Ersch. in: Complex data warehousing and knowledge discovery for advanced retrieval development : innovative methods and applications / ed. by Tho Manh Nguyen. - Hershey Pa. [u.a.] : Information Science Reference, 2010. - S. 129-173. - ISBN 978-1-605-66748-5

Conceptual Data Warehouse Design Methodology for Business Process Intelligence

Svetlana Mansmann
University of Konstanz, Konstanz, Germany

Thomas Neumuth
Innovation Center Computer Assisted Surgery (ICCAS), Leipzig, Germany

Oliver Burgert
Innovation Center Computer Assisted Surgery (ICCAS), Leipzig, Germany

Matthias Röger
University of Konstanz, Konstanz, Germany

Marc H. Scholl
University of Konstanz, Konstanz, Germany

ABSTRACT

The emerging area of business process intelligence aims at enhancing the analysis power of business process management systems by employing performance-oriented technologies of data warehousing and mining. However, the differences in the assumptions and objectives of the underlying models, namely the business process model and the multidimensional data model, aggravate straightforward and meaningful convergence of the two concepts. The authors present an approach to designing a data warehousing for enabling the multidimensional analysis of business processes and their execution. The aims of such analysis are manifold, from quantitative and qualitative assessment to process discovery, pattern recognition and mining. The authors demonstrate that business processes and workflows represent a non-conventional application scenario for the data warehousing approach and that multiple challenges arise at various design stages. They describe deficiencies of the conventional OLAP technology with respect to business process modeling and formulate the requirements for an adequate multidimensional presentation of process descriptions. Modeling extensions proposed at the conceptual level are verified by implementing them in a relational OLAP system, accessible via state-of-the-art visual frontend tools. The authors demonstrate the benefits of the proposed modeling framework by presenting relevant *analysis tasks from the domain of medical engineering and showing the type of the decision support provided by our solution.*

INTRODUCTION

Modern enterprises increasingly integrate and automate their business processes with the objective of improving their efficiency and quality, reducing costs and human errors. Business Process Management Systems (BPMS) are employed to optimize process design and execution. These systems track business processes by logging large volumes of data related to their execution and provide basic functionality for routine analysis and reporting. However, conventional BPMS focus on the design support and simulation functionality for detecting performance bottlenecks, with rather limited, if any, analysis capabilities to quantify performance against specific business metrics. Deficiencies of the underlying business process modeling approaches in terms of supporting comprehensive analysis and exploration of process data have been recognized by researchers and practitioners (Dayal, et al., 2001; Grigori, et al., 2004).

The ability to analyze process execution has become indispensable for eliminating the gaps in decision making. Last decade witnessed immense technological advancements in application integration, business rules and workflows, Business Intelligence (BI), and BPMS. Forward-thinking organizations are beginning to realize that process intelligence goes beyond simple automation of business processes and that the convergence of BI and business process management technologies would create value beyond the sum of their parts (Smith, 2002). The fundamental technology of BI is referred to as OLAP (*On-line Analytical Processing*), a term coined by Codd, et al. (1993). Data warehousing and OLAP are aimed at providing key people in the enterprise with access to whatever level of information they need for decision making.

BUSINESS PROCESS INTELLIGENCE

"*Business Process Intelligence* (BPI) refers to the application of business intelligence techniques (including for example OLAP analysis and data mining) in business process management, with the goal of providing a better understanding of a company's processes and of devising ways to improve them." (Castellanos & Casati, 2005). Recent advances in the above techniques as well as in business process and business performance management have come together to enable a near real-time monitoring and measurement of business processes as to identify, interpret, and respond to critical business events.

According to Hall (2004), BPI can help companies improve their process management initiatives by:

- providing a consistent, process-based view of the company,
- facilitating real-time business process monitoring,
- aligning execution with strategy,
- managing enterprise performance.

The BPI approach overcomes the deficiencies of standard BPMS by storing process execution data in a data warehouse in a cleansed, transformed, and aggregated form (Dayal, et al., 2001). Such data can be analyzed using OLAP and data mining tools to support various knowledge extraction tasks that can be subdivided into the following subareas (Castellanos & Casati, 2005):

- *Process discovery* is done by analyzing enterprise operations in order to derive the process model that can be used for automating process execution or increasing its efficiency.

- *Process mining and analysis* seeks to identify interesting correlations helpful for forecasting, planning, or explaining certain phenomena.
- *Prediction* is important for anticipating or preventing occurrence of certain situations.
- *Exception handling* assists the analyst in addressing specific problems, for instance, by retrieving the data on how similar problems were handled in the past.
- *Static optimization* is concerned with optimizing the process configuration against previously identified optimization areas.
- *Dynamic optimization* is an intelligent component for supervising process instances at runtime in order to influence their execution as to maximize certain business objectives.

The employment of BI within the BPI framework has also caused companies to rethink the ways they use data warehouses by blurring the traditional separation of operational systems from BI applications (Hall, 2004). Traditionally, data warehouses store consolidated historical data and, thus, provide a retrospective analysis. In BPI scenarios, data warehouses are fed with current transactional data that has to be available for near real-time analysis. This requirement of supporting day-to-day decision-making has triggered the emergence of a new branch called *Operational BI*, which links BI with business processes and enables process-oriented perspective of the analysis.

"*Operational BI* combines real-time operational transaction data with historical information to let decision-makers move beyond the "point-in-time" analysis associated with traditional BI and data warehousing applications" (Hall, 2004).

Within our research, the terms *Business Process Intelligence* and *Operational Business Intelligence* are treated interchangeably.

CONTRIBUTION AND OUTLINE

The area of BPI is still immature and controversial, with many open issues and very few examples of existing solutions. One of the major BPI challenges is finding a meaningful solution for converging business process and workflow modeling techniques with the multidimensional data model that lies at the heart of the OLAP technology. The task of unifying the flow-oriented process specification and the snapshot-based multidimensional design for quantitative analysis is by far not trivial due to differing and even conflicting prerequisites and objectives of the underlying approaches.

Concepts and proposals presented in this work have been inspired by practical challenges encountered in the ongoing project on designing and implementing a BPI platform for a specific domain of Surgical Workflow Analysis (SWA). The project is hosted by the Innovation Center Computer Assisted Surgery (ICCAS)[1] and involves collaborators from multiple scientific disciplines, such as medicine, medical engineering, databases and data warehousing, web technologies, scientific visualization, etc. Surgical Workflows will be used as a real-world usage scenario for demonstrating the applicability of the presented solution.

The contribution of this work is to design a methodological framework for enabling business process analysis. The fundamental challenge of invoking the OLAP approach in the BPI context is a conceptual one, namely, gaining an adequate multidimensional perspective of process execution data. We demonstrate that the classical data warehouse design steps are not feasible in this scenario due to general unavailability of pre-defined measures of interest. As a solution, we propose a cardinality-based approach of transforming existing process models and process execution schemes into a set of facts and dimensions in a unified multi-dimensional space. The multidimensional model itself had to be extended to handle complex patterns encountered in the data. These extensions are reflected in terms of formal concepts as well as a graphical notation *X*-DFM, which extends the popular DF Model of Golfarelli, et al. (1998). We expect the proposed extended model to be applicable to a variety of data warehouse scenarios dealing with complex data. As a proof of concept, we demonstrate its usage of our model for solving typical SWA tasks.

The remainder of the chapter is structured as follows: Section 2 provides an overview of the related work in the field of BPI in general and Surgical Workflow Analysis in particular. The case study and its analysis requirements is presented in Section 3. Section 4 contains the background information on the relevant conceptual data models, followed by Section 5 featuring the challenges of business process data warehouse design. In Sections 6 and 7 we present an extended conceptual model in terms of its fundamental elements and advanced concepts, respectively. Section 8 describes the overall approach to obtaining a multi-dimensional business process model from existing process descriptions, based on analyzing and refining the cardinalities of the relevant relationships between process components. Section 9 contains some considerations regarding the implementation and demonstrates the use of the presented framework for solving exemplary tasks from the field of SWA. Concluding remarks are given in Section 10.

RELATED WORK

Due to multidisciplinarity of our research, the related work falls into several categories, such as (a) enhancing business process analysis by employing the data warehousing approach, (b) extending OLAP to support complex scenarios, and (c) medical informatics research related to our application field of SWA.

Grigori, et al. (2004) present a comprehensive BPI tool suite for managing business process quality that was developed at Hewlett-Packard and implemented on top of HP Process Manager BPMS. The suite includes three main components: 1) the PDW loader for transferring the process log data into a Process Data Warehouse (PDW), 2) the Process Mining Engine for deriving sophisticated models from the data, and 3) the Cockpit, which is a graphical reporting tool of the end-user. The data warehousing approach was employed for structuring the relevant process data according to the star schema, with process, service, and node state changes as facts and the related definitions as well as temporal and behavioral characteristics as dimensions. This approach enables analysis of process execution and system state evolution in the environments where processes have a uniform and well-defined scheme.

Hao, et al. (2006) proposed an approach to visual analysis of business process performance metrics (impact factors) using *VisImpact*, a visualization interface especially suitable for aggregating over large amounts of process-related data and based on analyzing process schemes and instances to identify business metrics of interest. The selected impact factors and the corresponding process instances are presented using a symmetric circular graph to display the relationships and the details of the process flows.

Medical applications are frequently encountered in the data warehousing literature in the role of motivating case studies. Pedersen, et al. (2001) proposed an extended multidimensional data model for meeting the needs of non-standard application domains at the example of accumulated patient diagnosis data. Golfarelli, et al. (1998) demonstrate the methodology of obtaining multi-dimensional schemes from existing E/R schemes using hospital admission as a usage scenario. Song, et al. (2001) use patient diagnosing and billing case study to demonstrate various strategies of handling many-to-many relationships between facts and dimensions. Mansmann, et al. (2007a) describe how Surgical Process Modeling, used as a non-conventional data warehousing application scenario, results in the necessity to extend the conceptual foundations of the multidimensional data model. Implications of conceptual extensions for implementing a data warehouse and frontend tools for interactive analysis are given in (Mansmann, et al., 2007b).

Another category of related works refers to the modeling of Surgical Workflows. An approach to facilitating the complex task of surgery preparation by employing the workflow technology to automate and optimize the surgical process was presented by Qi, et al. (2006). Münchenberg, et al. (2000) designed instruction graphs to drive a surgical assist system for application in Frontal Orbital Advancements. Jannin, et al. (2003) used a ontologically designed scheme to model activities in the context of image-guided surgery. Ahmadi, et al. (2006) proposed an approach to automatic surgical workflow recovery without explicit models of surgery types. A more recent work of Padoy, et al. (2007) presents a model-based recovery approach based on automatics segmentation of surgeries into phases using hidden Markov models.

A pioneering interdisciplinary research on designing scientific methods for Surgical Workflows is carried out at ICCAS. Major directions of their projects are surgical workflow formalization (Neumuth, et al., 2006), semantics (Burgert, et al., 2006), analysis (Neumuth, et al., 2007), standardization (Burgert, et al., 2007), and visualization (Neumuth, Schumann, et al., 2006).

MOTIVATING CASE STUDY

Medical applications are frequent suppliers of motivating usage scenarios in workflow management research. Patient treatments, diagnostic investigations, hospitalization, surgical interventions, and the overall hospital operation are examples of complex processes where the workflow technology promises significant performance gains. Our case study is concerned with an emerging interdisciplinary field of SWA.

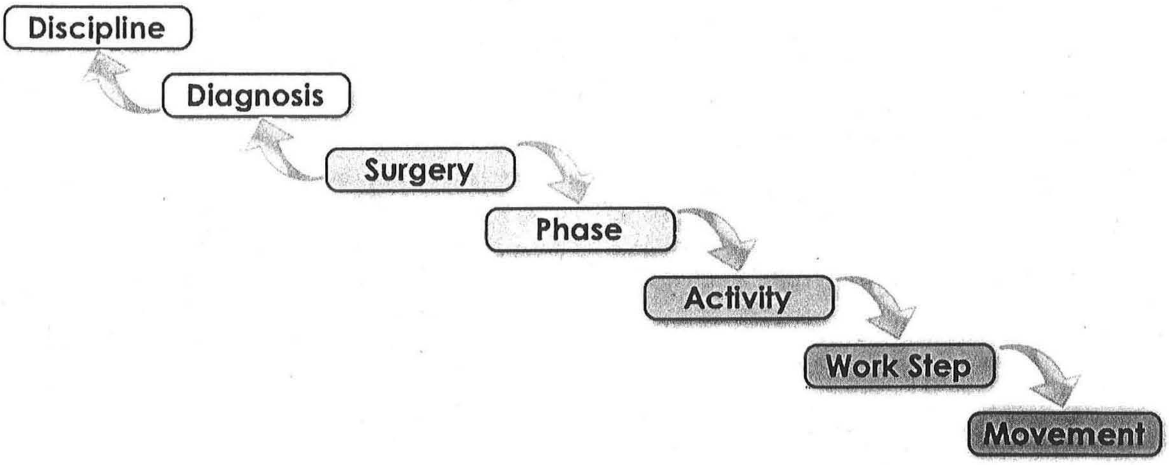
Surgical Workflows foster intelligent acquisition of process descriptions from surgical interventions for the purpose of their clinical and technical analysis, as defined by Neumuth, Strauß, et al. (2006). This type of analysis is crucial for developing surgical assist systems for the operating room of the future. Besides, it provides a framework for evaluating new devices or surgical strategy evolution. The medical informatics term *Surgical Workflows* describes the methodological concept of the data acquisition and consolidation procedure. Process data is obtained manually or semi-automatically by monitoring and recording the course of a surgical intervention. The manual part is carried out either in the real-time mode, i.e., by observing the surgical intervention live in the operating room, or retrospectively, e.g., from a video recording.

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. 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 ergonomic conditions, verification of medical hypotheses, gaining input for designing surgical assist systems and workflow automation. Obviously, such high diversity of potential applications

Figure 1. Vertical (de-)composition of a surgical process



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 query types, the multidimensional database technology seems a promising solution as it allows the analyst to view data from different perspectives, define various business metrics, and aggregate the data to the desired granularity.

STRUCTURING SURGICAL WORKFLOWS

Surgical Workflows provide an abstraction of surgical interventions 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 sub-processes, stages, work steps, etc. Figure 1 shows a possible decomposition hierarchy of a surgery.

From the logical point of view, surgical processes consist of phases, which, in their turn, consist of activities, i.e., work steps performing a certain action. Both phases and activities may overlap. Technically, an action may be executed by multiple participants using multiple instruments. To account for this observation, we refine the granularity to a "movement", which refers to a part of an action performed by a body part of a participant on a structure of a patient using a surgical instrument. In the upward direction, surgical instances can be grouped into classes by the diagnosis or therapy, which, in their turn, are associated with particular surgical disciplines. The above decomposition is called *logical*, or *task-driven* as it relies on the reasoning of a human expert for recognizing the constituent elements of a process.

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 state evolution of involved systems and 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, surgeon's eyes can be considered a system, their gaze direction can be then modeled as states, while surgeon's directives to other participants may be captured as events.

Both data acquisition practices can be used as complementary ones to benefit from combining a human perspective with a systemic one. We introduce a superordinate concept *component*, synonymous to the term *flow object* defined in BPMN (2006), to enable uniform treatment of logical (i.e., activities) and technical (i.e., 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.

With respect to the vertical decomposition depicted in Figure 1, we propose to distinguish between 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, etc. This data is normally supplied by other clinical information systems. Workflow-level data is useful for high-level analysis, such as hospital utilization, patient history, etc.
- *Intra-workflow level* refers to the properties of process components (e.g., events, activities), such as instrument and device usage or treated structures. Detailed data is acquired from running surgical interventions and used for analyzing workflow execution within as well as across multiple instances.

Figure 2 shows a simplified approximation of Surgical Workflows structure, expressed in the E/R (Entity-Relationship) modeling notation. This scheme will be refined in the upcoming sections. To identify the major design challenges, we proceed by inspecting the fundamentals of the involved modeling techniques.

