouse Schema Design," Ph.D. dissertation, Westfälische Wilhelms-Universität Münster, 2001.
- [93] J. Lechtenbörger, "Data warehouse schema design," in *BTW 2003: 10. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web*, ser. LNI. Leipzig, Germany: GI, 2003, pp. 513–522.
- [94] J. Lechtenbörger and G. Vossen, "Multidimensional normal forms for data warehouse design," *Information Systems*, vol. 28, no. 5, pp. 415–434, 2003.
- [95] H.-Y. Lee and H.-L. Ong, "A new visualisation technique for knowledge discovery in OLAP," in *PAKDD'97: Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 1997, pp. 279–286.
- [96] W. Lehner, J. Albrecht, and H. Wedekind, "Normal forms for multidimensional databases," in *SSDBM: Proceedings of 10th International Conference on Scientific and Statistical Database Management*, 1998, pp. 63–72.
- [97] W. Lehner, "Modelling large scale OLAP scenarios," in *EDBT '98: Proceedings of the 6th International Conference on Extending Database Technology*. London, UK: Springer, 1998, pp. 153–167.
- [98] H.-J. Lenz and A. Shoshani, "Summarizability in OLAP and statistical data bases," in *Proceedings of 9th International Conference on Scientific and Statistical Database Management*, 1997, pp. 132–143.
- [99] C. Li and X. S. Wang, "A data model for supporting on-line analytical processing," in *CIKM '96: Proceedings of the 5th International Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 1996, pp. 81–88.
- [100] H. Li, H. Huang, and Y. Lin, "A new incremental maintenance algorithm of data cube," in *RSFDGrC 2003: Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, ser. LNCS, vol. 2639. Springer, 2003, pp. 499–506.
- [101] W.-Y. Lin, C.-A. Wu, and C.-C. Wu, "An object-relational data warehouse modelig for complex data," in *JCIS 2006: Proceedings of the Joint Conference on Information Sciences*. Atlantis Press, 2006.
- [102] S. Luján-Mora, "Data warehouse design with uml," Ph.D. dissertation, Universidad de Alicante, June 2005.
- [103] S. Luján-Mora and J. Trujillo, "Physical modeling of data warehouses using uml," in *DOLAP '04: Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP*. New York, NY, USA: ACM Press, 2004, pp. 48–57.

- [104] S. Luján-Mora, J. Trujillo, and I.-Y. Song, "A uml profile for multidimensional modeling in data warehouses," *Data & Knowledge Engineering*, vol. 59, pp. 725–769, December 2006.
- [105] J. Mackinlay, "Automating the design of graphical presentations of relational information," *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 110–141, 1986.
- [106] J. Mackinlay, P. Hanrahan, and C. Stolte, "Show me: Automatic presentation for visual analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1137–1144, 2007.
- [107] A. Makinouchi, "A consideration on normal form of not-necessarily-normalized relation in the relational data model," in *VLDB'77: Proceedings of the 3rd International Conference on Very Large Data Bases*. IEEE Computer Society, 1977, pp. 447–453.
- [108] E. Malinowski and E. Zimányi, "OLAP hierarchies: A conceptual perspective," in *CAiSE'04: Proceedings of the 16th International Conference on Advanced Information Systems Engineering*, 2004, pp. 477–491.
- [109] E. Malinowski and E. Zimányi, "Hierarchies in a multidimensional model: From conceptual modeling to logical representation," *Data & Knowledge Engineering*, vol. 59, no. 2, pp. 348–377, 2006.
- [110] O. Mangisengi, A. M. Tjoa, and R. Wagner, "Multidimensional modeling approaches for OLAP based on extended relational concepts," in *IDC'99: Proceedings of the 9th International Database Conference on Heterogeneous and Internet Databases*, 1999.
- [111] A. S. Maniatis, P. Vassiliadis, S. Skiadopoulou, and Y. Vassiliou, "Advanced visualization for OLAP," in *DOLAP '03: Proc. of 6th ACM Int. Workshop on Data Warehousing and OLAP*, 2003, pp. 9–16.
- [112] F. Mansmann and S. Vinnik, "Interactive Exploration of Data Traffic with Hierarchical Network Maps," *IEEE Transactions on Visualization and Computer Graphics (Special Issue on Visual Analytics)*, vol. 12, no. 6, pp. 1440–1449, 2006.
- [113] S. Mansmann, T. Neumuth, O. Burgert, M. Röger, and M. H. Scholl, "Conceptual Data Warehouse Design Methodology for Business Process Intelligence," in *Complex Data Warehousing and Knowledge Discovery for Advanced Retrieval Development: Innovative Methods and Applications*, ser. Advances in Data Warehousing and Mining (ADWM). IGI Publishing, 2008, (Invited book chapter, to appear).
- [114] S. Mansmann, T. Neumuth, and M. H. Scholl, "Multidimensional Data Modeling for Business Process Analysis," in *ER 2007: Proceedings of the 26th International Conference on Conceptual Modeling*, ser. LNCS. Auckland, New Zealand: Springer, 2007, pp. 23–38.
- [115] S. Mansmann, T. Neumuth, and M. H. Scholl, "OLAP Technology for Business Process Intelligence: Challenges and Solutions," in *DaWaK'07: Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery*, ser. LNCS. Springer, 2007, pp. 111–122, DaWaK'07 Best Paper.
- [116] S. Mansmann and M. H. Scholl, "Extending Visual OLAP for Handling Irregular Dimensional Hierarchies," in *DaWaK'06: Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery*, ser. LNCS. Springer, 2006, pp. 95–105, DaWaK'06 Best Paper.
- [117] S. Mansmann and M. H. Scholl, "Decision Support System for Managing Educational Capacity Utilization," *IEEE Transactions on Education*, vol. 50, no. 2, pp. 143–150, 2007.

