On Relationships Offering New Drill-across Possibilities

Alberto Abelló
U. Politècnica de Catalunya
C/ Manuel Girona 1-3
E-08034 Barcelona
aabello@lsi.upc.es

José Samos
U. de Granada
Avd. de Andalucia 38
E-18071 Granada
jsamos@ugr.es

Fèlix Saltor
U. Politècnica de Catalunya
C/ Manuel Girona 1-3
E-08034 Barcelona
saltor@lsi.upc.es

ABSTRACT

OLAP tools divide concepts based on whether they are used as analysis dimensions, or are the fact subject of analysis, which gives rise to star shape schemas. Operations are always provided to navigate inside such star schemas. However, the navigation among different stars is usually overlooked. This paper studies different kinds of Object-Oriented conceptual relationships (part of UML standard) between stars (namely *Derivation*, *Generalization*, *Association*, and *Flow*) that allow to drill across them.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications

General Terms

Design

Keywords

Multidimensional design, UML, drill-across, semantics

1. INTRODUCTION

A Data Warehouse (DW) is, as it was defined in [8], a set of subject-oriented, integrated, historical, non-volatile data in support of the decision making process. Due to its historical and global nature, it contains lots of data that analysts find difficult to query. On-Line Analytical Processing (OLAP) tools are conceived to ease the navigation through the DW. The most important characteristic of OLAP is the multidimensional view it offers. This allows one to apply specific storage techniques in order to reduce response times. Thus, small DWs optimized for multidimensional access, i.e. Data Marts (DMs), are deployed. In this work we will forget storage and retrieval mechanisms to center our efforts in how conceptual multidimensional modeling helps users.

Multidimensionality conceives data as if they were placed in a hypercube (we will adopt the term *Cube* from here on)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DOLAP'02, November 4–9, 2002, McLean, Virginia, USA.
Copyright 2002 ACM 1-58113-492-4/02/0011 ...$5.00.

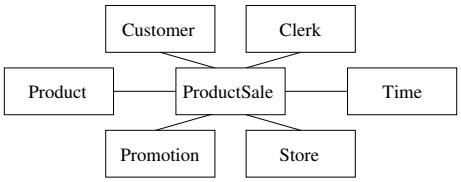

Figure 1: Star schema of product sales

where cells contain the data subject of analysis, and the different analysis dimensions show points of view to see data from. These data are conceptualized as instances of a star shape schema like that shown in figure 1. The fact of interest is surrounded by its analysis dimensions, so that a point in each one of them identifies a specific fact. We are interested in the analysis of the fact `ProductSale` with regard to the `Product` which was sold, the `Time` when was sold, the `Customer` to whom was sold, the `Clerk` who sold it, the `Promotion` that affects it, and the `Store` where it was sold.

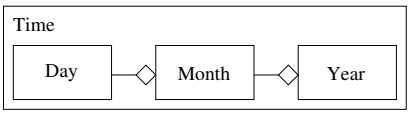

Figure 2: Aggregation hierarchy in Time dimension

How different analysis data could be defined at different granularities was already pointed out by [18], a statistical model close to multidimensional concepts. Thus, some multidimensional models, like [12], [3], [6], [17], [20], or [15], decompose analysis dimensions into hierarchies of aggregation levels. In these models, *roll-up* or *aggregation hierarchies* might be defined, which show the different *granularities* or *aggregation levels* at which data can be found in each analysis dimension. For instance, if we consider the `Time` dimension in figure 2, it contains `Day`, `Month`, and `Year` aggregation levels, so that data available by day can be rolled up to month or year.

Multidimensional operations have also been defined over *Cubes*: *Slice* reduces the dimensionality of a *Cube*; *Dice* selects a set of data; *Roll-up* aggregates data along the hierarchy in an analysis dimension; *Drill-down* gives more detail in an analysis dimension, by descending along its aggregation hierarchy; and *Drill-across* travels from a *Cube* to another one. An in depth explanation of these operations is found

in [5]. SQL syntax was already extended to support some multidimensional operations as can be seen in [9]. In this paper, we leave aside most of these operations to pay special attention to *Drill-across*.