CONTROL FLOWS VS. MULTIDIMENSIONAL CUBES

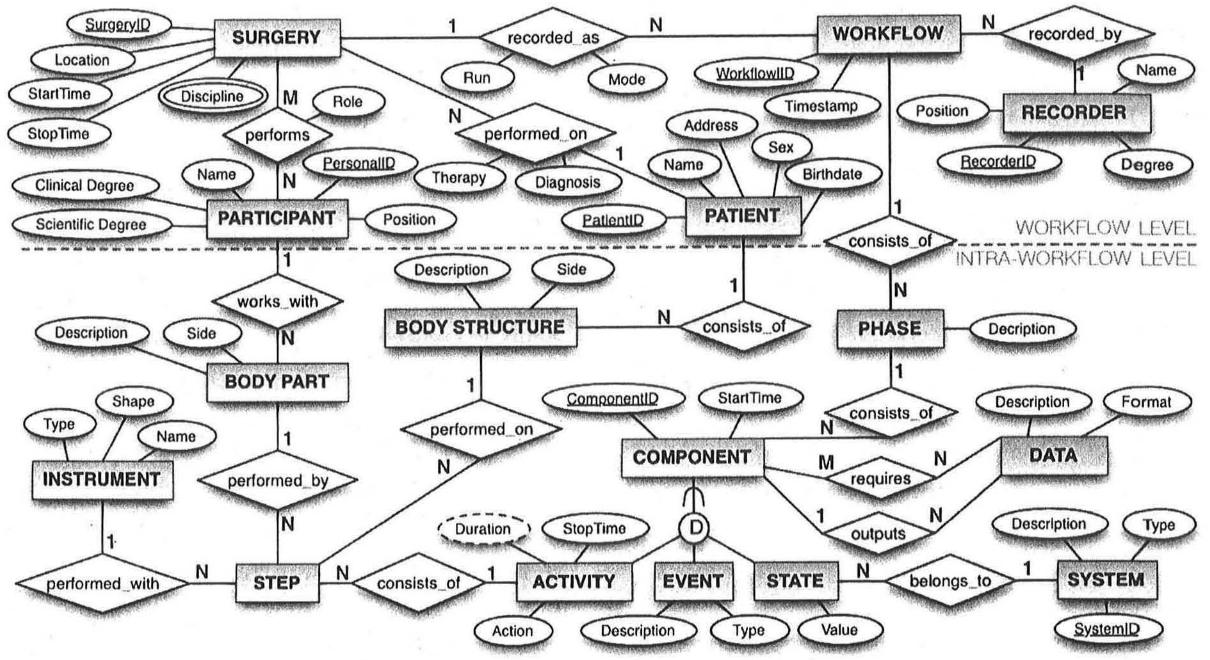
As mentioned in the introductory section, BPI aims at converging the techniques of business process modeling and business intelligence. More precisely, business process models serve as the input whereas the multidimensional data model builds the foundation of a BPI framework. In this section, we overview the main concepts of both models as a preparation step for finding ways of their meaningful convergence.

BUSINESS PROCESS MODELING

Business process models are employed to describe business activities in the real world. Business processes are typically described in terms of their objects, activities, and resources. WfMC (1999) defines *business process* as "a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships" and proposes to distinguish between manual and workflow activities. *Activities* are the work units of a process that have an objective and change the state of the objects. *Resources* are consumed to perform activities. Relationships between the entities may be specified using *control flow* (consecutive, parallel, or alternative execution) and/or hierarchical decomposition.

There is an important distinction between the conceptual and the actual manifestation of a

Figure 2. Recording scheme of a surgical process model as an E/R diagram

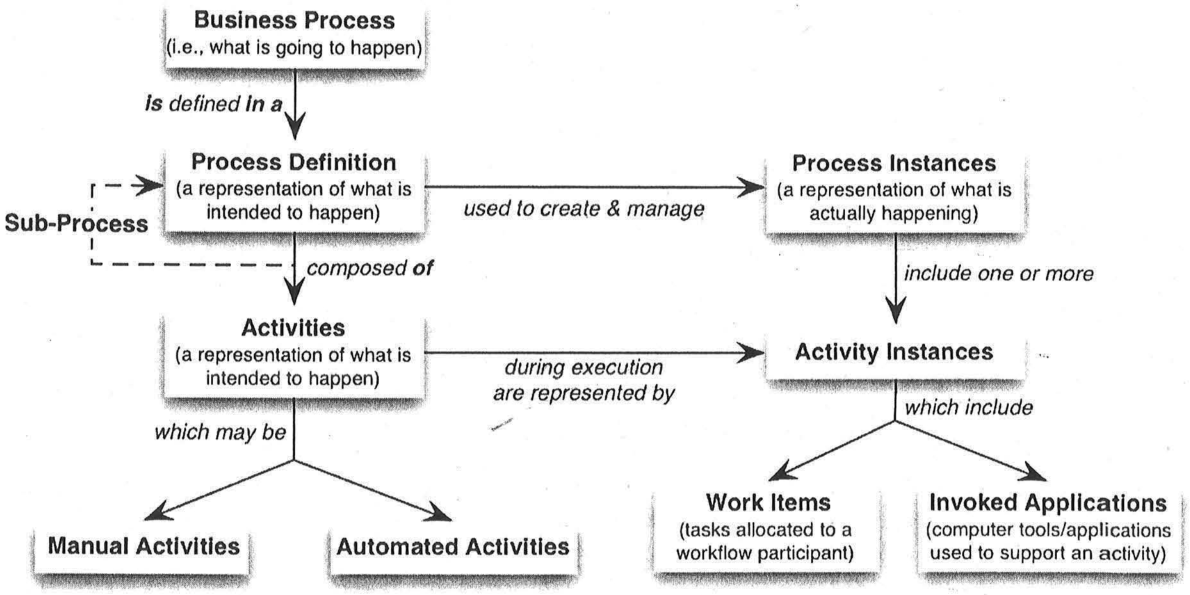


process: the term "process" refers to a conceptual, or abstract, way of organizing work and resources whereas process executions, or "instances", involving real resources and actors are the actual manifestation of a business process (Reijers, 2003). An example from the medical domain could be a surgery of type discectomy. Abstract process description of discectomy is "removal of all or part of an intervertebral disc (the soft tissue that acts as a shock absorber between the vertebral bodies)" (SRS, n.d.). This description may further define a typical cause of a surgery, major work steps, and the types of instruments and devices used at each step. Instances of discectomy as a surgical process are actual surgeries carried out by particular surgeons.

Another distinction has to be made between the concepts *process* and *workflow*. While these two terms are used interchangeably by some authors (Aalst & Hee, 2002), diverse workflow definitions can be found in the literature. One popular interpretation is that business processes output products while workflows deliver services (Reijers, 2003). Another use of the term "workflow" is to denote the control flow, i.e., dependencies among tasks during the execution of a business process (Sharp & McDermott, 2001). In this work, we adopt the differentiation in the levels of abstraction proposed by Muth, et al. (1998): while business processes are mostly modeled in a high-level and informal way, workflow specifications serve as a basis for the largely automated execution and are derived by refining the business process specification. Figure 3, adopted from (WfMC, 1999) with some adjustments, summarizes the relationships between the basic terms related to business processes.

Coexistence of different workflow specification methods is common in practice. We restrain ourselves to naming a few techniques and refer the interested reader to the book of Matoušek (2003) for a detailed overview. *Net-based*, or *graph-based*, methods enjoy great popularity due to their ability to visualize processes in a way understandable even for non-expert users. Especially the *activity and state charts* are frequently used to specify a process as an oriented

Figure 3. Relationships in the basic business process terminology



Business Process
(i.e., what is going to happen)
is defined in a
Process Definition
(a representation of what is intended to happen)
Sub-Process
composed of
used to create & manage
Process Instances
(a representation of what is actually happening)
include one or more
Activities
(a representation of what is intended to happen)
during execution are represented by
Activity Instances
which may be
which include
Manual Activities
Automated Activities
Work Items
(tasks allocated to a workflow participant)
Invoked Applications
(computer tools/applications used to support an activity)

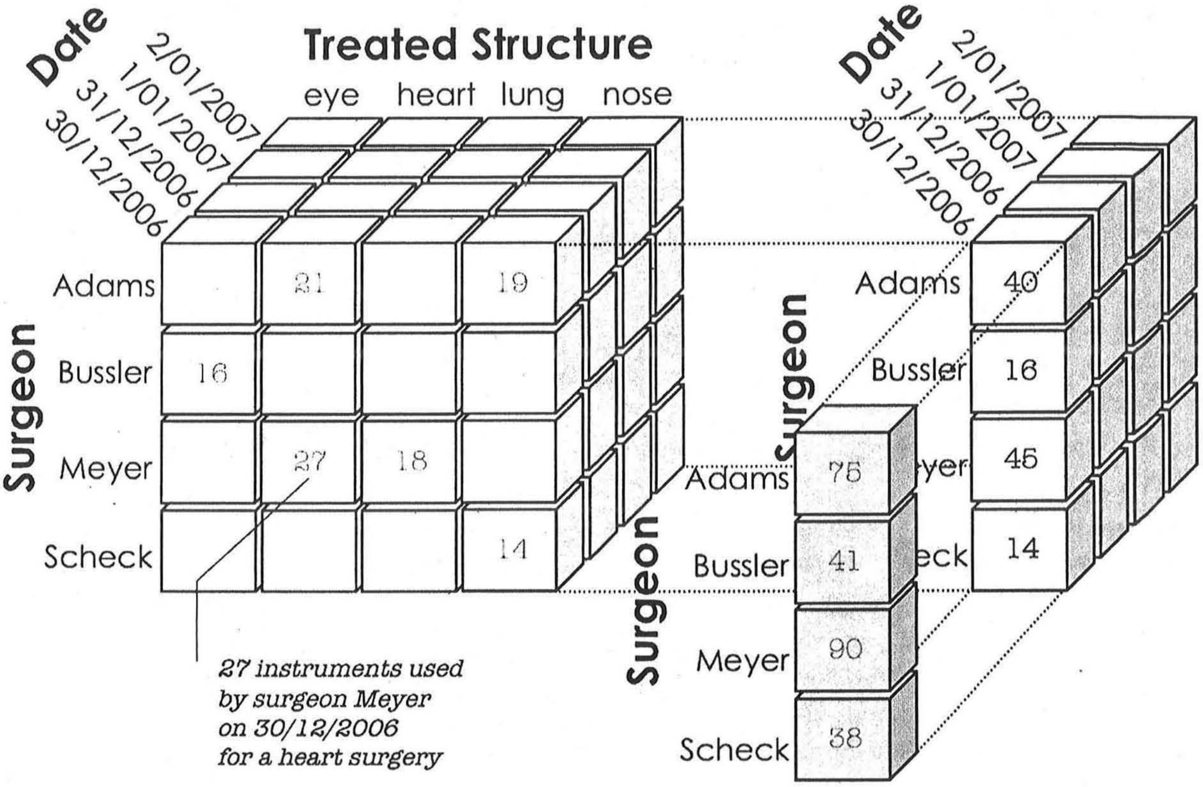
graph with nodes representing the activities and arcs defining the ordering in which these are performed. *Logic-based* methods use temporal logic to capture the dynamics of the system. Finally, *Event-Condition-Action* rules are used for specifying the control flow between activities in the conditional form.

MULTIDIMENSIONAL DATA MODEL AND OLAP

OLAP technology draws its analytical power from the underlying *multidimensional data model*. The data is modeled as cubes of uniformly structured *facts*, consisting of analytical values, referred to as *measures*, uniquely determined by descriptive values drawn from a set of *dimensions*. Each dimension forms an axis of a cube, with dimension members as coordinates of the cube cells storing the respective measure values. Figure 4 shows a simplified example of a 3-dimensional data cube, storing instrument usage statistics (measure number of instruments) determined by dimensions Surgeon, Treated Structure, and Date. Besides the original cube storing the data at the finest granularity, Figure 4 also displays the results of two "roll-up" operations totaling the measure over all treated structures and, subsequently, over all dates. In real-world applications, data cubes may have arbitrarily many dimensions, and are therefore denoted *hypercubes*.

Member values within a dimension are further organized into *classification hierarchies* to enable additional aggregation levels. For example, dates can be aggregated into months, quarters, years, and so on. Dimension hierarchies are strictly structured, i.e., values at each hierarchy level must be of the same *category*. Multiple hierarchies may be defined within a dimension and can be mutually exclusive (e.g., dates can be aggregated by month or by week, but not both), denoted *alternative*, or non-exclusive, or *parallel* (e.g., surgeons can be grouped by qualification and, subsequently, by the level of expertise, or vice versa). Within a dimension, the attributes that form the hierarchy are called *dimension levels*, or *categories*. Other descriptive attributes belonging to a particular

Figure 4. A sample 3-dimensional cube (fragment) storing surgical instrument usage statistics (left) and its aggregated views (right)



category are *property attributes*. For instance, Hospital and City are categories of the dimension Location, whereas hospital name and city code are properties of the respective categories. Categories along with parent-child relationships between them represent the *intension*, or *scheme*, of a dimension whereas the hierarchy of its members, i.e., the actual data tree, forms its *extension*.

Desired subsets and views for analysis can be retrieved from the "raw" data by applying OLAP operations, such as *slice-and-dice* to reduce the cube, *drill-down* and *roll-up* to perform aggregation and disaggregation, respectively, along a hierarchical dimension, *drill-across* to combine multiple cubes, *ranking* to find the outlier values, and *rotating* to see the data grouped by other dimensions (Pedersen& Jensen, 2001).

BUSINESS PROCESS DATA WAREHOUSE DESIGN: CHALLENGES

Transformation of semantically rich business process models into multidimensional data structures can be seen as a reduction of the complete set of extensible process elements, such as various types of flow objects and relationships between them, to a rigid format, which forces the former to be decomposed into a set of uniformly structured facts with associated dimensions.

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 this framework's validity for data warehouse design (Hüsemann, et al., 2000). In addition to the above three phases, Golfarelli & Rizzi, (1998) identify two phases preceding the conceptual design, namely, *i) analysis of the information system* for obtaining the (conceptual or logical) scheme of the pre-existing information system, and *ii) requirement specification* for defining the type of analysis and indicating the preliminary workload. Back to the Surgical Workflows scenario, the E/R scheme in Figure 2 may be taken as a model of the pre-existing system, whereas the expected types of queries and applications given in Section 3 correspond to the output of the requirement specification phase.

STAGES OF THE CONCEPTUAL MODELING

The convergence of the business process model and the multidimensional data model takes place primarily at the conceptual level. Therefore, the conceptual design phase is the central issue of this work. 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, such as E/R diagrams, UML and their variants, understandable by both designers and target users.

According to Hüsemann, et al., (2000), conceptual data warehouse design process evolves in the following consecutive phases:

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

The versatility of feasible application areas and analysis tasks of BPI imposes multiple challenges on the conventional data warehouse design methodology. Back to the kinds of queries in the SWA context, the same data field may serve as a measure, i.e., input of an aggregate function, in one query and 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 2. In order to decide whether those entity types should be mapped to facts or to 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 of symmetric treatment of measure and dimension roles. Similar examples can be specified for virtually any other entity of the case study. In order to support all kinds of expected queries, the detailed data, i.e., without pre-aggregation to any of the expected measures of interest, should 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 BPI. Kimball proposes a slightly different approach to structuring the conceptual design process, which appears more applicable in the context of BPI. According to Kimball (1996), 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 proposed by Hüsemann, et al. (2000), to a final step, as in the approach of Kimball (1996), that the support of operational BI scenarios can be achieved.

Quantitative queries represent just a fraction of SWA. Some BPI tasks go beyond mere aggregation and may address more complex issues, such as pattern recognition, relevance assessment, and process discovery. These tasks require the original process data in the warehouse to be stored without aggregation.