- [118] S. Mansmann and M. H. Scholl, "Empowering the OLAP Technology to Support Complex Dimension Hierarchies," *International Journal of Data Warehousing and Mining*, vol. 3, no. 4, pp. 31–50, 2007, (Invited Paper).
- [119] S. Mansmann and M. H. Scholl, "Exploring OLAP Aggregates with Hierarchical Visualization Techniques," in *SAC 2007: Proceedings of 22nd Annual ACM Symposium on Applied Computing, Multimedia & Visualization Track*. New York, NY, USA: ACM Press, 2007, pp. 1067–1073.
- [120] S. Mansmann and M. H. Scholl, "Extending the Multidimensional Data Model to Handle Complex Data," *Journal of Computing Science and Engineering*, vol. 1, no. 2, pp. 125–160, 2008, (Invited tutorial paper).
- [121] MDC (Meta Data Coalition), "Open Information Model, version 1.1," Aug. 1999.
- [122] E. Medina and J. Trujillo, "A standard for representing multidimensional properties: The common warehouse metamodel (CWM)," in *ADBIS 2002: Proceedings of the 6th East European Conference on Advances in Databases and Information Systems*, ser. LNCS, vol. 2435. Berlin / Heidelberg, Germany: Springer, 2002, pp. 232–247.
- [123] MicroStrategy, Inc, "MicroStrategy OLAP Services." [Online]. Available: http://www.microstrategy.com/Software/Products/Service_Modules/OLAP_Services/
- [124] Miner3D, Inc., "Miner3D: Exploratory data analysis for business intelligence and science." [Online]. Available: <http://www.miner3d.com/products/>
- [125] D. L. Moody and M. A. R. Kortink, "From enterprise models to dimensional models: a methodology for data warehouse and data mart design," in *DMDW 2000: Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses*, 2000, pp. 5.1–5.5.
- [126] T. Neumuth, N. Durstewitz, M. Fischer, G. Strauss, A. Dietz, J. Meixensberger, P. Jannin, K. Cleary, H. U. Lemke, and O. Burgert, "Structured recording of intraoperative surgical workflows," in *Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, 2006, pp. 54–65.
- [127] T. Neumuth, S. Mansmann, M. H. Scholl, and O. Burgert, "Data Warehousing Technology for Surgical Workflow Analysis," in *CBMS 2008: Proceedings of the 21st IEEE International Symposium on Computer-Based Medical Systems*. Jyväskylä, Finland: IEEE Computer Society, 2008, pp. 230–235.
- [128] T. Neumuth, S. Schumann, G. Strauß, P. Jannin, J. Meixensberger, A. Dietz, H. U. Lemke, and O. Burgert, "Visualization options for surgical workflows," *International Journal of Computer Assisted Radiology and Surgery*, vol. 1, no. 1, pp. 438–440, 2006.
- [129] T. Neumuth, G. Strauß, J. Meixensberger, H. U. Lemke, and O. Burgert, "Acquisition of process descriptions from surgical interventions," in *DEXA 2006: Proceedings of the 17th International Conference on Database and Expert Systems Applications*. Berlin, Germany: Springer, 2006, pp. 602–611.
- [130] T. Neumuth, C. Trantakis, F. Eckhardt, M. Dengl, J. Meixensberger, and O. Burgert, "Supporting the analysis of intervention courses with surgical process models on the example of fourteen microsurgical lumbar discectomies," *International Journal of Computer Assisted Radiology and Surgery*, vol. 2, no. Supplement 1, pp. 436–438, 2007.