Quite often, different data cubes in a business model are found closely related, and analysts want to jump from one to another. They are usually interested in generating several reports showing different sets of data organized from the same point of view, so that they are easily comparable. For instance, it could be interesting to analyze evolution of sales along `Time` and `Product`, and compare it with production in the same period of time, for the same product. Probably, that information will be stored in different *Cube*s, and users will need to drill across multidimensional schemas. The aim of this paper is to elaborate on the applicability of *Drill-across* by studying how a multidimensional schema could contain several, related star shape subschemas, even if the data cubes are physically stored in different DMs.

The structure of the paper is as follows: section 2 presents some work on relating different star schemas; section 3 shows our general approach to the problem; section 4 exemplifies the different kinds of relationships we found between *Star*s; section 5 contains a discussion about the relationships we found; finally, section 6 draws conclusions.

2. RELATED WORK

Lately, there has been a lot of work about OLAP tools. We can find literature devoted to specific storage techniques and access mechanisms, as well as to pure multidimensional modeling. Both areas benefit from the duality fact-dimension, and restrict their studies to isolated star shape schemas, i.e. how we can store/model one fact and its surrounding analysis dimensions. Nevertheless, some authors have already pointed out the importance of drilling across different data cubes, which means navigating through data in different star shape schemas. Unfortunately, the "Drill-across" operation, in these models, is limited to the case that some analysis dimensions are shared by the *Cube*s.

[10] proposes a logical model to implement "star schemas" on Relational databases. Each "star schema" contains a central "fact table" related by foreign keys to its corresponding "dimension tables". To support "Drill-across", Ralph Kimball contends that all constraints on dimension attributes must evaluate to exactly the same set of dimension instances in both schemas. This is clearly satisfied, if the respective *Dimension*s are exactly the same. However, he also explains how this matching can be satisfied, for instance, if the only difference between the *Dimension*s is their granularity, i.e. the finest detail level allowed.

A later work, [7], presents "multi-star" schemas obtained by normalization of "fact tables", while [16] defines a "multidimensional object family" as multidimensional objects possibly with shared subdimensions (i.e. subsets of levels in the aggregation hierarchy of the analysis dimension). [13] goes a little further, and distinguishes three kinds of schemas with more than one *Star*, i.e. "constellation", "galaxy", and "star cluster". A "constellation" schema consists of a set of star shape schemas with hierarchically linked "fact tables". A "galaxy" is a collection of star schemas with shared *Dimension*s. Finally, a "star cluster" is a set of star schemas sharing subdimensions. [5] defines "constellation", as well. In this case, it is a set of star shape schemas sharing analysis dimensions. The shared analysis dimensions must be "conforming dimensions", which means for this author that their values are consistent among the star schemas. Even though these models allow several star shape subschemas in a schema, they do not study relationships between them.

Some authors, like [20] and [19], show the usage of semantic relationships in multidimensional modeling. Nevertheless, none of them studies their usage on relating several star schemas. Specifically, *Flow* and *Derivation* have not even been considered.

3. MULTISTAR CONCEPTUAL SCHEMAS

Along this paper, we use a multidimensional extension of UML as presented in [2]. There, multidimensional elements are defined as follows:

- A *Fact* (formally defined in [1]) is a subject of analysis. It could contain different kinds of cells (we will call them *Cell*s), which, in turn, could contain different *Measure*s that we want to analyze. Each data cell is identified by a point in each of its analysis dimensions. These points may correspond to different granularities for every cell. For instance, in the introductory example of figure 1, `ProductSale` would contain different *Cell*s, if some *Measure*s were not available at `Day` granularity. Moreover, other *Measure*s could not be interesting for every `Customer`, but only for `CustomerProfiles`. Thus, there would be different kinds of cells depending on whether *Measure*s are available or meaningful for either `Day` or `Month`, `Customer` or `CustomerProfile`.

- A *Dimension* is a connected, directed graph representing a point of view on analyzing data. Every vertex in the graph corresponds to an aggregation *Level*, and an edge reflects that every instance at target *Level* decomposes into a collection of instances of source *Level* (i.e. edges reflect part-whole relationships between instances of *Level*s). Each *Level* corresponds to a granularity in the *Dimension*, and has attributes that allow us to select some of its instances. By selecting points in every analysis dimension, we choose the data cells of interest in our analysis. Thus, *Dimension*s contain those data that identify *Cell*s instances.

- A *Star* is a *Fact* and the set of *Dimensions* used in its analysis.

- A *Cube* is the representation of a set of *Cell* instances in the n-dimensional space defined by one *Level* for each *Dimension*.