FUNDAMENTAL CONSTRAINTS OF THE MULTIDIMENSIONAL DATA MODEL

Further modeling challenges come from the inherent constraints of the multidimensional model itself, such as prohibition of many-to-many relationships and NULL values, homogeneity of the fact's characteristics and their grain, and a requirement of summarizability for all dimension hierarchies. Many of these constraints are fundamental and, as such, may not be violated or trivially overcome. We proceed by enumerating some of such fundamental issues that aggravate straightforward applicability of OLAP to business process data:

- *"Rolls-up-to" as the only relationship type*. This relationship expresses inclusion between facts and dimensions as well as between hierarchy levels. It is impossible to explicitly model any other relationship types.
- *Any many-to-many relationship must be modeled as a fact*. This "law" of Kimball (1996) prohibits non-strict hierarchies and many-to-many relationships between facts and dimensions.
- *Fact homogeneity* implies that all fact entries fully adhere to the fact scheme, i.e., have the same dimensional characteristics and uniform granularity in each dimension.
- *Homogeneous aggregation* requires that all entries within the same fact type roll up along the same set of 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 and dimension schemes and 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 separately from other schemes. Whenever multiple fact or dimension schemes have identical or semantically related attributes, those are maintained redundantly. Besides, scheme isolation prevents from supporting advanced OLAP operators, such as drill-across, at the conceptual level.
- *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 (Lenz & Shoshani, 1997).
- *Duality of measure and dimension roles.* Measures reflect the focus of the analysis and, therefore, they should be known at design time and 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*. Properties may not be used as aggregation levels, even

though the relationship between a category attribute and its property is equivalent to "roll-up".

In the next section we present our approach to mapping business process schemes to multidimensional schemes and show how the above limitations of the multidimensional data model can be handled.

CONCEPTUAL DATA WAREHOUSE DESIGN: TERMINOLOGY AND FORMALIZATION

In the previous section we showed that the classical data warehouse design approach, based on identifying the measures of interest and their dimensional context, is not adequate for modeling business process schemes. Instead, we propose to derive a multidimensional scheme from a pre-existing conceptual model of the process, available as E/R or UML class diagrams. Entity-Relationship 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. Therefore, it is sufficient to provide a mapping for either of these two models. We use E/R model as the input graphical notation and consider the model depicted in Figure 2 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.

Two major components of semantic models are formalization and graphical notation. Existing multidimensional data models tend to focus either on the formalism or on the graphical toolkit, but not both. Formal models either adopt some existing notation (e.g., ER, UML or their variants) or do not employ any. For the purpose of completeness, we provide both the formalism and the graphical model that is fully aligned with the proposed formal concepts, i.e., that correctly captures its semantics.

Our conceptual model relies on the popular Dimensional Fact Model (DFM) proposed by Golfarelli, et al. (1998). DFM is based on a pragmatic scientific approach, in which the graphical framework emanates from the formal conceptual framework. The authors also provide a methodology for deriving multidimensional schemes from E/R diagrams. In the abundance of notations proposed in the literature, DFM stands out for its simplicity, elegance, and expressiveness for representing the concepts introduced in our work. However, we use an extended variant of DFM, called *X*-DFM (e<u>x</u>tended <u>D</u>imensional <u>F</u>act <u>M</u>odel), which provides an adequate mapping for a broader set of semantic elements. The formalization is adopted from our previous works (Mansmann & Scholl, 2007; Mansmann, et al., 2007a) with some modifications and builds upon the semantic models of Pedersen, et al. (2001) and Golfarelli, et al. (1998).

A UNIFIED MULTIDIMENSIONAL SPACE

One fundamental definitional issue in the conceptual model is whether global semantics, i.e., relationships across fact schemes, should be captured. A conventional approach would be to design each *n*-dimensional data cube in its own isolated *n*-dimensional space. The output of such model is a set of unrelated fact schemes. However, advanced models, such as DFM, support inter-factual semantics by allowing facts to share dimensions. The major advantage of the latter approach is given by the explicit support for a drill-across operation, which allows to compare measures of related data cubes or even to derive new measures.

A set of dimensions is merged into one shared dimension, if they are defined on a related semantic domain For example, dimensions StartTime and StopTime, both of type *date*, could be modeled as a common dimension time, containing the union of values from both dimensions. In addition to such full dimension sharing, our model recognizes further types of sharing by considering semantic compatibility at category level. The resulting conceptual schema is called *inter-stellar*, or *galaxy*. Inter-factual relationships 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 multi-dimensional space*, in which categories with semantically related value domains are represented in a non-redundant fashion.

FACTS AND DIMENSIONS

The output of the conceptual data warehouse design is a *multidimensional scheme*, i.e., a set of *fact schemes* composed of facts, measures, dimensions, and hierarchies. Golfarelli, et al. (1998) define a *fact scheme* to be a structured quasi-tree, which is a directed, acyclic, weakly connected graph, in which multiple directed paths may converge on the same vertex. Path convergence is the result of non-redundant dimensional modeling enforced by the constraint of the unified multidimensional space.

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

Definition 2. A ***dimensionD*****is defined by its aggregation scheme (intension)*D*and the associated data set (extension)** E, **so that***Type*(E) = D.

The samle data cube from Figure 4 can now be formally defined as a fact scheme INSTRUMENTS-CUBE with a set of measures $M^{\text{INSTRUMENTS-CUBE}} = \{\text{num\_instruments}\}$, characterized by a set of dimensions $D^{\text{INSTRUMENTS-CUBE}} = \{\text{Surgeon, Treated Structure, Date}\}$.

A dimension scheme is a connected, directed graph, in which each vertex corresponds to an aggregation level and each edge represents a full or partial roll-up relationship between a pair of levels, or formally:

Definition 3. A ***dimension scheme*** **is a quadruple** $D = (C^D, \sqsubseteq_D, T_D, \perp_D)$, where $C^D = \{C_k, k = 1, \ldots, p\}$ **is set of category types, or dimension levels, in D,** $\sqsubseteq_D$**is a partial order in C, and** T_D**and** $\perp_D$**are distinguished as the top and the bottom element of the ordering, respectively.**

$\perp_D$ corresponds to the finest grain of D, i.e., the one at which D is connected to the fact scheme. T_D corresponds to an abstract root node of the dimension's hierarchy that has a single value referred to as ALL.

Relation $\sqsubseteq_D$ captures the containment relationships between category types. This containment may be *full*, denoted $\sqsubseteq_D^{(\text{full})}$, or *partial*, denoted $\sqsubseteq_D^{(\text{part})}$. Therefore, relation $\sqsubseteq_D$ indicates the union of the two orders. Admission of partial containment 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 C^D. Partial and full direct containment predicates are denoted $\sqsubseteq^{(\text{part})}$ and $\sqsubseteq^{(\text{full})}$, respectively. Thereby, predicates $\sqsubseteq$ and $\sqsubseteq^*$ without fullness/partiality indication imply that the containment is either full or partial, or formally: $C_i \sqsubseteq C_j \Rightarrow (C_i \sqsubseteq^{(\text{full})} C_j \vee C_j \sqsubseteq^{(\text{part})} C_j)$. Partial containment between two categories $C_i \sqsubseteq^{(\text{part})} C_j$ occurs when members of C_i are not required to have parent members in C_j.

A pair of partial containment relationships of the same category C_i (i.e., $C_i \sqsubseteq^{(\text{part})} C_j \wedge C_i \sqsubseteq^{(\text{part})} C_k$) are *exclusive*, if each member of C_i rolls up either to C_j or C_k, but never to both. A set of exclusive partial roll-up relationships is denoted $C_i \sqsubseteq^{(part)} (C_j \mid C_k)$.

C_j is said to be a category type in C, denoted $C_j \in C$. Dimension scheme defines a skeleton of the associated data tree, for which the following conditions hold:

1. $\forall \mathbf{C}_j \in \mathbf{C}^{\mathbf{D}} \setminus \{\mathbf{T}_{\mathbf{D}}\}$: $\mathbf{C}_j \sqsubseteq^{*(full)} \perp_D$ (a non-top category type is fully contained in the top category type).
2. $\forall \mathbf{C}_j \in \mathbf{C}^{\mathbf{D}} \setminus \{\perp_D\}$: $\perp_D \sqsubseteq^{*} C_j$ (bottom category rolls up, fully or partially, to all upper category types).
3. $\exists \mathbf{C}_j \in \mathbf{C}^{\mathbf{D}}$: $\mathbf{C}_j \sqsubseteq \perp_D$ (the bottom category type is childless).

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

Definition 4. A *dimension category type* is a pair $C = (A^C, A)$ where $A^C$is the distinguished dimension level attribute and $A = \{A_r, r = 1, \ldots, x\}$ is a set of property attributes associated with A^C.

Definition 5. An *aggregation path* in D is given by a pair of category types C_i, $C_j$such that $(C_i, C_j) \in C^D \wedge C_i \sqsubseteq^{*} C_j$.

Having defined the scheme elements of the model, we proceed to dimension instances and their properties.

Definition 6. An *instance*, or *extension*, E associated with dimension scheme D is a pair $(C^E, \subseteq_E)$, where $C^E = \{C_j, j = 1, \ldots, m\}$ is a set of categories such that*Type(C$_j$)* = $C_j$and $\subseteq_E$is a partial order on $\cup_j C_j$, the union of all dimensional values in the individual categories.

Definition 7. A *dimension category*C of type C is a set of member values $\{e_i, i = 1, \ldots, n\}$ such that*Type(e$_i$)* = *C.*

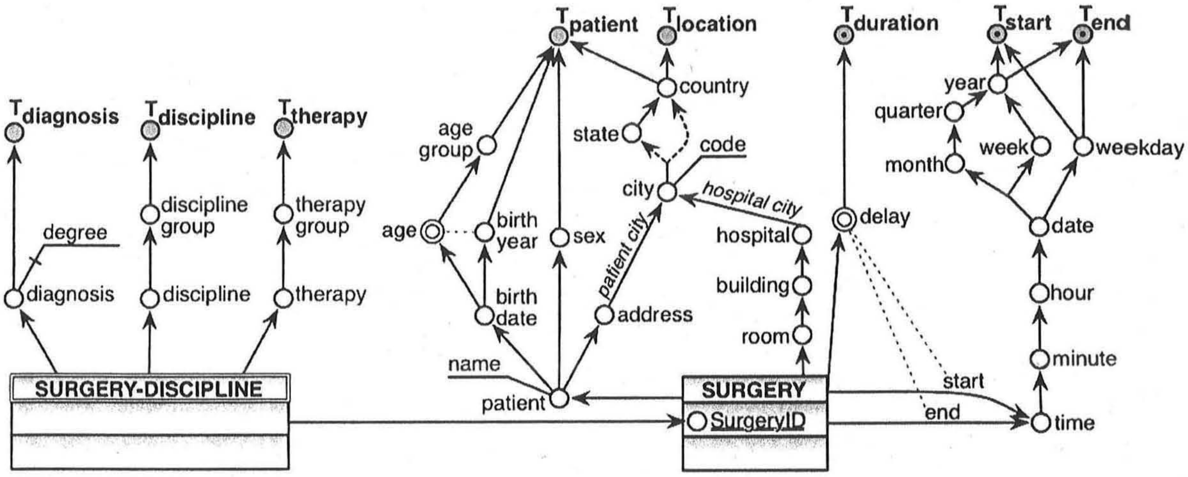
Distinction between the concepts *category* and *category type* is made in order to support modeling of fully and partially shared dimensions, in which the same category type, e.g., city, may be used as categories patient city, hospital city, etc.

Partial order $\subseteq_E$ on $\cup_j C_j$ is understood as follows: given $(e_1, e_2) \in \cup_j C_j$, $e_1 \subseteq e_2$, if e_1 is logically contained in e_2. Predicates $\subseteq$ and $\subseteq^{*}$ specify direct and transitive containment relationship, respectively, between a pair of member values. Apparently, containment relationships at the instance level are always full. The total number of members in category C_j is denoted $|C_j|$.

Figure 5 demonstrates the use of *X*-DFM for graphical modeling of multidimensional schemes. In this example, fact scheme SURGERY contains single surgical interventions as its fact entries. In *X*-DFM, each fact scheme is mapped to a box-shaped node holding the scheme's name, its measures, and degenerate dimensions. Dimension schemes are shown as directed graphs with categories as nodes and containment relationships between them as edges. Labeled circles represent dimension level attributes, while property attributes are terminal nodes shown as labeled lines and attached to their respective categories. Each dimension's graph finally converges at its top category (shaded circular nodes). A directed edge connecting a pair of nodes represents a many-to-one, i.e., a roll-up, relationship between them. Optional properties of a category, such as degree within the category diagnosis, are marked by placing a dash across their edges.

X-DFM provides unambiguous graphical constructs for all semantic elements of the model. An overview of the *X*-DFM constructs is given in the Appendix. Explanations of the constructs not yet mentioned will be provided as we proceed with the definitions of the corresponding formalisms. Further details of *X*-DFM can be found in (Mansmann & Scholl, 2008).

Figure 5. Multidimensional scheme fragment in X-DFM



ADVANCED ELEMENTS OF THE CONCEPTUAL MODEL

Classical designation of facts is to contain relevant measures of a 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 8. A fact scheme F is *measurable*, if it has a non-empty set of measures, i.e., $M^F \neq \varnothing$.

Technically, a fact type is given by a many-to-many relationship between a set of attributes. According to Kimball (1996), any many-to-many relationship is a fact by definition. 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 (1996). However, *fact table* is a logical design construct corresponding to the semantic concept of a *fact type*. We define a conceptual equivalent of factless fact tables.

Definition 9. A fact scheme F is *non-measurable*, if its set of measures is empty, i.e., $M^F = \varnothing$.

As explained in the previous section, non-measurable fact schemes are crucial for warehousing business process data as the former provide support for *event tracking* and *coverage* fact types. Event tracking facts model events as a robust set of many-to-many relationships between multiple dimensions, whereas coverage facts are used to track events that were eligible but did not happen (Kimball, 1996). Back to the fragment depicted in Figure 5, SURGERY is an example of a non-measurable event tracking fact type.

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, R. (1996) uses the concept of a *degenerate dimension* to handle such *id*-like attributes, while DFM treats them as *non-dimension attributes* of a fact. *Fact identifier* attribute is a special case of a degenerate dimension.

Definition 10. Dimension D is *degenerate*, if it has a single category C consisting of a single attribute, i.e., $C^D = \{C, T_D\} \wedge C = \{A^C, \varnothing\}$.

Definition 11. A degenerate dimension D **is a *fact identifier* in** F**, if all values of** D **in** F **are unique.**

Since a degenerate dimension is only valid in the context of its fact, *X*-DFM places the former inside the fact's node as shown in Figure 5. Fact identifiers, shown with a double-underlined name, provide the foundation for modeling multi-fact schemes, as discussed later in this section.

TYPES OF MULTI-FACT SCHEMES

There may exist a many-to-many mapping of a fact with some of its dimensional characteristic or even with another fact. Giovinazzo (2000) proposes a concept of a *degenerated 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. We suggest distinguishing between the following types of fact degeneration:

- *Satellite fact* scheme F ' extracts a many-to-many relationship between a fact scheme F and a dimension scheme D_i along with the corresponding measure characteristics of this relationship into a separate fact. Thereby, F acts as a dimension of F '. The term *satellite* reflects the accompanying nature of this fact with respect to its base fact.
- *Association fact* scheme F " extracts a many-to-many relationship between a pair of fact schemes F and F ' along with the corresponding measure characteristics of this relationship into a separate fact.
- *Self-association fact* F ' extracts a recursive relationship within a fact scheme F, converting the latter into two different dimensions in F '.

Consider a many-to-many relationship between SURGERY and PARTICIPANT in the E/R diagram (Figure 2). An attempt to map this relationship to a multidimensional scheme would yield a satellite fact SURGERY-PARTICIPANT, shown in Figure 6(a), with fee as a measure referring to that mapping. As an example of an association fact, consider a trigger relationship between the facts of type EVENT and ACTIVITY (e.g., event *X* triggered activity *Y*). Figure 6(b) shows the resulting EVENT-ACTIVITY association fact and its base facts acting as dimensions of the former. Similarly, a self-association of EVENT can be defined to store a trigger relationship between pairs of events and is also represented in Figure 6(b) as EVENT-EVENT scheme.

Similarly to dimension levels, facts may display a roll-up behavior, i.e., be in a many-to-one relationship with each other.