- [131] T. B. Nguyen, A. M. Tjoa, and R. Wagner, "An object oriented multidimensional data model for OLAP," in *WAIM '00: Proceedings of the 1st International Conference on Web-Age Information Management*. London, UK: Springer, 2000, pp. 69–82.
- [132] T. Niemi, J. Nummenmaa, and P. Thanisch, "Logical multidimensional database design for ragged and unbalanced aggregation," in *Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses*, 2001, pp. 7.1–7.8.
- [133] "Oracle Business Intelligence Foundation and Tools," 2007. [Online]. Available: <http://www.oracle.com/appserver/business-intelligence/>
- [134] "OLAP and OLAP Server Definitions: OLAP Glossary," OLAP Council. [Online]. Available: <http://www.olapcouncil.org/research/glossary.htm>
- [135] OMG (Object Management Group), "Common Warehouse Metamodel TM Specification, version 1.1," 2003, (OMG document). [Online]. Available: <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-02.pdf>
- [136] OMG (Object Management Group), "Unified Modeling Language Specification, version 1.4.2," 2005, (OMG document). [Online]. Available: <http://www.omg.org/docs/formal/05-04-01.pdf>
- [137] OMG (Object Management Group), "BPMN (Business Process Modeling Notation) 1.0: OMG Final Adopted Specification," Feb. 2006. [Online]. Available: <http://www.bpmn.org>
- [138] K. Orr, "Data warehousing technology," The Ken Orr Institute, 1996, White Paper.
- [139] "Pentaho BI Platform." [Online]. Available: http://www.pentaho.com/products/bi_platform/
- [140] V. K. Pang-Ning Tan, Michael Steinbach, *Introduction to Data Mining: Concepts and Techniques*. Addison Wesley, 2006.
- [141] W. Pearson, "Introduction to MSSQL Server 2000 Analysis Services: Reporting options for analysis services cubes: ProClarity Professional, Part I," *Database Journal*, Jan. 2006, article 19. [Online]. Available: http://www.databasejournal.com/features/mssql/article.php/10894_3300131_4
- [142] T. B. Pedersen and C. S. Jensen, "Multidimensional data modeling for complex data," in *ICDE'99: Proceedings of 15th International Conference on Data Engineering*, 1999, pp. 336–345.
- [143] T. B. Pedersen and C. S. Jensen, "Multidimensional database technology," *IEEE Computer*, vol. 34, no. 12, pp. 40–46, 2001.
- [144] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson, "Extending practical pre-aggregation in on-line analytical processing," in *VLDB'99: Proceedings of 25th International Conference on Very Large Data Bases*, 1999, pp. 663–674.
- [145] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson, "A foundation for capturing and querying complex multidimensional data," *Information Systems*, vol. 26, no. 5, pp. 383–423, 2001.
- [146] C. Phipps and K. C. Davis, "Automating data warehouse conceptual schema design and evaluation," in *DMDW'2002: Proceedings of the 4th International Workshop on Design and Management of Data Warehouses*, ser. CEUR Workshop Proceedings, vol. 58. CEUR-WS.org, 2002, pp. 23–32.

- [147] J. M. Poole, D. Mellor, D. Chang, and D. Tolbert, *Common Warehouse Metamodel Developer's Guide*, ser. OMG Series. Wiley, John & Sons, Inc., 2003.
- [148] E. Pourabbas and M. Rafanelli, "Characterization of hierarchies and some operators in OLAP environment," in *DOLAP'99: Proceedings of the 2nd ACM International Workshop on Data Warehousing and OLAP*. New York, NY, USA: ACM Press, 1999, pp. 54–59.
- [149] N. Prat, J. Akoka, and I. Comyn-Wattiau, "A UML-based data warehouse design method," *Decision Support Systems*, vol. 42, no. 3, pp. 1449–1473, 2006.
- [150] ProClarity Analytics, "ProClarity for reporting services (product fact sheet)," 2006. [Online]. Available: <http://download.microsoft.com/download/0/4/9/04930661-7a6f-46db-9281-e92ebe6891bc/ProClarity%20for%20Reporting%20Services.pdf>
- [151] J. Propach and S. Reuse, "Data Warehouse: ein 5-Schichten-Modell," *WISU - das Wirtschaftsstudium*, vol. 32, no. 1, pp. 98–106, Jan. 2003.
- [152] M. Rafanelli and A. Shoshani, "STORM: A statistical object representation model," in *Proceedings of the 5th International Conference on Statistical and Scientific Database Management*, 1990, pp. 14–29.
- [153] F. Ravat, O. Teste, R. Tournier, and G. Zurfluh, "A conceptual model for multidimensional analysis of documents," in *ER 2007: Proceedings of the 26th International Conference on Conceptual Modeling*, ser. LNCS, vol. 4801. Auckland, New Zealand: Springer, 2007, pp. 550–565.
- [154] E. Reingold and J. Tilford, "Tidier drawing of trees," *IEEE Transactions on Software Engineering*, vol. 7, pp. 223–228, 1981.
- [155] S. Rivest, Y. Bédard, and P. Marchand, "Towards better support for spatial decision-making: Defining the characteristics of Spatial On-Line Analytical Processing (SOLAP)," *Geomatica*, vol. 55, no. 4, pp. 539–555, 2001.
- [156] S. Rizzi, A. Abelló, J. Lechtenbörger, and J. Trujillo, "Research in data warehouse modeling and design: dead or alive?" in *DOLAP '06: Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP*. New York, NY, USA: ACM Press, 2006, pp. 3–10.
- [157] H. Rosling, A. Rosling Rönnlund, and O. Rosling, *New software brings statistics beyond the eye*. Organisation for Economic Co-operation and Development, 2006, pp. 522–530. [Online]. Available: <http://www.gapminder.org>
- [158] P. Russom, "Trends in data visualization software for business users," *DM Review*, May 2000. [Online]. Available: http://www.dmreview.com/article_sub.cfm?articleId=1675
- [159] C. Sapia, M. Blaschka, G. Höfling, and B. Dinter, "Extending the E/R model for the multidimensional paradigm," in *ER '98: Proceedings of the Workshops on Data Warehousing and Data Mining*. London, UK: Springer, 1999, pp. 105–116.
- [160] M. H. Scholl and S. Mansmann, "Visual OLAP (Online Analytical Processing)," in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds. Springer, 2008, (To appear).
- [161] R. Schütte, T. Rotthowe, and R. Holten, Eds., *Data Warehouse Managementhandbuch: Konzepte, Software, Erfahrungen*. Berlin, Germany: Springer, 2001.