Users could only be interested in a given *Fact*. However, sometimes, they will desire to relate data in different *Star*s, and OLAP tools should also allow it. We can understand *Drill-across* as re-using the same condition over the *Dimension*s on querying different *Fact*s. This means that we select a subspace in a given *Cube*, and want to view the corresponding space for a different *Fact*. At a first glance, this can be allowed, if both *Cube*s share some *Dimension*s. However, we argue that it is also possible, if *Dimension*s are not exactly the same. If this is the case, semantic relationships should exist between the *Dimension*s and/or *Fact*s in the *Star*s.

Actually, the analysis dimensions used on drilling across need not exactly coincide in both *Cube*s. We contend that

they should just be defined on a related semantic domain (i.e. domains reflecting the conceptualization of values in the mind of the designer). Thus, *Drill-across* between different *Cube*s is performed thanks to semantic relationships between *Star*s. Two *Star*s are related whether either their *Fact*s, or their *Dimension*s are related. We will distinguish four kinds of relationships in this paper, i.e. *Derivation*, *Generalization*, *Association*, and *Flow* (in UML terminology, as defined in [14]). Maybe the less known of this *Relationship*s is *Flow*, which relates two versions of an object, or an object and a copy of it.

These *Relationships* offer new analysis possibilities to the users. They free them from being chained to one subject of analysis, allowing to navigate between *Star*s, not only when there exist shared *Dimension*s, but also in other cases.

4. INTER-STELLAR RELATIONSHIPS

We are interested in providing multidimensional schemas with more than one *Fact*. However, having several *Fact*s in the same schema is absolutely useless, if it contains isolated *Star*s. *Fact*s need to be related in some way to allow *Drill-across*. Some models, and most OLAP tools allow it, if *Star*s share *Dimension*s. However, purely sharing *Dimension*s does not seem enough at conceptual level, where we could find much more meaningful semantic relationships among data.

In this section, we are going to detail the semantic relationships between different *Star*s, that allow navigation through them. Section 4.1 shows possible relationships between two *Dimension*s; section 4.2 exemplifies relationships between two *Fact*s; and section 4.3 explains how a *Fact* can be related to a *Dimension* in another *Star*, or vice versa.

4.1 Dimension-Dimension

[11], [5], and some multidimensional tools, stress the importance of having "conforming dimensions" to drill across. Thus, the schemas must share analysis dimensions so that their instances exactly coincide. This unnecessarily restricts the usage of *Drill-across*. Actually, it is just needed that the selected instances of the *Dimension*s of the origin *Star* determine instances in the *Dimension*s of the destination *Star*. Thus, domains used in both *Dimension*s should be related in some way, but *Dimension*s could still be different. This section shows four kinds of O-O relationships between *Dimension*s that allow *Drill-across*.

4.1.1 Derivation

Firstly, we could find that the same concept has different names depending on the subject. Therefore, the same *Dimension*, with exactly the same instances, will need a different name depending on the context where we are going to use it. Moreover, this *Dimension* could not play the same role for different *Fact*s. `Product`, may be considered `RawMaterial` in a different context. It is not enough to say they are "synonyms" (like in [10]), because they could even have different attributes of interest to the users. For example, keeping or studying the `benefit` of raw material can be meaningless. Moreover, we could find that elements in a *Dimension* are different from those in another one, even though they represent the same concepts. For instance, a given subject implies that `Red`, `Blue`, and `Yellow` are the instances in `Color` domain; while in a different case, we need to distinguish different kinds of `Blue`, like `Dark Blue` or `Light Blue`. It is also possible to find differences in how concepts are codified (for example, letters or numbers).

Sometimes, some *Dimension* instances in a *Star* are only considered grouped in another *Star* (only summarized data at some level are considered), because of lack of interest in the individuals, confidentiality issues, or space problems to keep information at maximum detail. In [10], a *Dimension* whose finer granularity is not of interest is called "Demographic minidimension". Thus, we could use the same *Dimension* in two star schemas at different aggregation levels. For instance, one of them could keep data by hour, while the other does it by day. Clearly, all we have to do to drill across them is roll data at `Hour` up to `Day`. In this way, both *Cell*s will be defined over the same kind of dimensional instances. Hence, they will be comparable.