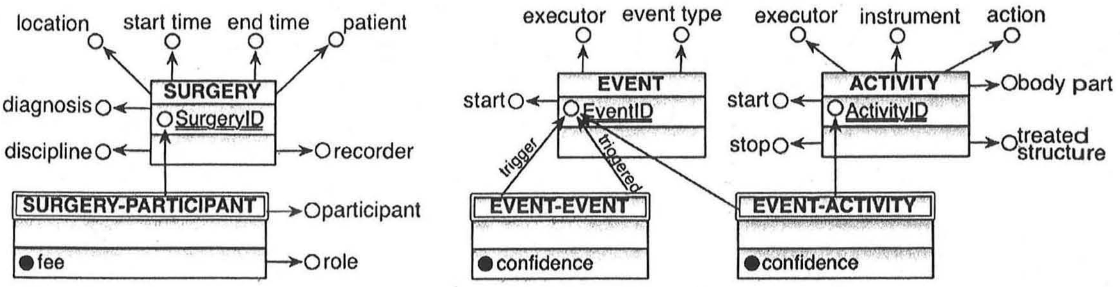
Definition 12. A pair of fact schemes F **and** F ' **form a *fact hierarchy*, or a *fact roll-up*,** F $\sqsubseteq^*$ F ', **if** F **has a dimension containing fact identifier of** F ' **as one of its categories at any level of the hierarchy.**

Intuitively, fact schemes form a roll-up if they represent different grains of the same process. Fact roll-up is *direct*, if fact identifier of F' serves as a bottom category in F, and is *transitive* otherwise. Hierarchical relationships between facts typically arise between event tracking schemes that model events at different grain. In our example, there is a transitive fact roll-up of ACTIVITY to SURGERY depicted in Figure 7(a), as category phase of ACTIVITY rolls up to SurgeryID, which is a fact identifier of SURGERY.

An object-oriented concept of *inheritance* is helpful for dealing with heterogeneity of fact entries. A surgical process consists of different types of components, such as activities and events, which have a subset of common properties as well type-specific ones. A *fact generalization* is obtained when heterogeneous fact types are extracted into a superclass fact type in part of their common characteristics.

In our example, EVENT and ACTIVITY are made subclasses of COMPONENT, as shown

Figure 6. Examples of satellite fact schemes



(a) SURGERY-PARTICIPANT as a satellite fact of SURGERY

(b) EVENT-ACTIVITY as an association and EVENT-EVENT as a self-association fact

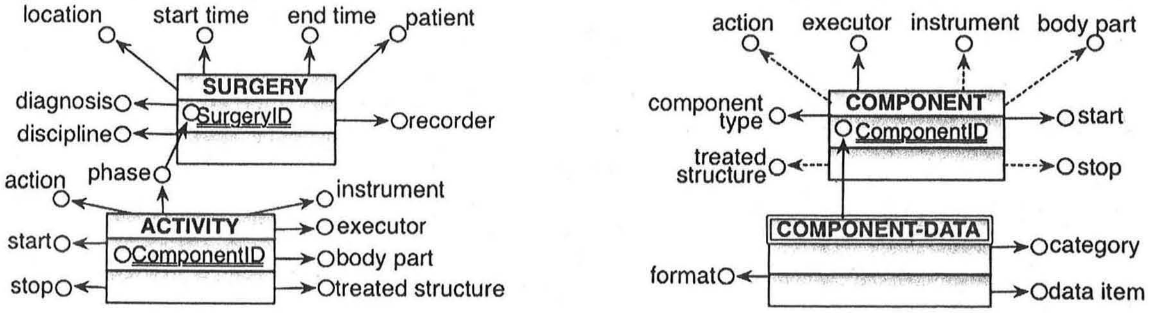
in Figure 8. The superclass extracts all those dimensions, which are shared by all its subclasses. Moreover, fact generalization enables modeling of the degenerate facts, common for all subclasses, at the superclass level. In our example, COMPONENT-DATA could be modeled as a satellite of the generalized fact scheme COMPONENT.

Finally, fact types can be divided into *homogeneous* and *heterogeneous*. A fact scheme is homogeneous, if it disallows partial roll-up relations between the fact and any of its dimensions, and is heterogeneous otherwise. Heterogeneous fact types result from mapping non-uniformly structured facts to the same type, i.e., avoiding specialization. Figure 7(b) shows a variant of COMPONENT modeled as a heterogeneous fact scheme storing all characteristics of both subclasses EVENT and ACTIVITY. Relationships with dimensions, not common for all subclasses, have to be modeled as optional (dashed-line edge).

Fact types considered so far are called *primary* as they store non-derived data. Facts derivable from other facts are called *secondary*. The latter can be further categorized according to the way they were obtained:

- *Summary* fact type contains measures from the base fact type, aggregated to a coarser granularity.
- *Drill-across* fact type contains measures obtained by combining multiple related fact types.
- *Partition* fact type contains a subset of fact entries from its base fact type.

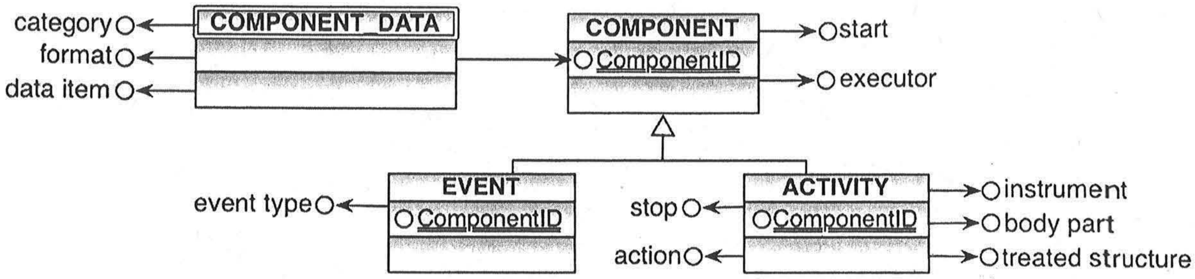
Figure 7. Examples of hierarchical relationships between fact schemes



(a) Fact rollup of ACTIVITY to SURGERY

(b) Heterogeneous fact type COMPONENT

Figure 8. Fact generalization of classes EVENT and ACTIVITY as a superclass COMPONENT



- *Conversion* fact type is obtained by applying a *push* and/or a *pull* operator.

MODELING DIMENSION HIERARCHIES

In the context of OLAP, only *structured* data hierarchies, i.e., those whose instances adhere to a certain scheme, qualify as dimensions. Values in a dimension may be organized into one or multiple hierarchies to provide additional levels of aggregation.

Definition 13. A *hierarchy scheme* H within D is a 5-tuple (C^H, $\sqsubseteq_C$, $\sqsubseteq_D$, T_D, $\perp_D$) for which holds: $\nexists(C_i, C_j, C_k) \in C^H$: $C_i \sqsubseteq^{(full)} C_j \wedge C_i \sqsubseteq^{(full)} C_k$, **i.e., no category has more than one full roll-up relationship.**

Definition 14. A *hierarchy instanceH*associated with hierarchy scheme H is a pair (C^H, $\subseteq_H$), where $C^H = \{C_j, j = 1, ..., m\}$ **is a set of categories such that***Type*(C_j) = C_j, $C_j \in C^H$, **and** $\subseteq_H$**is a partial order on** $\cup_j C_j$, **the union of all dimensional values in the individual categories.**

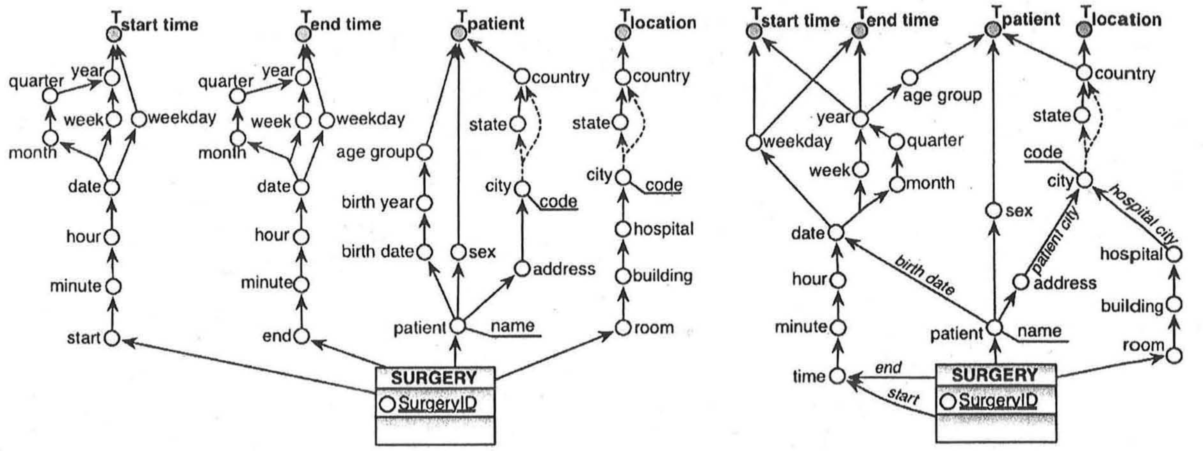
Decomposition of complex dimension schemes into their constituting hierarchy schemes is crucial for determining valid aggregation paths within a dimension. Consider the dimension scheme patient in Figure 9(a). Apparently, it is composed of multiple hierarchy schemes with the following sets of category types:

1. $\{\perp_{patient}$, sex, $T_{patient}\}$,
2. $\{\perp_{patient}$, birth date, birth year, $T_{patient}\}$,
3. $\{\perp_{patient}$, birth date, age, age group, $T_{patient}\}$,
4. $\{\perp_{patient}$, address, city, state, country, $T_{patient}\}$,
5. $\{\perp_{patient}$, address, city, country, $T_{patient}\}$.

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 patient address hierarchy, in which the members of city have parent members either in state or directly in the state's parent category country. Therefore, the last two hierarchies in the above enumeration can be considered parts of a single heterogeneous hierarchy. Further elaborations on heterogeneous hierarchies can be found in (Mansmann & Scholl, 2007; Malinowski & Zimányi, 2006; Hurtado & Mendelzon, 2002).

Multiple hierarchies in a dimension are of type *alternative* or *parallel* with respect to one another. *Multiple alternative* hierarchies are based on the same analysis criterion with at least one shared level in the dimension scheme. Time dimension is a classical example of multiple alternatives. In start time dimension in Figure 9(a), alternative paths emerge from the category date: date values can be grouped by month or by week. However, these two aggregation levels may not be used in combination due to an implicit many-to-many

Figure 9. Modeling shared dimensional elements in X-DFM



(a) A fact scheme without dimension category sharing categories

(b) A fact scheme with shared

relationship between the members of those categories: each month consists of multiple weeks and a week belongs to one or two months. *Parallel* hierarchies in a dimension account for different analysis criteria, such as the following patient hierarchies from the above list: the first hierarchy is based on the sex criterion, the third one groups patients by age, whereas the forth one is a hierarchy of patients' addresses. These three criteria have no relation to one another and, therefore, can be used in combination for aggregation. For instance, patient members can be first grouped by sex, and then by birth year, or vice versa.

Another important concept in the dimensional modeling is that of *derived* categories and dimensions. New categories may be derived as functions of the existing ones. For example, category age in Figure 5 is derived from birth year (by subtracting the birth year from the current year). Derived categories can be used in dimension schemes on the same terms as basic categories, as they provide additional aggregation levels. A category, derived from a bottom category or a set of bottom categories in a fact scheme, qualifies to be treated as a derived dimension of that fact scheme, since the former represents a derived characteristic of the fact itself. For instance, dimension duration in Figure 5 is a derived one, as its bottom category delay is derived from the bottom categories start and end (by subtracting start values from those of end).

UNIFICATION OF THE MULTIDIMENSIONAL SPACE

A set of dimensions of any given fact scheme represents the multidimensional space of that scheme. Intuitively, the common multidimensional space of a set of fact schemes encompasses all dimensions of those schemes. For proper modeling of multi-fact relationships as well as of the convergence of dimension hierarchies, isolated multidimensional spaces have to be unified by replacing each set of redundant categories with a single shared category. Our approach to the unification of the multidimensional space is based on distinguishing between the concepts of a *dimension category* and a *category type*. Since a category type describes the value domain of a category, it is possible to define multiple categories of the same type. In terms of the unified space S, categories are considered redundant, if they belong to the same category type. To formalize the above idea, we introduce the concepts of *compatible*, *conformed*, and *related* elements.

Definition 15. Categories C_i and C_j are *compatible*, if they belong to the same category type:$Compatible(C_i, C_{jj}) \Leftarrow (C_i \neq C_j \wedge Type(C_i) = Type(C_{jj}))$.

Top-level categories are considered distinct for each dimension 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 data subset. Therefore, top level categories are exempted from the compatibility test. In a unified space S, each set of compatible categories is modeled as one *shared* category type.

Definition 16. Compatible categories C_i and C_j are *conformed*, if they roll up along the same paths:$Conformed(C_i, C_{jj}) \Leftarrow Compatible(C_i, C_{jj}) \wedge (\forall C_m, C_i \sqsubseteq C_m: \exists C_n, C_j \sqsubseteq C_n \wedge Conformed(C_m, C_n))$.

Conformed categories are fully compatible because they roll-up along the same path. Back to Figure 9(a), start and end categories in start time and end time, respectively, are conformed, whereas date in start time and birth date in patient are compatible (the same value domain), but not conformed (different roll-up paths).

From category compatibility, the notions of *related dimensions* and *related fact schemes* are inferred:

Definition 17. A pair of dimensions D_i and D_j are *related*, if their schemes share at least one category type:$Related(D_i, D_j) \Leftarrow \exists C_m \in D_i, \exists C_n \in D_j: C_m = C_n$.

Definition 18. A pair of fact schemes F and F ' are *related*, if they have at least one pair of related dimensions:$Related(F, F') \Leftarrow \exists D_i \in F, \exists D_j \in F': Related(D_i, D_j)$.

With respect to dimension sharing, *X*-DFM can be used in different modes, such as (a) *non-shared*, (b) *partially shared*, and (c) *fully shared* mode. In a 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 9(a). In the partially shared mode, only conformed categories are considered shared. This mode was applied in the scheme shown in Figure 5, where compatible yet non-conformed categories birth date and date along with their aggregation paths were not merged. In the fully shared mode all compatible categories are represented as shared nodes, thus complying with the requirements of the unified multidimensional space.

In the fully shared *X*-DFM mode, compatible categories are represented as follows:

1. Proceeding from the bottom-level categories upwards, each set of conformed categories is merged into a single category type node. Subsequently, the same is done for the remaining compatible categories.
2. Shared nodes are labeled by the name of their category types.
3. The actual names of single categories behind the shared node are shown as labels of the respective incoming roll-up edges.
4. 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 equal to its category type name.
5. 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 9 pictures the concept of modeling shared dimensions at the example of the fact scheme SURGERY. Figure 9(a) shows the initial state of the model, in which each category is represented by a distinct node in the scheme. Applying the above rules of presenting shared categories in a unified multidimensional space, we derive a scheme depicted in Figure 9(b). Dimensions start time and end time now appear fully merged as their schemes are identical. The bottom categories are merged into a node of type time, whereas category names start and end are shown as edge labels. Dimensions patient and location are partially shared as both of them contain a category of type city.

In case of conformed categories, the entire roll-up graphs rooted at those categories can be merged in a single step. In case of non-conformed categories, graph merging may appear 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 multiple alternative hierarchies. At this stage, the designer has to decide, whether these roll-up relationships should also be made available for birth date. In that case, the category birth year is simply mapped to year, as shown in Figure 9(b).

Category age group, however, which is a parent of year in patient, does not appear feasible as an aggregation level in start time or stop time dimensions, and, therefore, it is not added to their schemes.

With respect to the degree of convergence, three levels of dimension sharing can be identified, namely (a) *conformance*, (b) *inclusion*, and (c) *overlap*. Any of these patterns may occur between dimensions belonging to the same or to different fact schemes.

Definition 19. A pair of dimensions*D*and*D*' are *conformed*, if their bottom categories are conformed:$Conformed(D, D') \Leftarrow \exists C_i \in D, Type(C_i) = \perp_{D}, \exists C_j \in D', Type(C_j) = \perp_{D'}: Conformed(C_i, C_{jj})$.

Since category conformance is defined as a recursive property, dimension conformance implies the identity of the respective dimension schemes, or formally: $Conformed(D, D') \Leftrightarrow C \setminus \{T_D\} = C' \setminus \{T_{D'}\} \wedge \sqsubseteq_D = \sqsubseteq_{D'}$. As an example of conformed dimensions, consider start time and end time of SURGERY in Figure 9(b).