- [162] J. Shanmugasundaram, U. Fayyad, and P. S. Bradley, "Compressed data cubes for OLAP aggregate query approximation on continuous dimensions," in *SIGKDD '99: Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 1999, pp. 223–232.
- [163] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Transactions on Graphics*, vol. 11, no. 1, pp. 92–99, 1992.
- [164] M. Sifer, "A visual interface technique for exploring olap data with coordinated dimension hierarchies," in *CIKM '03: Proceedings of the 12th International Conference on Information and Knowledge Management*. New York, NY, USA: ACM Press, 2003, pp. 532–535.
- [165] E. Soler, R. Villarroel, J. Trujillo, E. Fernandez-Medina, and M. Piattini, "Representing security and audit rules for data warehouses at the logical level by using the Common Warehouse Metamodel," in *ARES'06: Proceedings of the 1st International Conference on Availability, Reliability and Security*. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 914–921.
- [166] I.-Y. Song, W. Rowen, C. Medsker, and E. F. Ewen, "An analysis of many-to-many relationships between fact and dimension tables in dimensional modeling," in *DMDW'01: Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses*, ser. CEUR Workshop Proceedings, vol. 39. CEUR-WS.org, 2001, pp. 6.1–6.13.
- [167] C. Stolte, D. Tang, and P. Hanrahan, "Query, analysis, and visualization of hierarchically structured data using Polaris," in *ACM SIGKDD '02: Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 112–122.
- [168] C. Stolte, D. Tang, and P. Hanrahan, "Multiscale visualization using data cubes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 176–187, 2003.
- [169] "Tableau Software." [Online]. Available: <http://www.tableausoftware.com>
- [170] K. Techapichetvanich and A. Datta, "Interactive visualization for OLAP," in *ICCSA 2005: Proceedings of the International Conference on Computational Science and its Applications (Part III)*, 2005, pp. 206–214.
- [171] TechTarget, "Business Intelligence," SearchDataManagement.com Definitions (Powered by WhatIs.com). [Online]. Available: http://searchdatamanagement.techtarget.com/sDefinition/0,,sid91_gci213571,00.html
- [172] D. P. Tegarden, "Business information visualization," *Communications of the AIS*, vol. 1, no. 1, 1999, article 4.
- [173] A. Totok, "Modellierung von OLAP- und Data-Warehouse-Systemen," Ph.D. dissertation, Technische Universität Braunschweig, 2000.
- [174] J. Trujillo, M. Palomar, J. Gomez, and I.-Y. Song, "Designing data warehouses with OO conceptual models," *Computer*, vol. 34, no. 12, pp. 66–75, 2001.
- [175] N. Tryfona, F. Busborg, and J. G. B. Christiansen, "Starer: a conceptual model for data warehouse design," in *DOLAP '99: Proceedings of the 2nd ACM International Workshop on Data Warehousing and OLAP*. New York, NY, USA: ACM Press, 1999, pp. 3–8.