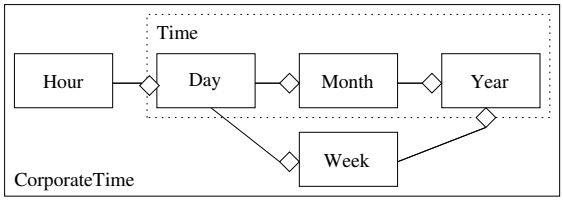

Figure 3: Example of containment of *Dimension*

If two *Dimension*s coincide in one of their aggregation levels, both can be conceived again as derived from a common, more general *Dimension* which contains their hierarchies. Figure 3 shows how the `Time` *Dimension*, previously presented in figure 2, is included in a more general *Dimension* `CorporateTime`. `Time` does neither provide `Hour` nor `Week` *Level*s, because they are not of interest for the `ProductSales` *Star* where it is used.

Considering *Derivability* would allow one to reflect that two *Dimension*s come from a common concept, in spite of the fact that they look different. Thus, two *Dimension*s can be related by *Derivation*, and users can drill across from a *Fact* to another one through it. The *Dimension* is not shared, because it appears different in each *Star*. However, by means of this kind of relationship, we can offer users the desired view while they are still able to drill across.

4.1.2 Generalization

We can also find relationships between analysis dimensions along *Generalization/Specialization*. *Dimension*s of different *Star*s could be related by *Generalization*, so that *Drill-across* would be allowed. For instance, `Customer` and `Clerk` are both specialization of `People`. Therefore, we could travel from a *Star* with `Customer` *Dimension* to another one with `Clerk` *Dimension*, if their sets of instances are not disjoint. Moreover, they will have in common all attributes in the superclass, and maybe some aggregation levels.

As outlined in [5], using superclasses and subclasses in star schemas, we gain better understanding. Figure 4 exemplifies how we can specialize `People` *Dimension* at `SaleRole` *Level* (solid arrow) to get `Clerk` *Dimension*, which contains a *Level* (i.e. `Clerk`) with instances corresponding to people acting as clerks, and another one with only one instance representing the set of all clerks (i.e. `All`). Dashed arrows show that a *Level* is specialization of another one. By a similar specialization of `People`, we obtain `Customer` *Dimension*.

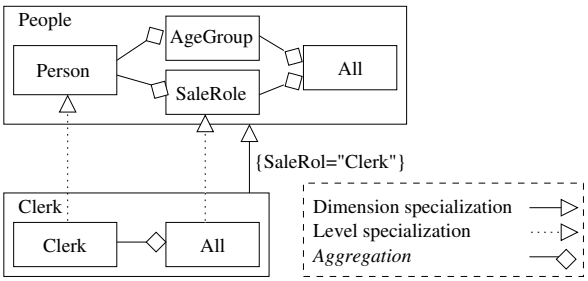

Figure 4: Example of *Dimension Generalization*

Sharing *Dimension*s would not be enough in this case, neither. `Clerk` has more attributes and much less instances than `Customer`. Therefore, sharing `People` *Dimension* in both *Star*s would generate lots of undesirable null values. Nevertheless, we can still drill across, if instances of `Clerk` can also be instances of `Customer`.

4.1.3 Association

It is possible to have associated analysis dimensions, as well. The domain of a *Dimension* could be used as an attribute domain in another *Dimension*. Selected instances in a *Dimension* would allow to identify instances in the other, so that it is possible to drill across the corresponding *Fact*s. Clerks used to be assigned to stores. Thus, `Clerk` would be associated with `Store` *Dimension* (maybe multivalued). This is not what is called "outrigger table" in [10]. That is at a *Logical* level, and refers to normalization. In this case, it is not normalizing at all, but showing that two different analysis dimensions are semantically related.

We can also find stronger associations between analysis dimensions, if we join more than one to give rise to another. This is not a simple association, because if we remove one of the aggregated *Dimension*s, we loose the aggregate one. For example, `Colors` could be used to define `ColoredProduct`. Dissociating `Colors` from `ColoredProduct` means we do not have colored products any more.