Kimball & Ross (2002) introduced the term *conformed dimensions* to refer to dimensions, which are not physically centralized but which have identical schemes. Our definition differs from the latter one in that we do not regard logical design issues (e.g., centralization and normalization) at the stage of conceptual modeling. Therefore, in our model, a unified multidimensional space approach does not impose any particular logical or physical design scheme. On contrary, this approach is beneficial for generating semantically rich metadata to support advanced OLAP operators and data navigation options in frontend tools irrespective of the implementation.

Inclusion pattern of dimension sharing occurs when some category in a dimension fully rolls-up to the bottom-level category of another dimension, i.e., when two dimensions represent different grain of the same characteristic. In our scenario, this is the case with the dimensions patient of SURGERY and treated structure of ACTIVITY. Bottom-level category of patient serves as an upper aggregation level in treated structure. As a result, ACTIVITY facts, if grouped by treated structure, can be further aggregated along the entire dimension scheme of patient.

Definition 20. Dimension*D*is *included* in dimension*D*', if its scheme is a sub-graph in the scheme of*D*':$Included(D, D') \Leftarrow C \setminus \{T_D\} \subset C' \setminus \{T_{D'}\} \wedge \sqsubseteq_D \subset \sqsubseteq_{D'}$.

Dimensions are said to be *overlapping*, if their schemes converge only partially.

Definition 21. A pair of dimensions*D*and*D*' are*overlapping*, if they are related via a category, non-bottom for either of them:$Overlapping(D, D') \Leftarrow \exists C_i \in C \setminus \{\perp_D\}, \exists C_j \in C' \setminus \{T_{D'}\}: C_i = C_j$.

Overlapping dimensions may belong to the same or to different fact schemes. The latter provides inherent support for a drill-across operation. Dimensions patient and location in Figure 9(b) overlap as they contain hierarchies that converge in city.

Notice how presence of distinct top-level categories helps to distinguish between seemingly and truly converging paths. The former case occurs in case of category sharing between dimensions. For instance, even though country is the highest aggregation level in both location and patient, each of these dimensions ends at its own top-level node. True path convergence occurs in multiple and heterogeneous hierarchies within a dimension, as in the case of start time and end time, where multiple paths converge in year.

"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 "rolls-up-to" 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 (Abelló, Samos, & Saltor, 2001). Fact and dimension roles are fixed only in the context of isolated fact schemes. What happens to those roles in the context multidimensional multi-fact schemes? Apparently, 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 dimensions. Altogether, multidimensionality implies that what is considered a fact in one task could be considered a dimension by another one, and vice versa.

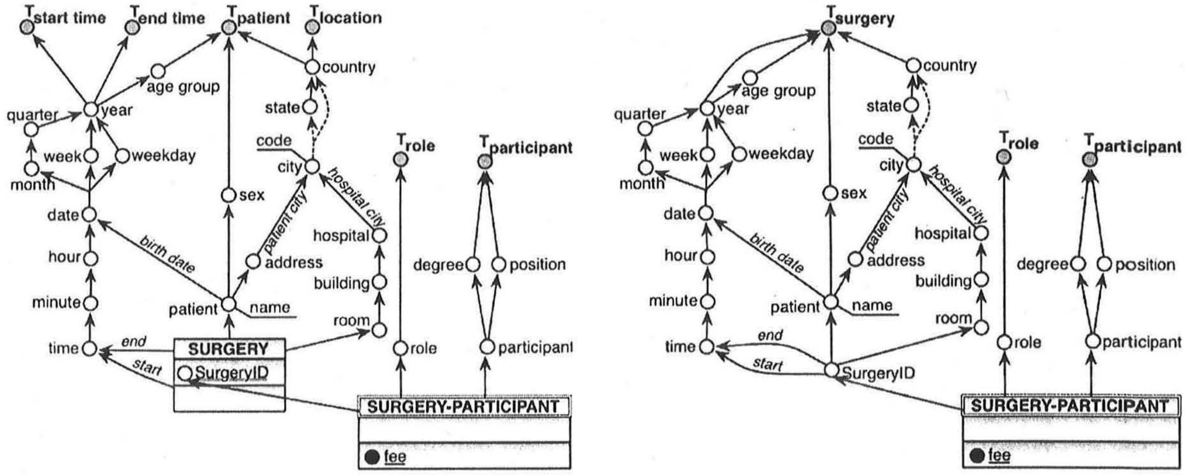
The first interchangeability case is concerned with a fact scheme acting as dimension of another fact scheme. Fact scheme F can be treated as a dimension in fact scheme F' while querying its measures when F' contains the fact identifier dimension of F. This relationship may be encountered in satellite facts and hierarchies of event tracking facts. One implication of this interchangeability is that it results in multiple focus-dependent conceptual schemes for the same data fragment. Figure 10 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 10(a) and a perspective focused on SURGERY-PARTICIPANT and its valid aggregation paths is given in Figure 10(b).

Thereby, fact scheme SURGERY is 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 treating 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 rolls-up) relationship to all other dimensions.

Another kind of interchangeability is related to treating dimensions as measures, and vice versa. Support of advanced OLAP operators, such as *push* for converting a dimension category into a measure and *pull* for converting a measure into a dimension, as well as *drill-across* for combining measures from multiple related fact schemes, is a challenge not handled by conventional conceptual models. The output of these operators is a new conceptual multidimensional scheme. Our solution for supporting scheme-transforming operators at the conceptual level is straightforward - to explicitly model their output schemes. Figure 11 exemplifies this idea by showing the conceptual consequences of "pushing" a dimension category hospital in SURGERY (see Figure 9(b)) into a measure attribute (e.g., to query a measure 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 granularities become available. Dashed lines connecting the measure attribute hospital with all dimensions indicate its non-additivity.

*Figure 10. Fact **SURGERY** as a dimension in its satellite fact **SURGERY-PARTICIPANT***



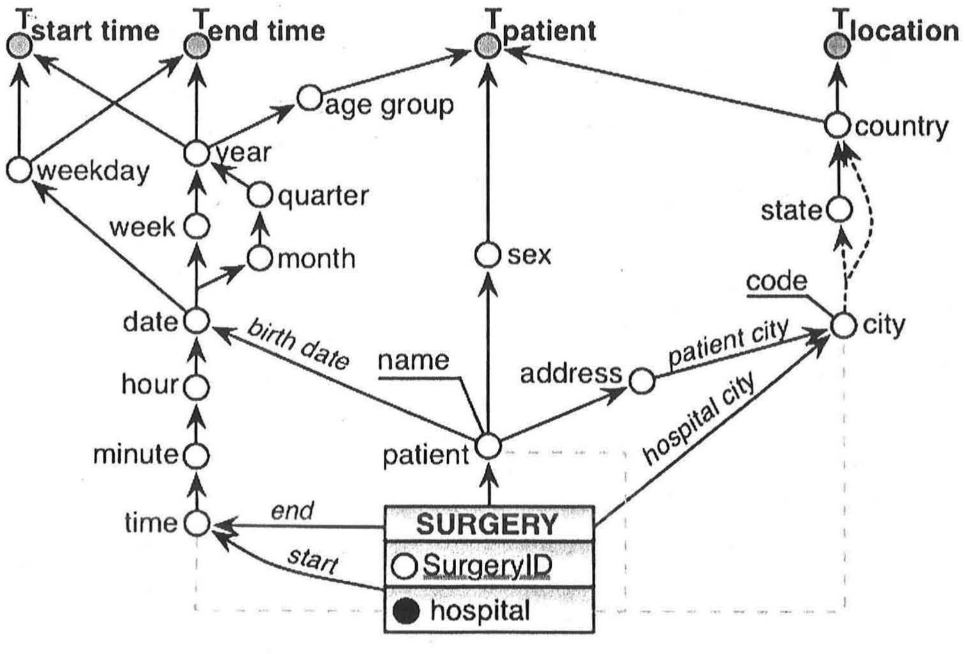
(a) Focus-independent view of a satellite fact scheme

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

OBTAINING A MULTIDIMENSIONAL SCHEME OF A PROCESS

In the two preceding sections we formalized the properties of the advanced multidimensional conceptual model that overcomes the restrictions of the conventional OLAP technology. The presented formalisms were illustrated using relevant multidimensional fragments from the cases study. However, we did not elaborate on how those frag-

*Figure 11. Transformation of the original fact scheme **SURGERY** caused by a push operation*



ments had actually been obtained. The algorithm of acquiring the multidimensional model of a process is the subject of this section.

The idea of developing methods for systematic acquisition of multidimensional models from E/R diagrams is well represented in the data warehouse research. Most of the existing business information management systems are relational, and, therefore, it appears feasible to derive the conceptual model of a data warehouse from that of the existing system, typically available in form of E/R or UML class diagrams. Outstanding contributions in this field were made by Cabibbo & Torlone (1998), Golfarelli, et al. (1998), Franconi & Sattler (1999), Tryfona, et al. (1999), Phipps & Davis (2002). Some of the approaches, such as the ones proposed by Cabibbo & Torlone (1998) and Franconi & Sattler (1999), are based on "encoding" the 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 are starER of Tryfona, et al. (1999) and *Multidimensional Entity Relationship (ME/R) Model* of Sapia, et al. (1999). Yet another group of works provides mapping of E/R schemes to ad-hoc multidimensional models. The DFM approach of Golfarelli, et al. (1998), which is the predecessor of our proposed *X*-DFM model, is an example of such methodology.

The above methods proceed by determining the facts and subsequently refining their dimensional context. However, none of those methods is directly applicable in our scenario due to their fundamental assumption that the measures of interests are known at design time. Dealing with a "factless" event-tracking data warehouse application scenario implies the necessity for a different procedure of identifying the facts.

Our approach to identifying candidate fact entities in an E/R scheme is based on analyzing the set of each entity's relationships with other entities by looking at the cardinalities and structural constraints of those relationships. From the basic definitions of facts, dimensions, and dimension hierarchies provided in Section 6, as well as the definitions of degenerate facts and dimensions in Section 7, 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 guidance for automatic recognition of fact and dimensions candidates in E/R schemes, subject to the condition that the input scheme accurately and fully maps all required attributes as well as relationships and dependencies between attributes.

VERIFICATION AND REFINEMENT OF THE E/R SCHEME

In most cases, pre-existing conceptual models of the system are tailored towards specific application needs and are thus focused on the properties and relationships relevant in the application context. Besides, the level of detail, accuracy and completeness of the model may not be adequate to meet 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: (a) pruning / enriching the data set and (b) refining the relationships in the data.

The data set is pruned as 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 available for the analysis. This data is obtained by integrating additional data sources. 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 data by zip code, city, region, and so on.

The aim of the refinement phase is to have an accurate mapping of all relationships between all entities 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 correspond to that of an *attribute* in the E/R model, 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 specify cardinalities as simple ratios (1:1, 1:*n*, or *m*:*n*) as this notation does not reveal whether the relationship is optional for any of participating entity types. Therefore, representation of cardinality by 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 are fundamental for formulating the ultimate goal of approximating an E/R scheme to a multidimensional one.

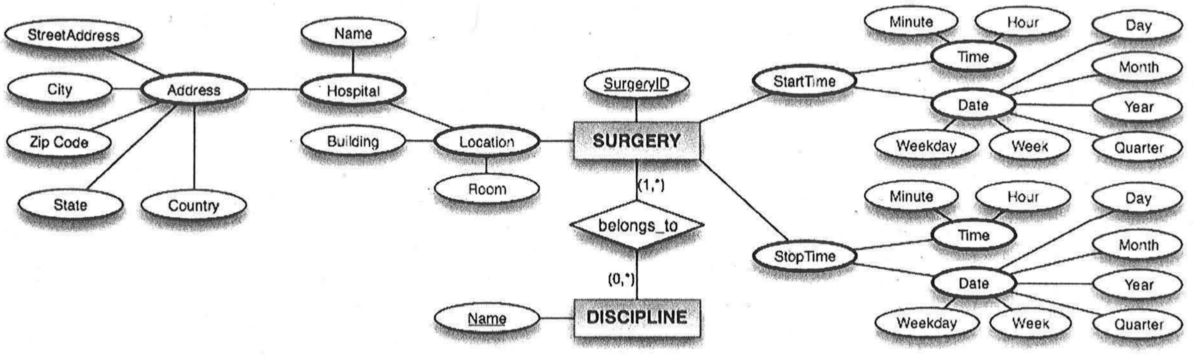
Definition 22. An E/R scheme is accurate, if the structural constraints are fully specified for each relationship typeRand each entity typeEparticipating inR, if all generalization / specialization relationships are made explicit, and if for each attribute$A_i$in the scheme holds:

1. ***$A_i$is simple (i.e., non-composite),***
2. ***$A_i$is single-valued,***
3. **$A_i$is either a key property (or a part of the key) or is functionally dependent on the key property,**
4. ***$A_i$is not related (i.e., has no functional dependency) to any other attribute apart from the key of its entity type.***

To achieve the above accurate state, we propose the transformation procedure that evolves as follows:

1. Identify implicitly composite attributes (i.e., consisting of multiple data fields) and replace

Figure 12. ***Examplesof presenting complex attributes as composite ones and re-modeling multivalued attributes into related entity types***



them by explicit composite attributes.

2. Similarly, re-shape explicit composite attributes into entity types consisting of simple attributes.
3. Multivalued attributes are reshaped into entity types, related to that attribute's original entity type.
4. Identify dependencies and relations between attributes, not explicit in the scheme. Each attribute, involved into 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.
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 2.

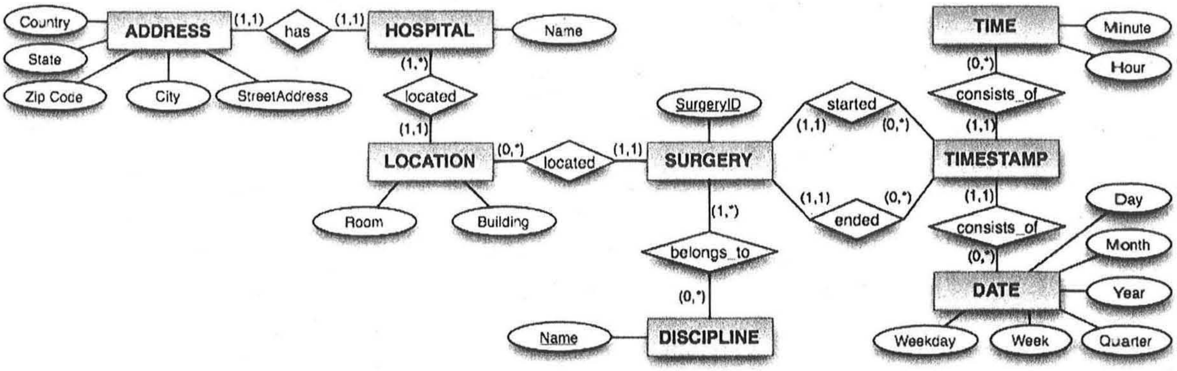
In the first step, attribute Location was identified as implicitly composite, as its values are full addresses of respective operating theatres specified as the room, the building, the name of the hospital and its full address. The address values, in their turn, are also decomposable into multiple fields. Similarly, attributes of type date and time should be decomposed into their constituent fields. Figure 12 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 13 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, consider the result of transforming Discipline attribute into an entity type, depicted in Figure 12.