- [176] E. R. Tufte, *The visual display of quantitative information*. Cheshire, CT, USA: Graphics Press, 1986.
- [177] J. W. Tukey, *Exploratory data analysis*. Reading, MA, USA: Addison-Wesley, 1977.
- [178] E.-J. van der Linden, "Comparison of magnaview to treemapping tools and visual analysis tools," 2006, magnaView White Paper 2006-551. [Online]. Available: <http://www.magnaview.nl/documents/MagnaView%20Memo%20comparison%20treemap%20tools.pdf>
- [179] P. Vassiliadis, "Modeling multidimensional databases, cubes and cube operations," in *SSDBM '98: Proceedings of the 10th International Conference on Scientific and Statistical Database Management*. Washington DC, USA: IEEE Computer Society, 1998, pp. 53–62.
- [180] P. Vassiliadis and T. Sellis, "A survey of logical models for OLAP databases," *ACM SIGMOD Record*, vol. 28, no. 4, pp. 64–69, 1999.
- [181] I. Vessey, "Cognitive fit: A theory-based analysis of the graphs versus tables literature," *Decision Sciences*, vol. 22, no. 2, pp. 219–240, 1991.
- [182] S. Vinnik and F. Mansmann, "From Analysis to Exploration: Building Enhanced Visual Hierarchies from OLAP Cubes," in *EDBT 2006: Proceedings of the 10th International Conference on Extending Database Technology*, ser. LNCS. Springer, 2006, pp. 496–514.
- [183] S. Vinnik and M. H. Scholl, "Decision Support System for Managing Educational Capacity Utilization in Universities," in *Proceedings of the International Conference on Engineering and Computer Education (ICECE'05)*, Madrid, Spain, 2005, ICECE'05 Best Paper.
- [184] S. Vinnik and M. H. Scholl, "UNICAP: Efficient Decision Support for Academic Resource and Capacity Management," in *TCGOV2005: Proceedings of the TED Conference on e-Government*, ser. LNAI. Bozen-Bolzano, Italy: Springer, 2005.
- [185] R. Vliegen, J. J. van Wijk, and E.-J. van der Linden, "Visualizing business data with generalized treemaps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 789–796, 2006.
- [186] J. J. V. Wijk and E. R. V. Selow, "Cluster and calendar based visualization of time series data," in *INFOVIS '99: Proceedings of the 1999 IEEE Symposium on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 4–9.
- [187] Wikimedia Commons, "Category: Vertebra," in *Wikimedia Commons*. Wikimedia Foundation, Inc., 2007. [Online]. Available: <http://commons.wikimedia.org/wiki/Category:Vertebra>
- [188] XMLA Consulting, "Report Portal: Zero-footprint OLAP web client solution for Microsoft Analysis Services." [Online]. Available: <http://www.reportportal.com>
- [189] J. Yang, M. O. Ward, E. A. Rundensteiner, and A. Patro, "InterRing: a visual interface for navigating and manipulating hierarchies," *Information Visualization*, vol. 2, no. 1, pp. 16–30, 2003.
- [190] T. Zurek and M. Sinnwell, "Datawarehousing has more colours than just black & white," in *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, 1999, pp. 726–729.

Index

-
- A**
- ADAPT 26
- additivity 17, 27, 38, 97
- Advizor 34, 169, 176
- aggregability 96
- constraint 182
- aggregate function 17, 33, 42, 96, 181–182, 196
- algebraic 17
- default 182
- distributive 17, 36, 96
- holistic 17
- non-cumulative 196
- self-maintainability 17
- standard set 96
- aggregation statement 96
- analysis criterion 15, 66, 84, 85
- antireflexivity 53–54
- antisymmetry 53–54
- association fact 100, 103, 119
- association relationship 45, 46
- optional 45, 46
- attribute 15, 25, 32, 114, 115
- composite 115, 116
- derived 26, 135
- multivalued 115, 116
- optional 75, 94, 140
- single-valued 114, 115
- visual 32, 171, 172
-
- B**
- bridge table 132, 152
- Business Intelligence (BI) 9
- applications 10
- definition 10
- levels 10
- operational 113
- perspectives 10
- platform 13, 90
- tools 13, 176
- BusinessObjects Enterprise 13
-
- C**
- candidate fact 114, 118, 119
- terminal 118
- cardinality 68, 69, 114, 115
- constraint 114, 118
- categorization of
- dimension and hierarchy types 66
- dimension sharing patterns 106
- dimension types 4, 68
- fact and multi-fact types 95
- fact types 4
- hierarchy types 4, 37, 38, 62, 71
- by strictness 71
- heterogeneous 75
- homogeneous 73
- measure types 95
- multi-fact types 4, 99
- multiple hierarchies 84
- category 15, 36, 39, 44, 54, 56, 114, 178, 201
- abstract 44, 45, 53, 77
- bottom-level 22, 164
- derived 45, 70, 94, 135
- generalized 76–83, 122, 146, 147
- non-covering 137
- top-level 43, 45, 57
- totally ordered 43–45, 55
- category type 54, 58, 74, 106, 121, 132, 201, 204
- shared 55, 57, 83, 105, 106
- classification hierarchy *see* hierarchy
- classification level *see* category
- classification scheme *see* dimension scheme
- CoDecide 169
- Common Warehouse Metamodel (CWM) .. 155–156
- layers 155–156

- OLAP package 156–158
- packages 156
- Transformation package 158–159
- compatible categories 56, 57, 105, 106, 123, 132
- compatible dimensions 57
- compatible fact schemes 27
- conform categories 56, 74, 83, 105, 106
- conform dimensions 50, 52, 107
- connection layout 194
- connectivity 55
- coverage fact 16, 98
- cross tab *see* pivot table
- cube 2, 14, 38, 56, 60, 174
 - computation 103
 - data model 60
 - operator 60
 - virtual 185, 186
- Cube Presentation Model (CPM) 171

D

- data cube *see* cube
- data mart 12
- data mining 2, 10, 13, 96, 155, 169
 - techniques 13
- data warehouse
 - centralized 12
 - characteristics 11
 - components 11
 - definition 11
 - departmental 12
 - reference architecture 2, 3, 6, 11, 133, 155
 - Analysis Layer 2, 6, 13, 162
 - Application Layer 2, 4
 - Data Sources Layer 11
 - Data Warehouse Layer 12
 - ETL Layer 2, 11
 - Presentation Layer 2–4, 10, 13, 29
 - Transformation Layer 2
 - system 11
- data warehouse design 2, 24, 111
 - conceptual 2, 25, 112
 - phases 24
 - physical 28, 134
 - relational 28, 132
- data warehousing 1, 2, 5, 11
 - characteristics 13