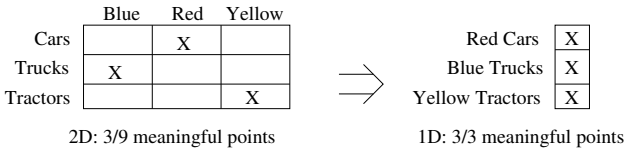

Figure 5: Example of correlated *Dimension*s

If `ColoredProduct` would be represented as two separate *Dimension*s (i.e. `Color` and `Product`), all combinations of color-product would be allowed. Sometimes this could be the case, but other times, products are only available for a reduced set of colors (if both analysis dimensions are correlated, as exemplified in figure 5). Therefore, it is much better for the designer to reduce the analysis space only to those meaningful values by modeling all of them in just one *Dimension*. In the figure, we have six meaningless values out of nine possibilities. Thus, it should be better to model it as only one *Dimension* where all three values are meaningful.

Whether a case like this is modeled as two independent *Dimension*s (i.e. `Color` and `Product`), or only one (i.e. `ColoredProduct`), will depend on the distribution of the cells related to the *Dimension* instances, and the user point of view. Thus, we can find `Color` and `Product` in a *Star*, and `ColoredProduct` in another one, and a user should be able to navigate from one to another through *Aggregation* relationships (as shown in figure 6). It is important to notice that the elements at homonym levels in different *Dimension*s, in this case, do not coincide. For instance, `Color` *Level* in `Colors` *Dimension* contains elements representing colors. Nevertheless, `Color` level in `ColoredProduct` *Dimension* contains elements representing sets of colored products grouped by color.

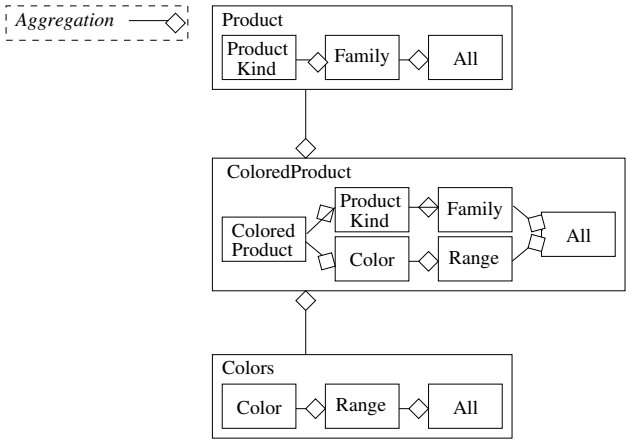

Figure 6: Example of *Dimension Aggregation*

Again, this offers navigation possibilities that just sharing *Dimension*s does not. If a psychological study included `Colors`, `ColoredProduct` could not be shared with the *Star* corresponding to that study, because it was about colors, and did not consider products at all. However, analysts will probably be interested in navigating from one *Star* to the other. Thus, navigation should be allowed through the *Association* between `Colors` and `ColoredProduct`.

4.1.4 Flow

Because of the long periods of interest in analysis tasks and how fast business change nowadays, it is expected that analysis dimensions in a multidimensional schema evolve. It is not acceptable to throw away old *Dimension*s. Old data would still be stored following the old schema, while we are currently using a new one.

As our business grows, it could become international, so that a new *Level* `Country` will appear in the `Store` *Dimension*. Attributes could also appear or disappear in any *Dimension*, as the information systems and the enterprise evolve. We should not study data with regard to those attributes, because their values at the time data was collected are unknown. Therefore, old *Dimension*s should be kept as they were, but related to the corresponding new *Dimension*s by *Flow* relationships. This will show to the user which dimensional data can be used at each analysis depending on the period of time in which s/he is interested. As studied in [4], functions can be offered to the users to estimate values of dimension attributes in the periods of time they are unknown. Anyway, the schema should reflect the difference to let users know whether the values are real or estimated.

Sharing is not possible in this case. It would mean studying old data with regard to new attributes or vice versa. In any case, it could generate absolutely wrong results.

4.2 Fact-Fact

Points in the multidimensional space are always identified by its analysis dimensions. However, using those *Dimension*s is not always necessary to select a set of points. Having functions from points in a space to points in another one is another possibility. Therefore, if we identify a set of cells in a *Fact*, we will be able to select the corresponding set in a related *Fact*. This means that we can also use *Relationship*s between *Fact*s to navigate.