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

Figure 13. Transforming composite attributes into related entity types



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 14 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 2) 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 15 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, which have an address property, with the same entity type ADDRESS. Some elements become obsolete at different stages of refinement. For example, entity type LOCATION (Figure 13) gets dissolved into ROOM and

*Figure 14. **Transforming attributes into entity types to reveal implied roll-up relationships between them***

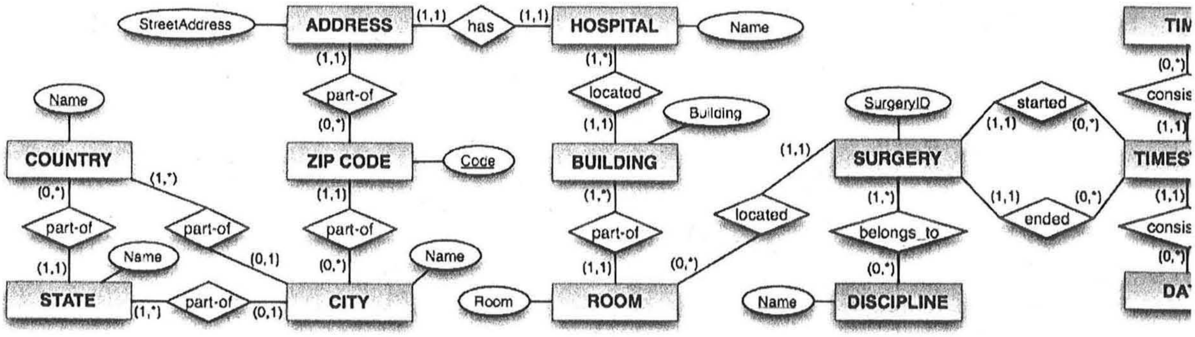
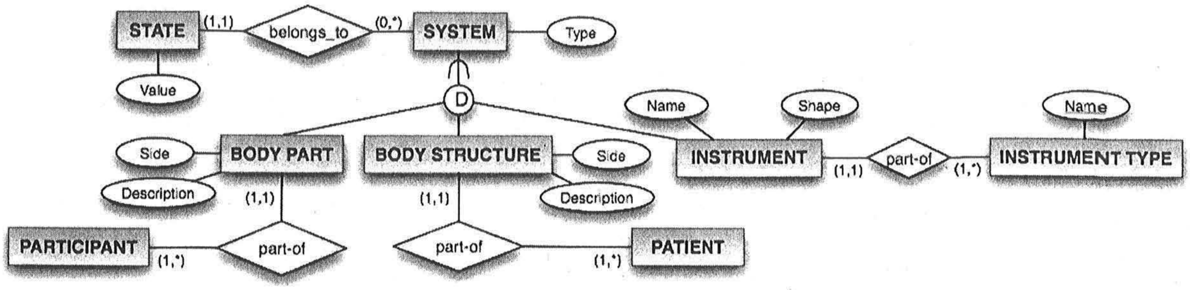


Figure 15. Adding specialization to the heterogeneous entity type ***SYSTEM***



BUILDING along with a "part-of" relationship between them (Figure 14). In the final step, the scheme is verified to ensure that it contains no obsolete elements.

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.

Since facts build the focus of a multidimensional scheme, the first step is concerned with identifying fact candidates. 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 measure(s). 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.

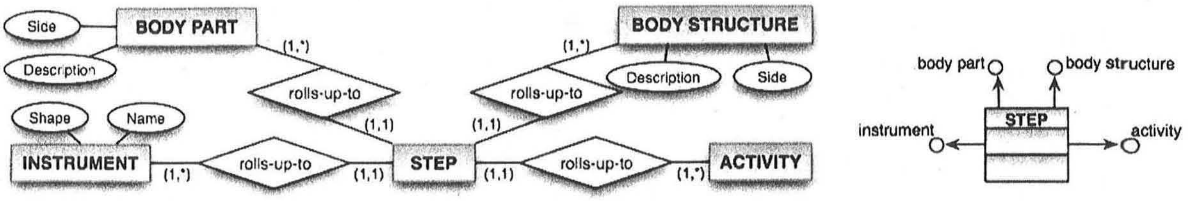
For the sake of simplicity, the first two cases can be merged into one, since any *n*-ary relationship is convertible into an entity type by replacing each branch with a binary relationship towards the respective participating entity type. Besides, the concept of an entity type is generally superior 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 granularity, and 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:

- $E^{<rec>}(E_f)$ is a set of recursive (i.e., connecting the entity type to itself) relationships of E_f,
- $E^{<n:1>}(E_f)$ 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,

Figure 16. Transforming entity type ***STEP*** *(left) to a fact scheme (right)*



- $\mathrm{E}^{(<super>)}(E_f)$ is a set of superclasses, i.e., direct generalizations, of E_f,
- $\mathrm{E}^{(<sub>)}(E_f)$ is a set of subclasses, i.e., direct specializations, of E_f,
- $\mathrm{E}^{<1:1>}(E_f)$ 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,
- $\mathrm{E}^{<1:n>}(E_f)$ is a set of E_f's candidate sub-facts, i.e., a set of entity types with which E_f has a 1:*n* relationship,
- $\mathrm{E}^{<m:n>}(E_f)$ 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 an E/R scheme into a multidimensional one evolves in a bottom-up fashion, starting with entity types that qualify as terminal facts, i.e., the elements of the finest grain, and proceeding tö coarser grained elements.

Definition 23. Entity typeE_f corresponds to a *terminal* fact, if it is not involved into any decomposition or specialization relationship, i.e., $\mathrm{E}^{<1:n>}(E_f) = \mathrm{E}^{(<sub>)}(E_f) = \varnothing$.

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 E_f implies that each subclass inherits all characteristics of E_f and may have further characteristics of its own.

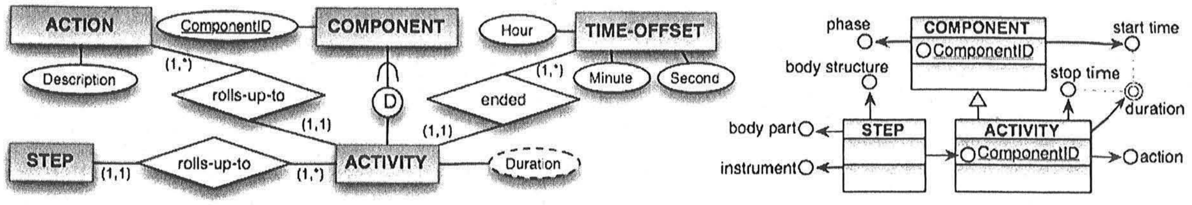
In our surgical workflow model, entity types STEP, EVENT, and STATE qualify as terminal facts. Figure 16 shows the part of the E/R diagram referring to STEP and its relationships types as well as its mapping to a 4-dimensional fact scheme. For consistency, *n*:1 relationship with full participation, i.e., with (1,1) and (1,*) as its structural constraints, are all renamed to "rolls-up-to". The transformation appears straightforward as the only non-empty set of related categories $\mathbf{E}^{<n:1>}$(STEP) = {INSTRUMENT, BODY PART, BODY STRUCTURE, 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 17, with its non-empty sets $\mathbf{E}^{<n:1>}$(ACTIVITY) = {TIME-OFFSET,ACTION}, $\mathbf{E}^{<super>}$(ACTIVITY) = {COMPONENT}, and $\mathbf{E}^{<1:n>}$(ACTIVITY) = {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, let us consider an example of identifying and modeling degenerated facts. Once entity type E_f has been converted to a fact, its degenerated facts correspond to the relationships in $\mathbf{E}^{<m:n>}(E_f)$ (satellite facts and fact associations) and $\mathbf{E}^{<rec>}(E_f)$ (fact self-associations). Figure 18 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 18.

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

*Figure 17. Transforming entity type **ACTIVITY** (left) to a fact scheme (right)*



algorithmic description of acquiring fact schemes from accurate E/R schemes. Algorithm 1 (Figure 19) 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 fact candidate. Sets $\mathbf{E}^{(<sub>)}(E_f)$ and $\mathbf{E}^{(<1:n>)}(E_f)$ 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 (Figure 19) creates an empty fact type and converts the attributes of the underlying entity into measures and degenerate dimensions, as shown in the subroutine Algorithm 2 (Figure 20).

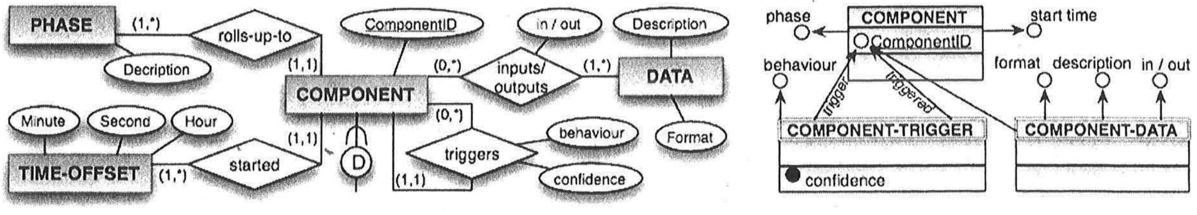
IDENTIFYING DIMENSIONS

Fact schemes produced by Algorithm 1 (Figure 19) are incomplete in a sense that fact's dimensions are defined solely in terms of their bottom categories. Therefore, the next step consists in constructing complete dimension hierarchies implied by the E/R scheme. Once the E/R scheme has been brought into an accurate state, as defined in the previous subsection, dimension hierarchies become easily identifiable: each category 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 will contain diverging and converging paths.

Roll-up behavior 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 21.

From the perspective of a candidate dimension category given by the entity type E_d, possible roll-up behaviors of E_d can be categorized based on the number of its relevant relationships, their

*Figure 18. Transforming m:n and recursive relationships of **COMPONENT** (left) to degenerated facts (right)*



structural constraints and inter-dependencies:

Homogeneous (non-)hierarchy emerges in the existence of at most one relevant relationship:

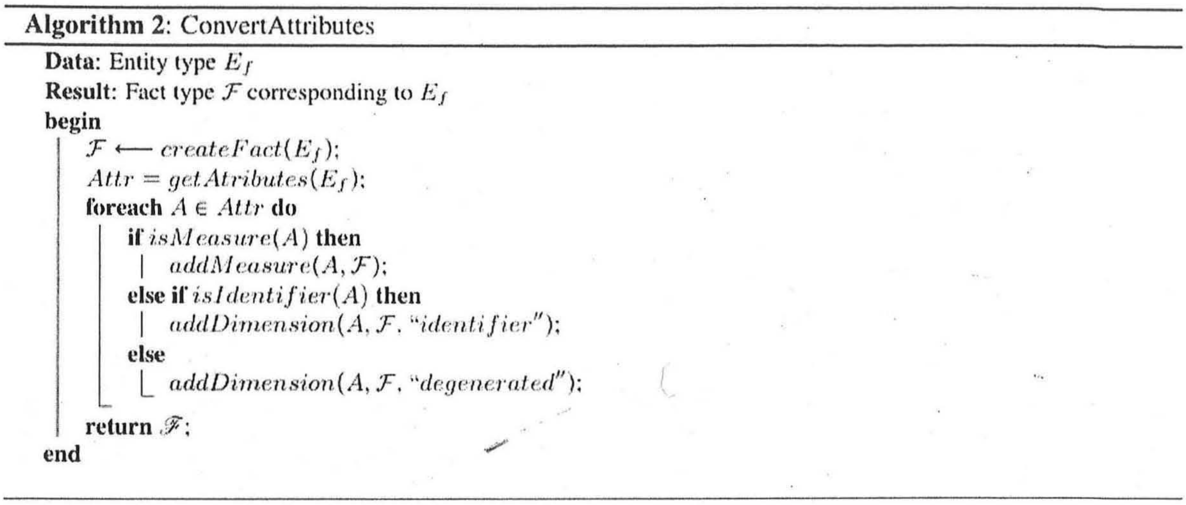
- *Non-hierarchy* is given, if E_d is not involved into any relevant relationship. In Figure 21, RECORDER would be a non-

Figure 19. Algorithm 1

```
Algorithm 1: ConvertToFact
Data: Entity type $E_f$, Set of previously identified fact schemes $\mathscr{F}$
Result: Updated set of fact schemes $\mathscr{F}$
begin
    $\mathcal{F} \longleftarrow ConvertAttributes(E_f, \mathcal{F})$;
    $\mathcal{E}^{<rec>} \longleftarrow \emptyset$;
    $\mathcal{E}^{<super>} \longleftarrow \emptyset$;
    $\mathcal{E}^{<1:1>} \longleftarrow \emptyset$;
    $\mathcal{E}^{<n:1>} \longleftarrow \emptyset$;
    $\mathcal{E}^{<m:n>} \longleftarrow \emptyset$;
    $Rel \longleftarrow getRelationships(E_f)$;
    foreach $E_f \diamond E_i \in Rel$ do
        if $E_f = E_i$ then
            $append(E_f \diamond E_i, \mathcal{E}^{<rec>})$;
        else if $E_i = Generalization(E_f)$ then
            $append(E_i, \mathcal{E}^{<super>})$;
        else
            $c = Cardinality(E_f \diamond E_i)$;
            switch $c$ do
                case $1 : 1$
                    $append(E_i, \mathcal{E}^{<1:1>})$;
                case $n : 1$ $append(E_i, \mathcal{E}^{<n:1>})$;
                otherwise
                    $append(E_i, \mathcal{E}^{<m:n>})$;

    foreach $E_i \in \mathcal{E}^{<1:1>}$ do
        $addDimension(E_i, \mathcal{F}, "shadow")$;
    foreach $E_i \in \mathcal{E}^{<n:1>}$ do
        $addDimension(E_i, \mathcal{F}, "normal")$;
        if $qualifiesAsFact(E_i)$ then
            $\mathscr{F} \longleftarrow ConvertToFact(E_i, \mathscr{F})$;
    foreach $E_i \in \mathcal{E}^{<super>}$ do
        $addDimension(E_i, \mathcal{F}, "superclass")$;
        $\mathscr{F} \longleftarrow ConvertToFact(E_i, \mathscr{F})$;
    foreach $E_f \diamond E_i \in \mathcal{E}^{<rec>}$ do
        $\mathcal{F}_k \longleftarrow CreateFactSelfAssociation(\mathcal{F}, E_f \diamond E_i)$;
        $append(\mathcal{F}_k, \mathscr{F})$;
    foreach $E_i \in \mathcal{E}^{<m:n>}$ do
        $\mathcal{F}_k \longleftarrow CreateDegenerateFact(\mathcal{F}, E_i)$;
        $append(\mathcal{F}_k, \mathscr{F})$;
    foreach $E_i \in \mathcal{E}^{<1:n>}$ do
        $addDimension(E_i, \mathcal{F}, "normal")$;
    $append(\mathcal{F}, \mathscr{F})$;
    return $\mathscr{F}$;
end
```

Figure 20. Algorithm 2



```
Algorithm 2: ConvertAttributes
Data: Entity type $E_f$
Result: Fact type $\mathcal{F}$ corresponding to $E_f$
begin
    $\mathcal{F} \longleftarrow createFact(E_f)$;
    $Attr = getAtributes(E_f)$;
    foreach $A \in Attr$ do
        if $isMeasure(A)$ then
            $addMeasure(A, \mathcal{F})$;
        else if $isIdentifier(A)$ then
            $addDimension(A, \mathcal{F}, "identifier")$;
        else
            $addDimension(A, \mathcal{F}, "degenerated")$;
    return $\mathscr{F}$;
end
```

hierarchical dimension in the fact scheme WORKFLOW.

- *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 as this relationships produces a 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.