- decomposition tree 188–191, 194, 197, 199
 - space-filling 195–197
 - spatial 200
 - temporal 199
- derivation relationship 46, 62
- derived element 43, 45, 46, 134
- derived table 139, 140
- dimension 1, 14–15, 45, 49, 50, 53, 56, 95, 178
 - “frozen” 75
 - ad hoc 40, 104, 183
 - continuous 183
 - degenerated 44, 69, 94, 99, 134, 164
 - derived 44, 69, 94
 - heterogeneous 61, 75
 - hierarchical 70
 - included 107
 - non-hierarchical 54, 70, 122
 - optional 49, 69
 - shared 39, 40, 48, 50, 162
 - totally ordered 44, 50, 55
- dimension attribute 41, 42, 45, 134
- dimension conformance 106
- dimension extension *see* dimension instance
- dimension hierarchy *see* hierarchy
- dimension instance 54, 67
- dimension level *see* category
- dimension level attribute 22, 39, 58, 115
- dimension member 14, 15
- dimension scheme 53, 54, 67, 68, 121
- dimension sharing 39, 50, 105, 106
 - convergence 106, 110
 - fact-as-dimension 106–108
 - full 22, 106
 - inclusion 106, 107
 - overlap 106, 108–110
 - partial 22, 39, 106, 107
- dimension table 22, 132, 135
 - centralized 132
 - de-normalized 22
 - normalized 22
 - shared 22
- dimension type 49, 50, 202
 - shared 52
- dimensional design pattern 86
- Dimensional Fact Model (DFM) . 27, 41–43, 48, 112
- discretization 183
- disjointness 55, 56, 82, 141

E

elementary perceptual task 174
 Enhanced Decomposition Tree 5, 191, 194
 Entity-Relationship (E/R) model ... 25, 62, 112, 114, 115
 – extensions 25, 112
 Extended Dimensional Fact Model *see* x-DFM
 Extract, Transform, Load (ETL) 1, 2, 10–12, 65, 154

F

fact ... 14, 16–17, 38, 42, 44, 49, 53, 57, 95, 98, 118
 – accumulating (cumulative) snapshot 16, 95
 – complex 38
 – degenerate 44, 45, 94, 118, 119, 134, 162
 – definition 100
 – derived 49, 134
 – event-tracking 16, 98, 99
 – measurable 1, 3, 14, 16, 95, 111, 114, 133
 – non-measurable (factless) ... 16, 95, 98, 114, 123, 133, 162, 182, 183
 – periodic snapshot 16, 95
 – primary 98, 103, 134
 – secondary 98, 103
 – transactional 16, 95, 97
 fact cluster 51, 57, 58, 95
 fact constellation schema 23
 fact derivation 103, 134, 135
 fact family 51, 52, 56
 fact generalization 102
 fact hierarchy *see* fact roll-up
 fact identifier 44, 69, 94, 99, 103, 133
 – surrogate 133
 fact roll-up 101, 103, 118, 162
 – direct 101
 – transitive 101
 fact scheme 27, 49, 114, 201
 – heterogeneous 102, 103
 – homogeneous 102, 103
 fact specialization 102
 fact table 22–23, 98, 132–135, 140, 158, 162
 – factless 98
 foreign key 22, 29, 63, 132, 134, 135, 140

G

galaxy schema 22–23, 132, 139
 galaxy scheme 28, 48, 56, 185

Gapminder 171
 generalization hierarchy 61, 80
 generalization/specialization relationship 45, 46, 75–77, 114, 115, 117
 Generalized Multidimensional Normal Form (GMNF) 61
 granularity 16, 53, 60, 101

H

heterogeneity 38, 61, 101, 102
 hierarchical domain 67
 Hierarchical Dynamic Dimensional Visualization 171
 Hierarchical HeatMap 196
 Hierarchical Network Map (HNMap) 200
 hierarchy 2, 39, 66, 114, 121, 132, 178, 180
 – ad hoc 40
 – complete 39, 74
 – covering 149
 – fuzzy 73, 204
 – generalized 76–77, 82, 136, 141
 – heterogeneous .. 37, 45, 46, 53, 61, 64, 68, 71, 75, 114, 122, 136, 177
 – homogeneous 71, 73, 114, 122
 – mixed-grain 82–83, 136, 141, 143, 146
 – non-covering 45, 75–77, 122, 136–140, 149
 – non-onto (asymmetric) 73–74, 149, 153
 – non-strict .. 36, 40, 45, 71, 72, 118, 122, 132, 136, 149, 150
 – weighted 72, 152
 – non-summarizable . 60, 65, 72, 74, 77, 80, 82, 136
 – onto (symmetric) 73, 153
 – optional 75, 122
 – simple 122
 – strict 71, 150–152
 – summarizable 73, 77, 83, 148, 202
 – totally ordered 39, 43, 55
 hierarchy instance 67, 75
 hierarchy scheme 67–68
 hierarchy types 27, 61, 66, 202
 Higher Education Information System (HIS) Ltd. 62, 63, 65
 homogeneity 2, 36, 37, 60, 61, 71
 Hybrid OLAP (HOLAP) 21