4.2.1 Derivation

Firstly, two *Fact*s can be related by *Derivation*, i.e. *Measure*s in one *Fact* are obtained by applying some operation to *Measure*s in other *Fact*s. For instance, on analyzing efficiency of employees, some *Measure*s could be obtained by operating the profits of some products sold (the best sales, sales involving relevant products, etc.). Probably, most of the *Dimension*s will be shared by both *Fact*s (i.e. `EmployeeEfficiency`, and `ProductSale`). However, we could also travel between them due to the fact that data in some cells are obtained by processing other cells. We could navigate from data in `EmployeeEfficiency` to the data in `ProductSales` used in their calculation. This does not correspond to *Drill-down*, because both *Fact*s represent different subjects, and selection of *Cell* instances is not performed by means of aggregation hierarchies.

4.2.2 Association

As between *Dimension*s, we can also find *Association*s between *Fact*s. A *Fact* in a *Star* can be associated with *Fact*s in other *Star*s. For instance, a `Deal` is composed by several individual `ProductSale`. Notice that *Measure*s of `Deal` are not necessarily obtained from those of `ProductSale` (for instance, discount in the deal). Thus, if we are studying a set of sales, it can be interesting to see data corresponding to deals in which they were done. Coincidences or differences in *Dimension*s do not matter. We should be able to travel from a *Star* to another one just because there exists an *Association* between the *Fact*s.

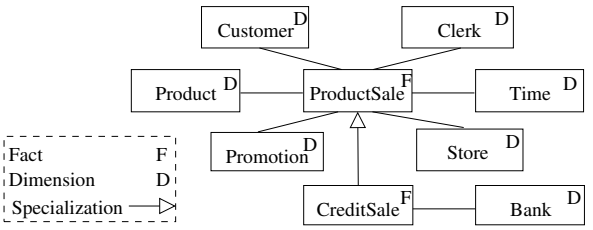

Figure 7: Example of *Generalization* between *Fact*s

4.2.3 Generalization

Some *Fact*s do not have exactly the same *Measure*s nor associated *Dimension*s, but still are closely related. They can be related by *Generalization*. For instance, `ProductSale` can be seen as an specialization of `Contract`. Since a sale is a kind of contract, it will have its specific *Measure*s and *Dimension*s. In turn, as it is shown in figure 7, `ProductSale` could be specialized into `CashSale` or `CreditSale` depending on how it is paid. We will have different information for each of the specializations (for example, number of credit card). Analysis dimensions are inherited from the superclass, but others could be added, like `Bank`. Users should be allowed to navigate through different *Fact*s just because their *Cell*s are specialization of one another. Usually, they will also share most analysis dimensions, but sharing them is not needed in this case to drill across, since *Fact* domains are subset of one another.

4.2.4 Flow

New *Measure*s could appear, possibly replacing others; precision of the measurements could also be improved or even worsened; new interesting analysis dimensions could be found (besides or instead of the already existing ones); and so on. Even if the subject stays, how information is captured can evolve. Data sources, measurement instruments, or calculation algorithms are probably going to change, and these changes should be reflected in our model by means of *Flow* relationships between *Fact*s. All this is not reflected by just relating our *Fact*s to `Time` *Dimension*, since we actually have different *Cell* structures. One day we start recording discount checks in `ProductSale`, hence we need to keep both incomes (i.e. cash, and discount checks). From this day on, we should have different *Star*s containing data about the same kind of facts before and after the acceptance of the checks, because the *Cell* structure changed. An analist would be able to relate those data by means of *Flow* relationships between *Fact*s. At query time, *Flow* relationships would allow to provide conversion functions between different versions of data (like in [4]), or just show a warning to the analysts. However, implementation issues are out of the scope of this paper.

None of all these O-O semantic relationships can be reflected by sharing *Dimension*s. Instead, they show correspondences among factual information, which offers new navigation possibilities. We can go from a *Star* to another, independently of *Dimension*s, if cells in the first determine cells in the other.

4.3 Fact-Dimension

The last possibility to navigate through different *Star*s is that the *Fact* in one of them is used as *Dimension* in the other, or vice versa. This should not be properly regarded as *Drill-across*, since we do not want to analyze data in two *Fact*s from the same point of view. Rather, we are using results of querying a *Fact* to query a different one. For instance, some people could be interested in the analysis of promotions. Thus, the promotions selected by studying `Promotion` *Fact*, can be used as *Dimension* to study `ProductSale`. A *Fact* is not conceived to be used as *Dimension*, so that it will not exactly coincide in both schemas.