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

*Figure 21. **Fragment of the E/R scheme relevant for building the dimension scheme phase in COMPONENT***

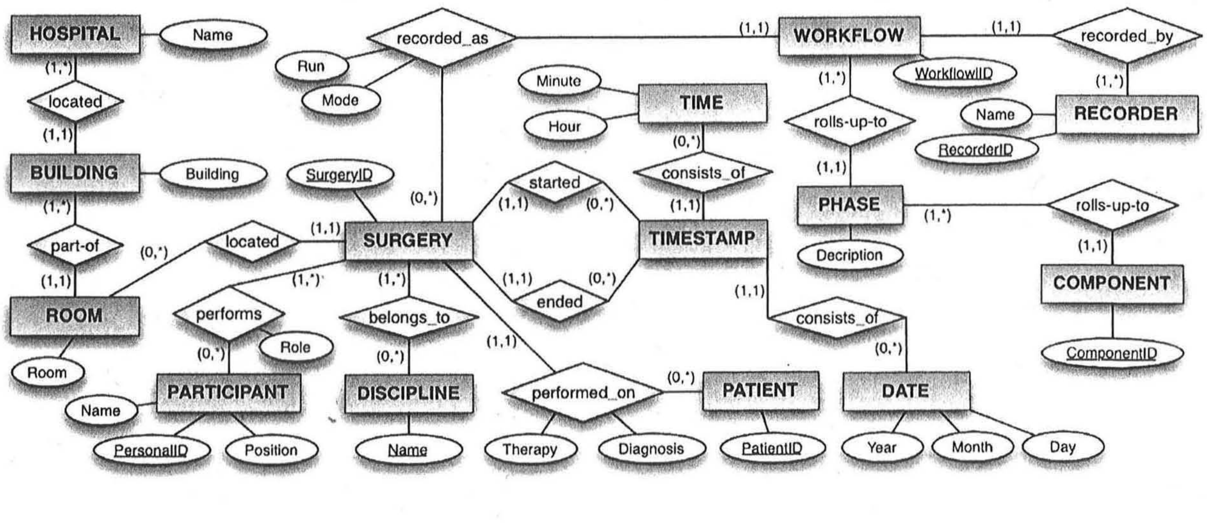
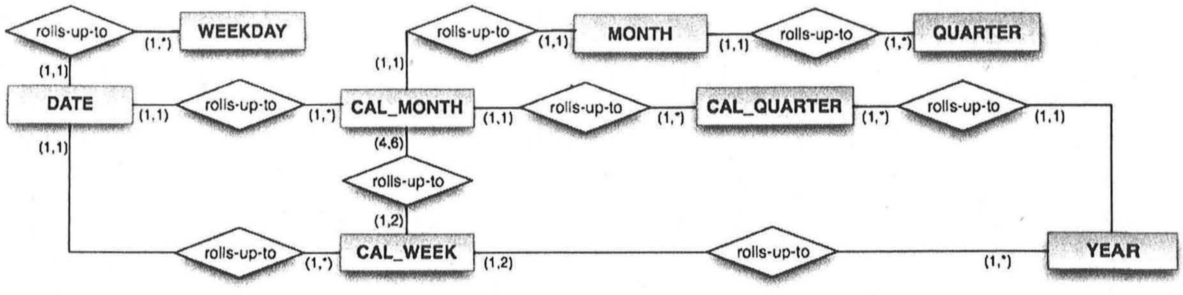


*Figure 22. Multiple alternative and parallel hierarchies in **DATE** dimension*



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 partial related 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. Example of such partial related roll-up is the relationship between CITY, STATE, and COUNTRY in Figure 14.
- *Specialization hierarchy* emerges from a specialization relationship of E_d into multiple subclass categories. As an example, consider a generalized category SYSTEM in Figure 15.

Multiple hierarchies correspond to multiple relevant relationships that are mutually nonexclusive. Figure 22 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 mutually unrelated entity types. For instance, the relationship of DATE with CAL_MONTH is parallel to that of DATE and WEEKDAY.

Figure 23 shows the results of converging the fragment of the E/R model from Figure 21 into a dimension. Additionally, the structure of the hierarchical category DATE is shown corresponding to the E/R model shown in Figure 20.

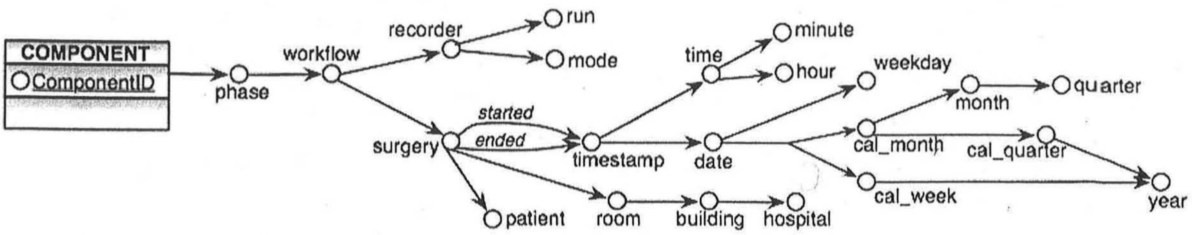
Once the construction of the dimension scheme is complete, an abstract top-level category is added as a root node at which all dimension's hierarchies converge. In case of a unified multidimensional space, redundant elements of dimension schemes have to be eliminated by merging compatible categories.

Since dimension hierarchy modeling techniques are well highlighted in the data warehousing literature, we omit further details of the methodology for obtaining dimensions from the E/R schemes.

EVALUATION OF THE PROPOSED DESIGN

In the previous sections we focused on the conceptual data modeling for BPI applications. The data warehouse is implemented by transforming the conceptual scheme into a logical and, finally, a physical one. Once the data warehouse is set

Figure 23. The resulting dimension scheme of the ***PHASE*** *dimension in* ***COMPONENT***



up and running, end-users access the data using so called OLAP tools. Advanced tools offer a user-friendly visual interface for interactive data analysis by implementing OLAP operators in form of interactive events, such as browsing, clicking, marking regions of interest, drag-&-drop, zooming, panning, etc., and by providing a set visual layouts (pivot tables, business charts, scatter-plots, dash boards, etc.) for convenient exploration of the retrieved data.

IMPLEMENTATION REMARKS

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)* systems store data in relational DBMS and employ SQL extensions and specialized access structures to efficiently implement OLAP operations.
- *Multidimensional OLAP (MOLAP)* systems directly 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 MOLAP and ROLAP databases: MOLAP pursues a top-down approach by first focusing on business problems, 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 (Dodds, et al., 1999). 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 very large data volumes. ROLAP attributes its success to the established and proven technology, good scalability in terms of the number of facts and their dimensionality, flexibility for cube redefinitions, and support for frequent updates (Pedersen & Jensen, 2001).

Considering the complexity of the conceptual modeling for BPI applications, the relational technology appears an adequate option. Especially the bottom-up design approach relying on the existing models and data sources and the ability to adjust and modify cube definitions at runtime make ROLAP an attractive option. Besides, the relational data model with its normalization techniques, integrity constraints and object-relational features has the necessary flexibility to adequately map advanced concepts of the semantic model.

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. *Star schema* and *snowflake schema* – both introduced by Kimball (1996) – are the two options of the relational data warehouse design. Both schemata are composed of a *fact table* and a set of associated *dimension tables*. Star schema places the entire dimension hierarchy into a single relation by pre-joining all aggregation levels, while snowflake schema decomposes complex dimensions into separate tables according to the relational normalization rules. Snowflake schema becomes the only option when dimensional hierarchies are prone to irregularities, such as heterogeneity, non-strictness, missing values, mixed granularity etc. Multiple fact tables with dimensions modeled using either star schema or snowflake schema may be arranged into a *galaxy* (Kimball, et al., 1998), also referred to as *fact constellation*. This schema is constructed by allowing dimension tables to be shared amongst many fact tables: each fact table is explicitly assigned to the dimensions, relevant for that fact table. This solution is very flexible and powerful as it offers a logical equivalent of a unified multidimensional space. A comprehensive methodology for obtaining a fact constellation schema from semantic schemes was proposed by Lechtenbörger (2001).

Relational concepts of virtual tables (known as "*views*") and *materialized views* are helpful for modeling derived elements in fact and dimension schemes. Foreign key constraints are used to link related schemes. Object-relational feature of inheritance enables intuitive handling of heterogeneous facts and dimensions.

VISUAL ANALYSIS

Visual exploration has evolved into the prevailing method of modern data analysis at end-user level. Therefore, the ultimate value of the proposed conceptual and relational model extensions is determined by the easiness of incorporating those extensions into visual OLAP tools. In this subsection we sketch a prototypical implementation of an end-user interface for multidimensional business process analysis.

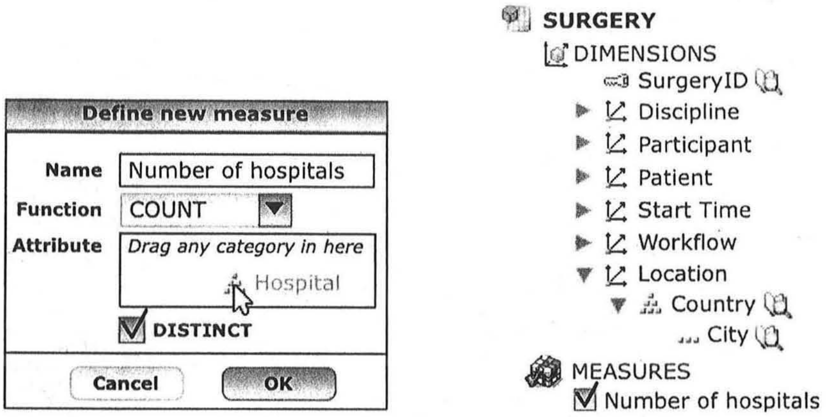
Analysts interact with data in a predominantly "drill-down" fashion, i.e., gradually descending from coarsely grained overviews towards the desired level of detail. Queries are specified interactively via a navigation hierarchy, as the one depicted in Figure 242(b): a cube (i.e., fact table) is a navigation object that can be expanded to access its dimensions and measures. Complex dimensions are represented as hierarchical nodes that can be expanded to access their aggregation levels (child levels are nested in their parent levels). Compulsory elements of any analytical query are 1) a measure specified as an aggregation function (e.g., sum, average, maximum etc.) and its input attribute and 2) a set of dimension categories defining the granularity of the aggregation. In addition to the pre-configured measures, the navigation hierarchy supports derivation of user-defined measures from any attribute of the scheme at query time.

New measures are defined through a wizard, as depicted in Figure 24(a), by providing the following input:

1. The aggregation function is selected from the drop-down menu;
2. The attribute of the measure is dragged from the navigation into the wizard.
3. The DISTINCT option allows activating duplicate elimination.
4. The newly defined measure may be supplied with a user-friendly name.

Each new measure has to be defined just once and remains available for further analysis. Let us consider an example of analyzing the distribution of hospitals by discipline. Intuitively, the measure of interest is the number of hospitals that has to be created from the category Hospital of the dimension Location. Figure 24(a) shows the process of creating this measure by dragging Hospital node into the wizard. Obviously, to support this measure, fact entries in SURGERY need to be aggregated to the Hospital level, the category Hospital and

Figure 24. Example of interactively defining a new measure (i.e., invoking PUSH operator) © 2009 University of Konstanz. Used with permission.



Define new measure

Name: Number of hospitals

Function: COUNT

Attribute: Drag any category in here — Hospital

☑ DISTINCT

Cancel | OK

SURGERY
DIMENSIONS
SurgeryID
▶ Discipline
▶ Participant
▶ Patient
▶ Start Time
▶ Workflow
▼ Location
▼ Country
City
MEASURES
☑ Number of hospitals

(a) Defining a measure

(b) Data cube navigation

all categories below it, i.e., Room and Building, must be removed from the navigation tree of the dimension Location, as they are invalid in the context of the specified measure. The navigation fragment of SURGERY containing a new measure is shown in Figure 24(b).

SAMPLE USAGE SCENARIOS

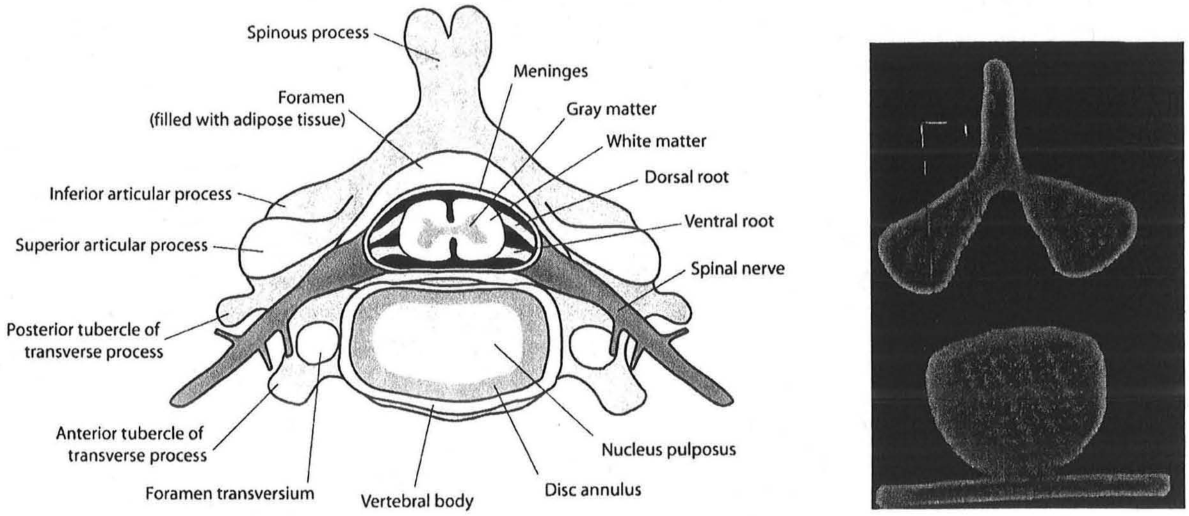
We demonstrate the use of the proposed analysis framework by considering an application case from the area of instrument usage analysis in surgical interventions of type discectomy, which is an intervention at the spine. The intervention goal of a discectomy is the partial removal of the herniated intervertebral 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.

During a discectomy, parts of the vertebra are removed to assess the underlying intervertebral disc. Figure 25 should give the reader some insight into the affected anatomic structure. The main elements of vertebra are depicted in Figure 25(a), adopted from Wikimedia Commons (2007), and Figure 25(b) shows a computer-tomographic image of a rapid prototyping model of the human spine (cross-section). The intervertebral disc is hidden from surgical access in the center angle under the bone material (white segments in Figure 25(b)). The red-marked area represents the volume of the vertebra to be removed by the surgeon to gain access to the intervertebral disc in order to remove it.

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 can reach, and then decides whether further access is needed. If so, he ablates the next portion of the vertebra and removes the tissue again, and so on.

The conventional bone ablation at the vertebra is performed using different surgical instruments, such as surgical punch, trephine, and/or surgical mallet/chisel. Each of the instrument types are available in different sizes and has different properties regarding invasiveness or handedness.

Figure 25. Human spine as the treated structure of a discectomy



(a) Annotated diagram of vertebra

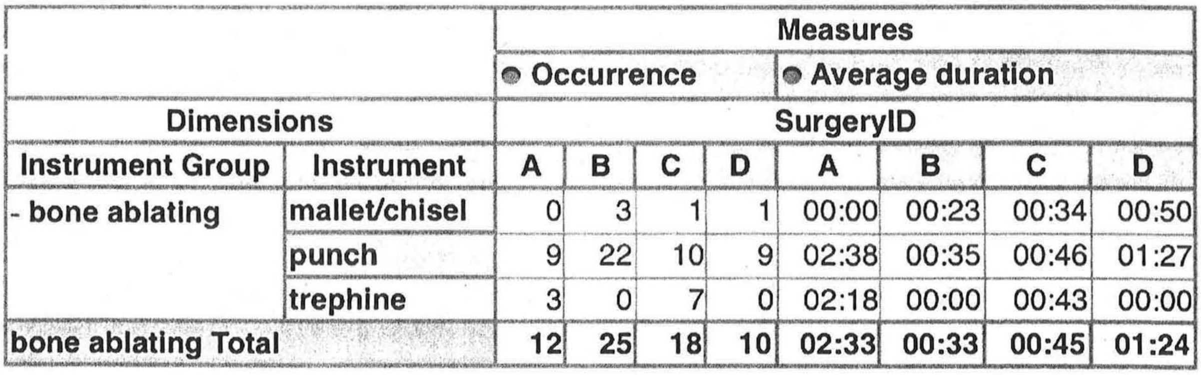
(b) Spine cross-section view

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, end-users can obtain the required aggregates in few simple interaction steps. Figure 26 contains the results of the first two of the following four queries, arranged into a pivot table.**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 the respective instrument was used by the surgeon.*

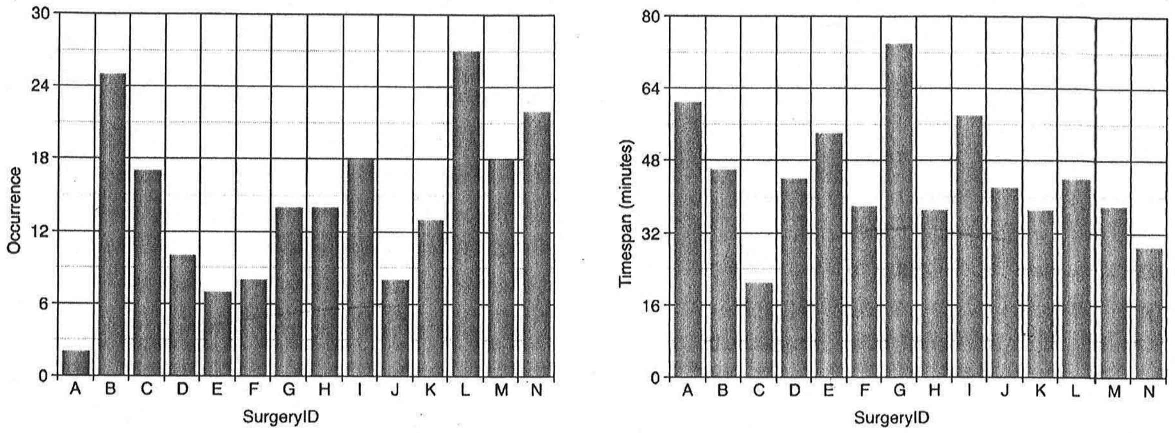
The query is answered by specifying a new measure Occurrence, defined as COUNT(*), i.e., simple counting of qualifying fact entries, in fact table STEP. The aggregates are then computed by a roll-up of Occurrence by Surgery and Instrument with selection conditions on Instrument Type ('bone ablating') and on Participant ('surgeon').**Query II.** *For each intervention of type discectomy and each of the specified bone*

Figure 26. Instrument usage statistics as a pivot table



		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 27. ***Occurrence and duration of bone ablation work steps in discectomy interventions as bar-charts***



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

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

ablating instruments, return the average duration of a work step, in which the respective instrument was used by the surgeon.