I

ICCAS (Innovation Center Computer Assisted Surgery) 90, 93, 123

inheritance 36, 62, 76, 80, 101, 102, 136, 139
 inheritance hierarchy 77–78, 143, 146
 integrity constraint 26, 29, 37, 61, 75, 132, 140
 inter-stellar scheme *see* galaxy scheme
 interaction techniques 31, 32, 169, 171, 172, 187
 Intermediate Level (IL) model 47, 53–57
 InterRing 194

J

joining level 76, 77
 Joolap 65

L

Lower Level (LL) model 47, 57–58

M

MagnaView Explorer 34
 managed query environment 30
 many-to-many relationship 17, 36, 38, 43, 45, 46,
 71, 86, 90, 94, 98, 100, 118, 122, 150, 152,
 201
 – fact-dimensional 39, 44
 many-to-one relationship 42, 69, 71, 72
 materialized view 132, 134, 135, 140
 measure 1, 14, 17, 36, 38, 43, 45, 49, 95, 181
 – ad hoc 3, 40, 104, 182
 – additive 16, 17, 27, 97
 – aggregable 97
 – complex 38
 – default 181, 182
 – derived 17, 38, 44, 45, 49, 96, 135, 182
 – flow 97
 – multicube 185
 – non-additive 16, 17, 42, 97, 164
 – primary 96, 182
 – semi-additive 17, 97
 – stock (snapshot) 97
 – – accumulating 97
 – – periodic 97
 – value-per-unit 97, 182
 measure attribute 32, 40, 44, 57, 96, 114, 182
 measure-centrism 112
 Meta Data Coalition (MDC) 155
 Meta Object Facility (MOF) 155
 metacube 25
 metadata 4, 13, 33, 133, 154–155, 204

 – administrative 13
 – business 13
 – classification 154–155
 – model 5
 – operational 13
 – repository 12, 133
 multi-fact scheme *see also* galaxy scheme, 39, 56, 99
 multi-star scheme 48, 50
 multicube 40, 184, 186
 multicube join *see* OLAP operator DRILL-ACROSS
 multicube navigation 184, 185
 multidimensional data model 1, 4, 14, 115, 168
 – constraints 36
 – elements 14–17
 Multidimensional Entity Relationship (ME/R) Model
 25, 112
 Multidimensional Modeling Language (MML) 26
 Multidimensional OLAP (MOLAP) 21, 27, 28, 158
 multidimensional pattern 51
 multidimensional property 37–41, 52, 57, 66, 156,
 162–165, 201
 – dynamic 39–40, 89, 204
 – static 38–39
 multidimensional space 14, 51, 52, 56, 105, 201
 – conformed 57
 – isolated 36
 – unified 4, 39, 49, 52, 57, 99, 106, 123, 162
 Multidimensional UML (m UML) 26
 MultiDimER model 62
 multiple hierarchies 15, 39, 42, 54, 61, 62, 68, 70, 77,
 78, 84, 122, 177
 – alternative 39, 43, 45, 84–86, 122, 138, 163, 180
 – – patterned 86
 – dependent 84–85
 – independent 84, 136
 – parallel 39, 43, 84–85, 104, 122, 163, 180

N

navigation scheme 4, 31–33, 38, 60, 157, 162, 173,
 176, 182, 184, 187
 – category-based 176–178
 – instance-based 176
 – scheme-based 178
 Nested Multidimensional Data Model (NMDM) 27
 node-link diagram *see* connection layout
 non-additivity 42, 46, 97

non-aggregability 42, 46, 96, 183
 non-dimension attribute 42–44, 99

O

O-O Multidimensional Model (OOMD) . . 26, 37, 48
 O3LAP 28
 Object Management Group (OMG) 155
 Object-Relational View (ORV) 28
 OLAP 1, 2, 10, 13, 35, 155, 156
 – Council 14
 – definition 13
 – operator
 – – CONDITIONAL HIGHLIGHTING 18
 – – DELETE LEVEL 18
 – – DICE 18, 19
 – – DRILL-ACROSS 18, 19, 28, 40, 103, 104, 109,
 178, 182, 184–186
 – – DRILL-AROUND 18
 – – DRILL-ASIDE 18, 19, 40
 – – DRILL-DOWN 18, 188
 – – DRILL-THROUGH 18
 – – DRILL-WITHIN 18, 19
 – – FILTER 18
 – – INSERT LEVEL 18, 183
 – – PIVOT 18, 19
 – – PROJECT 18
 – – PULL 18, 19, 61, 103, 104, 178, 182–183
 – – PUSH 18, 19, 61, 103, 104, 178, 182–183
 – – RANK 18, 19
 – – ROLL-UP 18, 188
 – – SELECT 18, 19
 – – SLICE 18, 19
 – – SLICE&DICE 18, 19
 – – SWITCH 18
 – operators 17, 96, 148, 186–187
 – – scheme-transforming 104, 134, 182
 – query 18, 29, 58, 173–175, 188
 – tools 29, 31, 60, 71, 125, 163, 168–170, 188
 On-line Analytical Processing *see* OLAP
 On-line Transaction Processing (OLTP) 14
 one-to-many relationship 115, 118
 one-to-one relationship 43–46, 69, 76, 77
 Open Information Model (OIM) 155
 Operational Data Store (ODS) 11, 13
 optional relationship 27, 42, 115
 Oracle BI Suite 13