4.3.1 Derivation

A *Dimension* can be obtained by deriving it from a *Fact*. The name can be changed, some attributes added or removed, others recalculated, some instances selected, etc. in order to adapt it to its new usage. *Fact*s use to have much more instances than *Dimension*s. Nevertheless, by grouping them, we could obtain coarser aggregation levels of interest. *Measure*s use to be numerical, while attributes in *Dimen-*

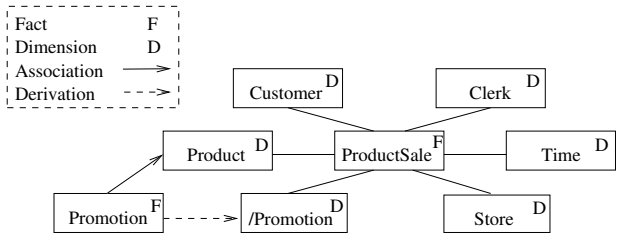

Figure 8: Example of *Fact-Dimension Relationships*

*sion*s use to be descriptive. Therefore, *Derivation* (exemplified in figure 8) will probably imply a change in the kind of attributes. `Promotion` *Fact* could have a numerical attribute `benefits`, while `Promotion` *Dimension* would have an enumerated, derived attribute `success` instead. Thus, users can drill across through the relationship between a *Fact* and a *Dimension* even if they do not coincide, but there exists a *Derivation* between them.

4.3.2 Association

Instances of a *Fact* could also be associated with those of a *Dimension*, or vice versa. For instance, some products can be affected by promotions. Thus, `Product` *Dimension* could have an attribute defined on domain `Promotion` *Fact* (i.e. association arrow in figure 8). Notice the difference between that and relating the `Promotion` to another *Fact* (i.e. deriving a *Dimension* and using it to analyze `ProductSale`). The latter would mean that a sale was performed during a promotion, while the former would show all promotions that have been applied to a kind of product.

5. DISCUSSION

Drill-across implies the usage of the analysis framework that we are using for a given *Fact*, on analyzing a different one. That is to study different data at the same granularity, and constrained by the same conditions over the analysis dimensions. Other authors restrict that to *Cubes* that share *Dimension*s. However, we have seen in previous section, that there are four relationships between *Dimension*s that would also allow it, because *Dimension* instances in one *Star* determine *Dimension* instances in the other. If two *Dimension*s are related by *Generalization*, or *Flow*, they will be different. However, there will be a one-to-one relationship between their instances so that *Drill-across* can be performed. If the *Dimension*s are related by *Derivation*, or *Association*, instances do not coincide, but an instance of a *Dimension* determines instances in the other.

Actually, it is not necessary the *Dimension*s in the destination *Fact* to be related to those in the origin. It could be that selected cells in the latter determine a set of cells in the former. This is the case if both *Fact*s are related in some way. Thus, we just need to substitute *Measures* of one cell, by those of its counterpart in the other *Fact*. In this way, we are also able to study data in two different *Cubes* whose *Fact*s are related.

Relationships between a *Fact* and a *Dimension* do not allow proper *Drill-across*. However, if selected cells during the analysis of a *Fact* determine a set of points in a *Dimension*, these can be used in the analysis of another *Fact*. Notice that neither *Generalization*, nor *Flow* relationships between a *Fact* and a *Dimension* were considered. We think that a temporal transformation can convert neither *Fact*s into *Dimension*s, nor vice versa. Moreover, they represent different kinds of concepts so that they cannot be related by *Generalization*, neither. If we want to obtain one from the other, we must derive it. Nevertheless, factual information cannot be derived from dimensional data. That is because *Fact*s represent measurements, while *Dimension*s show given information.

Relationships	D-D	F-F	F-D	D-F
Derivation	√	√	√	×
Generalization	√	√	×	×
Association	√	√	√	√
Flow	√	√	×	×

Table 1: Summary table

Table 1 sums up the kinds of *Relationship*s between different *Star*s that we found of interest for *drill-across*.

6. CONCLUSIONS

In this paper we have described relationships we found between star schemas for OLAP tools. We explained how different *Star*s can be related by *Derivation*, *Generalization*, *Association*, or even *Flow* relationships (in UML terminology). We have shown how those relationships, between analysis dimensions and the different kinds of cells, could be used to navigate or *Drill-across* between *Star*s, even when *Dimension*s are not shared.