The query is answered by specifying a new measure Average Duration, defined as AVG(Duration), in fact table STEP and performing the same roll-up as in Query I.**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 rollup step (by removing Instrument from the GROUP BY clause). The results of the query, arranged into a bar-chart, are shown in Figure 27(a).**Query IV.** *For each intervention of type discectomy, calculate the total time span between the begining of the first and the end of the last 'bone ablating' activity.*

The query is answered by specifying a new measure Timespan, defined as MAX(StopTime) - MIN(StartTime), 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 27(b).

The above queries describe a real-world example from the field of medical engineering. The aggregates obtained in the above queries reveal the usage pattern for bone ablating instruments and provide crucial information for predicting the success of a new surgical instrument in this field (Neumuth, et al., 2007). 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 (Jank, et al., 2006). This 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.

CONCLUSION

Motivated by the growing research interest to the evolving area of business process intelligence, we proposed a conceptual framework for providing OLAP support to business process analysis. Conventional BPMS are rather limited in the types of supported analytical tasks, whereas the data warehousing techniques appear more suitable when it comes to managing large amounts of data, defining various business metrics and running complex queries. As a challenging real-world application, we chose a case study from the innovative and promising domain of Surgical Workflows Analysis, aimed at designing a recording scheme for acquiring process descriptions from surgical interventions for their subsequent analysis and exploration.

We demonstrated the deficiencies of the standard relational OLAP approach with respect to the requirements of our case study and proposed an extended multidimensional data model that addresses multiple challenges, such as non-quantitative and heterogeneous facts, many-to-many relationships between facts and dimensions, full and partial dimension sharing, dynamic specification of new measures, and interchangeability of fact and dimension roles. We also presented a prototypical implementation of the enhanced conceptual model in a relational OLAP system where the data is stored according to the fact constellation schema and can be queried with standard SQL. The work is concluded by presenting a relevant analytical task from the domain of the case study and its sample solution, obtained interactively using an advanced visual OLAP frontend tool that supports dynamic measure definition.

ACKNOWLEDGMENT

The authors would like to thank Christos Trantakis and Jürgen Meixensberger from the Neurosurgery Department at the University Hospital of Leipzig for the valuable domain expertise.

REFERENCES

Aalst, W. M. P. van der, & Hee, K., van. (2002). *Workflow management: Models, methods, and systems (cooperative information systems)*. Cambridge, MA: The MIT Press.

Abelló, A., Samos, J., & Saltor, F. (2001). Understanding facts in a multidimensional object-oriented model. In *Proceedings of the 4th ACM international workshop on Data Warehousing and OLAP* (pp. 32-39). New York: ACM Press.

Ahmadi, S.-A., Sielhorst, T., Stauder, R., Horn, M., Feussner, H., & Navab, N. (2006). Recovery of surgical workflow without explicit models. In *Proceedings of the Medical Image Computing and Computer-Assisted Intervention (MICCAI 2006)* (pp. 420-428). Berlin, Germany: Springer.

BPMN. (2006, February). *BPMN (Business Process Modeling Notation) 1.0: OMG Final Adopted Specification*. Retreived from http://www.bpmn.org

Burgert, O., Neumuth, T., Gessat, M., Jacobs, S., & Lemke, H. U. (2007). Deriving DICOM surgical extensions from surgical workflows. *Medical Imaging 2007: PACS and Imaging Informatics, 8*(35), 651604.1-651604.11.

Burgert, O., Neumuth, T., Lempp, F., Mudunuri, R., Meixensberger, J., & Strauß, G. (2006). Linking top-level ontologies and surgical workflows. *International Journal of Computer Assisted Radiology and Surgery, 1*(1), 437–438. doi:10.1007/s11548-006-0032-x

Cabibbo, L., & Torlone, R. (1998). A logical approach to multidimensional databases. In *EDBT'98: Proceedings of the 6th International Conference on Extending Database Technology* (Vol. 1377, pp. 183-197). Berlin, Germany: Springer.

Castellanos, M., & Casati, F. (2005). Is there anything new about business process intelligence? [panel]. In *ICDE 2005: Proceedings of the 21st International Conference on Data Engineering* (pp. 1141-1141). Los Alamitos, CA: IEEE Computer Society.

Codd, E. F., Codd, S. B., & Salley, C. T. (1993). *Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate* (Tech. Rep.). E. F. Codd & Associates.

Dayal, U., Hsu, M., & Ladin, R. (2001). Business process coordination: State of the art, trends, and open issues. In *VLDB 2001: Proceedings of the 27th International Conference on Very Large Data Bases* (pp. 3-13). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Dodds, D., Hasan, H., & Gould, E. (1999). Relational versus multidimensional databases as a foundation for online analytical processing. In *IRIS 22: Proceedings of the 22nd Information Systems Research Seminar in Scandinavia* (pp. 281-288).

Franconi, E., & Sattler, U. (1999). A data warehouse conceptual data model for multidimensional aggregation. In *DMDW'99: Proceedings of the International Workshop on Design and Management of Data Warehouses* (Vol. 19, pp. 13.1-13.10). Retrieved from http://www.CEUR-WS.org

Giovinazzo, W. A. (2000). *Object-oriented data warehouse design: Building a star schema.* Upper Saddle River, NJ: Prentice Hall PTR.

Golfarelli, M., Maio, D., & Rizzi, S. (1998). The dimensional fact model: A conceptual model for data warehouses. *International Journal of Cooperative Information Systems, 7*(2/3), 215–247. doi:10.1142/S0218843098000118

Golfarelli, M., & Rizzi, S. (1998). A methodological framework for data warehouse design. In *DOLAP '98: Proceedings of the 1st ACM International Workshop on Data Warehousing and OLAP* (pp. 3-9). New York: ACM Press.

Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., & Shan, M.-C. (2004). Business process intelligence. *Computers in Industry, Special Issue on Process/Workflow Mining, 53*(3), 321-343.

Hall, C. (2004, June). Business process intelligence. *BPTrends Newsletter*. Retrieved from http://www.bptrends.com/publicationfiles/06%2D04%20NL%20BPI%20-%20Hall%20PH-2.pdf

Hao, M. C., Keim, D. A., Dayal, U., & Schneidewind, J. (2006). Business process impact visualization and anomaly detection. *Information Visualization, 5*, 15–27. doi:10.1057/palgrave.ivs.9500115

Hurtado, C. A., & Mendelzon, A. O. (2002). OLAP dimension constraints. In *PODS 2002: Proceedings of the 21st ACM Symposium on Principles of Database Systems* (pp. 169-179).

Hüsemann, B., Lechtenbörger, J., & Vossen, G. (2000). Conceptual data warehouse design. In *DMDW'2000: Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses at CAiSE'00* (Vol. 28, pp. 6.1-6.11). Retrieved from http://www.CEUR-WS.org

Jank, E., Rose, A., Huth, S., Trantakis, C., Korb, W., & Strauss, G. (2006). A new fluoroscopy based navigation system for milling procedures in spine surgery. *International Journal of Computer Assisted Radiology and Surgery, 1*(Supplement 1), 196–198.

Jannin, P., Raimbault, M., Morandi, X., Riffaud, L., & Gibaud, B. (2003). Model of surgical procedures for multimodal image-guided neurosurgery. *Computer Aided Surgery, 8*(2), 98–106. doi:10.3109/10929080309146044

Kimball, R. (1996). *The data warehouse toolkit: Practical techniques for building dimensional data warehouses*. New York: John Wiley & Sons, Inc.

Kimball, R., Reeves, L., Ross, M., & Thornwaite, W. (1998). *The data warehouse lifecycle toolkit: Expert methods for designing, developing and deploying data warehouses*. New York: John Wiley & Sons, Inc.

Kimball, R., & Ross, M. (2002). *The data warehouse toolkit: The complete guide to dimensional modeling*. New York: John Wiley & Sons, Inc.

Lechtenbörger, J. (2001). Data warehouse schema design (Doctoral dissertation, Westfälische Wilhelms-Universität Münster). *Dissertations in database and information systems, 79.*

Lenz, H.-J., & Shoshani, A. (1997). Summarizability in OLAP and statistical data bases. In *SSDBM 1997: Proceedings of the 9th International Conference on Scientific and Statistical Database Management* (pp. 132-143).

Malinowski, E., & Zimányi, E. (2006). Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data & Knowledge Engineering, 59*(2), 348–377. doi:10.1016/j.datak.2005.08.003

Mansmann, S., Neumuth, T., & Scholl, M. H. (2007a). Multidimensional data modeling for business process analysis. In *ER 2007: Proceedings of the 26th International Conference on Conceptual Modeling* (pp. 23-38). Berlin, Germany: Springer.

Mansmann, S., Neumuth, T., & Scholl, M. H. (2007b). OLAP technology for business process intelligence: Challenges and solutions. In *DaWaK'07: Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery* (pp. 111-122). Berlin, Germany: Springer.

Mansmann, S., & Scholl, M. H. (2007). Empowering the OLAP technology to support complex dimension hierarchies. *International Journal of Data Warehousing and Mining, 3*(4), 31–50.

Mansmann, S., & Scholl, M. H. (2008). Extending the multidimensional data model to handle complex data. *Journal of Computer Science and Engineering, 1*(2).

Matoušek, P. (2003). Verification of business process models (Doctoral dissertation, VŠB - Technická Univerzita Ostrava). In *Kolekce vysokoškolských kvalifikačních prací zpracovaných do konce 1. pololetí 2006.* VŠB-TUO.

Münchenberg, J., Brief, J., Raczkowsky, J., Wörn, H., Hassfeld, S., & Mühling, J. (2000). Operation planning of robot supported surgical interventions. In *IROS 2000* []. Washington, DC: IEEE.]. *Proceedings of Intelligent Robots and Systems, 1*, 547–552.

Muth, P., Wodtke, D., Weißenfels, J., Weikum, G., & Kotz-Dittrich, A. (1998). Enterprise-wide workflow management based on state and activity charts. In *Workflow management systems and interoperability* (Vol. 164, pp. 281-303). Berlin, Germany: Springer.

Neumuth, T., Schumann, S., Strauß, G., Jannin, P., Meixensberger, J., & Dietz, A. (2006). Visualization options for surgical workflows. *International Journal of Computer Assisted Radiology and Surgery, 1*(1), 438–440.

Neumuth, T., Strauß, G., Meixensberger, J., Lemke, H. U., & Burgert, O. (2006). Acquisition of process descriptions from surgical interventions. In *DEXA 2006: Proceedings of the 17th International Conference on Database and Expert Systems Applications* (pp. 602-611). Berlin, Germany: Springer.

Neumuth, T., Trantakis, C., Eckhardt, F., Dengl, M., Meixensberger, J., & Burgert, O. (2007). 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, 2*(Supplement 1), 436–438.

Padoy, N., Horn, M., Feußner, H., Berger, M.-O., & Navab, N. (2007). Recovery of surgical workflow: A model-based approach. In *CARS 2007: Proceedings of the 21st International Congress and Exhibition on Computer Assisted Radiology and Surgery*. Berlin, Germany: Springer.

Pedersen, T. B., & Jensen, C. S. (2001). Multidimensional database technology. *IEEE Computer, 34*(12), 40–46.

Pedersen, T. B., Jensen, C. S., & Dyreson, C. E. (2001). A foundation for capturing and querying complex multidimensional data. *Information Systems, 26*(5), 383–423. doi:10.1016/S0306-4379(01)00023-0

Phipps, C., & Davis, K. C. (2002). Automating data warehouse conceptual schema design and evaluation. In *DMDW'2002: Proceedings of the 4th International Workshop on Design and Management of Data Warehouses* (Vol. 58, pp. 23-32). Retrieved from http://www.CEUR-WS.org

Qi, J., Jiang, Z., Zhang, G., Miao, R., & Su, Q. (2006). A surgical management information system driven by workflow. In *SOLI '06: Proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics* (pp. 1014-1018). Washington, DC: IEEE.

Reijers, H. A. (2003). *Design and control of workflow processes: Business process management for the service industry* (Vol. 2617). Berlin, Germany: Springer.

Sapia, C., Blaschka, M., Höfling, G., & Dinter, B. (1999). Extending the E/R model for the multidimensional paradigm. In *ER 1998: Proceedings of the Workshops on Data Warehousing and Data Mining* (pp. 105-116).

Sharp, A., & McDermott, P. (2001). *Workflow modeling: Tools for process improvement and application development*. Norwood, MA: Artech House, Inc.

Smith, M. (2002, December, 5). Business process intelligence. *Intelligent Enterprise*. Retrieved from http://www.intelligententerprise.com/021205/601feat2 1.jhtml

Song, I.-Y., Rowen, W., Medsker, C., & Ewen, E. F. (2001). 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* (Vol. 39, pp. 6.1-6.13). Retrieved from http://www.CEUR-WS.org

SRS. (n.d.). *SRS glossary: Definitions of scoliosis terms*. Retrieved from from http://www.srs.org/patients/glossary.asp

Tryfona, N., Busborg, F., & Christiansen, J. G. B. (1999). starER: A conceptual model for data warehouse design. In *DOLAP '99: Proceedings of the 2nd ACM International Workshop on Data Warehousing and OLAP* (pp. 3-8). New York: ACM Press.

WfMC. (1999, February). *Terminology & glossary*. Retrieved from http://www.wfmc.org/standards/docs/TC-1011 term glossary v3.pdf

Wikimedia Commons. (2007). *Category: Vertebra*. Retrieved from http://commons.wikimedia.org/wiki/Category:Vertebra

APPENDIX A.

Figure 28. Graphical node type constructs of X-DFM

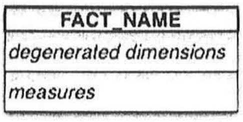
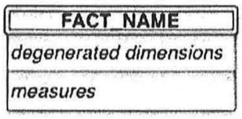
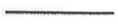
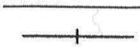
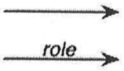
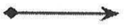
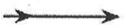
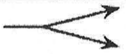
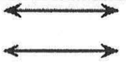
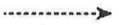
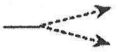
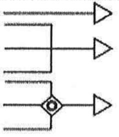
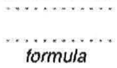
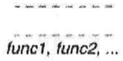
Element	Description
	A **fact** is a box-shaped node labeled by the fact name and containing two sets of elements: 1) **degenerated dimensions** and 2) **measures**. Both sets are allowed to be empty.
	A **degenerated fact** is a many-to-many fact-dimensional mapping extracted into a separate fact, shown 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).
○ attribute_name	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.
attribute_name	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.
attribute_name	A **"degree-of-belonging" attribute** is a property associated with a child category of a non-strict weighted roll-up relationship.

Figure 29. Graphical node 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.