P

Parallel Coordinates 170, 171, 176
 partial order 15, 53, 54, 61
 partial related roll-ups . 42, 45, 46, 55, 122, 136, 164
 Pentaho BI Platform 13, 165
 pivot table 30, 65, 125, 126, 129, 169, 172, 173, 176,
 196
 Polaris 169, 171
 primary key 22, 29, 45, 63, 132–134, 143, 144
 ProClarity 189, 194
 property attribute . . 15, 22, 27, 28, 36, 39, 41, 44–46,
 58, 115
 – “degree-of-belonging” 44, 72
 pseudo-star schema 64, 65

Q

query pattern 58, 77, 188
 query specification 4–6, 167, 182, 199
 – visual 32, 60, 168, 172–173, 188
 query specification cycle 172

R

Recursive Pattern 199
 related dimensions 56
 related fact schemes 39, 50, 56, 94, 99, 104, 132, 133
 related specializations 45, 46, 76
 Relational OLAP (ROLAP) . . . 20–23, 28, 132, 158,
 159, 173
 roll-up relationship . 36, 39, 42, 43, 45, 117, 146, 204
 – complete 46
 – degenerated 69
 – fact-dimensional 49, 69, 103, 106, 108, 134
 – full 39, 45, 46, 49, 55, 69, 103
 – fuzzy 46, 73
 – heterogeneous 43
 – interfactual 101, 108
 – non-strict . 45, 46, 71, 72, 132, 136, 150–152, 164
 – – weighted 44, 72, 152
 – optional 103, 122
 – partial 36, 39, 43, 45, 46, 50, 53, 55, 69, 102, 103,
 132, 164
 – strict 46, 72, 151

S

satellite fact 100, 103, 108, 119, 162, 202
 scheme accuracy 115

Secure Relational Data Warehouse (SECRDW) meta-model 165
 self-association fact 100, 101, 119
 self-specialization 143
 semi-additivity 97
 shadow dimension 69
 snowflake schema 22, 28, 132
 SolarPlot 194, 195
 space-filling (enclosure) layout 194, 197
 Spatial OLAP (SOLAP) 172
 specialization 66, 75–79, 117, 118
 – complete 79, 80
 – disjoint 46, 76, 79, 82
 – incomplete 80, 81, 141, 202, 207
 – overlapping 46, 80, 82, 136, 141, 202, 207
 specialization hierarchy 45, 79–82, 122, 180
 splitting level 76, 77, 140
 SQL (Structured Query Language) .. 20, 28, 32, 126, 140, 145, 154, 172–174
 SQL(\mathcal{H}) 37, 61
 staging area 12
 star cluster schema 23
 star join 22
 star schema 22, 28, 60
 starER 25, 112
 stratification 55
 structured cube model 61
 subdimension 23, 70–71, 76
 summarizability .. 2, 4, 36, 37, 60–62, 65, 71, 72, 82, 136, 148
 summary table 22, 23, 134
 SuperX 62–63
 surgical workflow 90, 92–94, 102, 119, 124
 Surgical Workflow Analysis (SWA) 90–91, 123
 – applications 91
 – query types 91

T

Tableau Software 32, 34, 169, 176
 Third Multidimensional Normal Form (3MNF) .. 27
 Third Normal Form (3NF) 135–136
 total order 15, 55
 transitivity 53–54
 TreeMap 194, 195, 200
 trigger 135, 140

U

UML (Unified Modeling Language) .. 25–26, 28, 45, 62, 112, 134, 155, 156, 162
 – class diagram 26, 112, 114
 – profile mechanism 26
 – self-extensibility 37
 unified multidimensional metamodel 26
 Upper Level (UL) model 47, 49–52

V

view 134
 VisDB 168
 visual analysis 174, 188
 visual layout 175, 176
 Visual OLAP 31, 32, 168, 172
 – definition 31
 visual scalability 169, 188
 visualization 30–33, 168–170, 173, 174
 visualization interface 33–34
 visualization techniques 4, 31–33, 168–172, 174, 176
 – 3D 171
 – hierarchical 4, 38, 168, 188, 194
 – multidimensional 5, 169, 171, 176
 – multiscale 171, 188
 – space-filling 171, 188, 194, 196
 – spatio-temporal 199–200
 VizQL (Visual Query Language) 34

X

x-DFM 4, 41, 43, 99, 112
 – dimension sharing modes 105–106
 – edge constructs 45–46
 – IL constructs 57
 – node constructs 43–44
 – UL constructs 52
 XML (Extensible Markup Language) 38, 155, 160–162, 165

XML Metadata Interchange (XMI) 155, 160

Y

YAM² *see* Yet Another Multidimensional Model
 Yet Another Multidimensional Model 26, 48, 62

Z

zoom graph 171