The implementation of the operations will depend on the kind of DBMS we are using. Currently, we are working on the implementation of a UML design tool, which will offer the translation of a multidimensional schema to a relational system. The next step will be to built a graphic interface to query such schema.

Acknowledgements

This work has been partially supported by the Spanish Research Program PRONTIC under projects TIC2000-1723-C02-01, and TIC2000-1723-C02-02.

7. REFERENCES

[1] A. Abelló, J. Samos, and F. Saltor. Understanding Facts in a Multidimensional Object-Oriented Model. In *Proc. of the 4th Int. Workshop on Data Warehousing and OLAP (DOLAP)*, pages 32–39. ACM Press, 2001.

[2] A. Abelló, J. Samos, and F. Saltor. *YAM*^{2} (Yet Another Multidimensional Model): An extension of UML. In *Proc. of the Int. Database Engineering & Applications Symposium (IDEAS'02)*, pages 172–181. IEEE Press, 2002.

[3] L. Cabibbo and R. Torlone. A logical approach to multidimensional databases. In *Advances in Database Technology - EDBT'98*, number 1377 in LNCS, pages 183–197. Springer, 1998.

[4] J. Eder and C. Koncilia. Changes of Dimension Data in Temporal Data Warehouses. In *Proc. of the 3rd Int.*

Conf. on Data Warehousing and Knowledge Discovery (DaWaK), volume 2114 of *LNCS*. Springer, 2001.

[5] W. Giovinazzo. *O-O Data Warehouse Design*. Prentice Hall, 2000.

[6] M. Golfarelli, D. Maio, and S. Rizzi. The Dimensional Fact Model: a Conceptual Model for Data Warehouses. *Int. Journal of Cooperative Information Systems*, 7(2&3):215–247, 1998.

[7] V. Gopalkrishnan, Q. Li, and K. Karlapalem. Star/snow-flake schema driven object-relational data warehouse design and query processing strategies. In *Proc. of 1st Int. Workshop on Data Warehousing and Knowledge Discovery (DaWaK)*, number 1676 in LNCS, pages 11–22. Springer, 1999.

[8] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, 1996.

[9] ISO. *ISO/IEC 9075:1999: Information technology — Database languages — SQL*. Int. Organization for Standardization, 1999.

[10] R. Kimball. *The Data Warehouse toolkit*. John Wiley & Sons, 1996.

[11] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite. *The Data Warehouse lifecycle toolkit*. John Willey & Sons, 1998.

[12] W. Lehner. Modeling large scale OLAP scenarios. In *Advances in Database Technology - EDBT'98*, number 1377 in LNCS. Springer, 1998.

[13] D. L. Moody and M. A. R. Kortink. From enterprise models to dimensional models: A methodology for data warehouse and data mart design. In *Proc. of 2nd Int. Workshop on Design and Management of Data Warehouses (DMDW)*. Swiss Life, 2000.

[14] OMG. *Unified Modeling Language Specification*, September 2001. Version 1.4.

[15] T. B. Pedersen. *Aspects of Data Modeling and Query Processing for Complex Multidimensional Data*. PhD thesis, Faculty of Engineering and Science, Aalborg University (Denmark), 2000.

[16] T. B. Pedersen and C. S. Jensen. Multidimensional data modeling for complex data. In *Proc. of 15th Int. Conf. on Data Engineering (ICDE)*, pages 336–345. IEEE Computer Society, 1999.

[17] C. Sapia, M. Blaschka, G. Höfling, and B. Dinter. Extending the E/R model for the multidimensional paradigm. In *Proc. Int. Workshop on Data Warehouse and Data Mining (DWDM) in conjunction with the ER'98*, number 1552 in LNCS, pages 105–116. Springer, 1999.

[18] A. Shoshani and M. Rafanelli. A Model for Representing Statistical Objects. In *Proc. of the 3rd Int. Conf. on Information Systems and Management of Data (COMAD)*. McGraw-Hill, 1991.

[19] J. C. Trujillo, M. Palomar, J. Gmez, and I.-Y. Song. Designing Data Warehouses with OO Conceptual Models. *IEEE Computer*, 34(12):66–75, 2001.

[20] N. Tryfona, F. Busborg, and J. G. B. Christiansen. starER: A conceptual model for data warehouse design. In *Proc. of ACM 2nd Int. Workshop on Data Warehousing and OLAP (DOLAP)*, 1999.