Multi-star Conceptual Schemas for OLAP Systems

Alberto Abelló [(a)], José Samos [(b)], and Fèlix Saltor [(a)]

[(a)] U. Politècnica de Catalunya (UPC), Dept. de Llenguatges i Sistemes Informàtics
{aabello,saltor}@lsi.upc.es
[(b)] U. de Granada (UGR), Dept. de Lenguajes y Sistemas Informáticos
jsamos@ugr.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 use to be forgotten. This paper studies different kinds of conceptual relationships between stars (i.e. Derivability, Generalization/Specialization, Aggregation/Decomposition, or Temporal), and proposes a 3-level schemas architecture to ease the implementation and usage of multi-star schemas.

Key words: Multidimensional schemas, Conceptual modeling, Semantics, Drill-across

1 Introduction

A Data Warehouse (DW) is, as it was defined in [Inm96], 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 could find difficult to query. On-Line Analytical Processing (OLAP) applications are tools conceived to ease the navigation through the DW. The most important characteristic of OLAP tools is the multidimensional view they offer, close to analysts concepts. Multidimensionality allows 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 retrieve mechanisms to center our efforts in conceptual multidimensional modeling and how it helps OLAP users and designers.

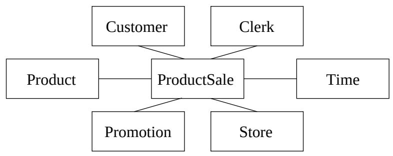

Fig. 1. Star schema of product sales

Multidimensionality conceives data as if they were placed in a hypercube (we will misuse the term *Cube* from here on) 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 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` whom was sold, the `Clerk` who sold it, the `Promotion` that affects it, and the `Store` where was sold.

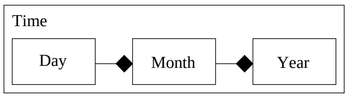

Fig. 2. An aggregation hierarchy in Time dimension

How different analysis data could be defined at different granularities was already pointed out by [SR91], a statistical model close to multidimensional concepts. Thus, some multidimensional models, like [Leh98], [CT98], [GMR98], [SBHD99], [TBC99], or [Ped00], 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 *Cube*s: *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 a 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 [Gio00]. SQL syntax was already extended to support some multidimensional operations as can be seen in [ISO99]. In this paper, we leave aside most of these operations to pay special attention to *Drill-across*.

Quite often, different *Cube*s in a business model are found closely related, and analysts want to jump from one to another. They use to be 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 them. The aim of this paper is to dig into the applicability of *Drill-across* by studying how a multidimensional schema could contain several, related stars, even if the *Cube*s are 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 and presents a 3-level schemas architecture to benefit from relationships between stars; section 4 exemplifies the different kinds of relationships we found between stars; section 5 contains a discussion about the relationships we found; finally, section 6 draws conclusions. Acknowledgements and bibliography close the paper.

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 stars, 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 *Cube*s, which means navigating through data in different star schemas. Unfortunately, the *Drill-across* operation, in these models, is limited to the case that some analysis dimensions are shared by the *Cube*s.

[Kim96] 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". In order 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 stars. This is clearly satisfied, if both dimensions are exactly the same. However, he also explains how it can be satisfied, for instance, if the only difference between the dimensions is their granularity, i.e. the finest detail level they allow.

A later work, [GLK99], presents "multi-star" schemas obtained by normalization of "fact tables", while [PJ99] 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). [MK00] 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 schemas with hierarchically linked "fact tables". A "galaxy" is a collection of star schemas with shared dimensions. Finally, a "star cluster" is a set of star schemas sharing subdimensions. [Gio00] defines "constellation", as well. In this case, it is a set of stars sharing dimensions. The shared dimensions must be "conforming dimensions", which means their values are consistent among the stars. Even though these models allow several stars in a schema, they do not study relationships between them.

Regarding multi-stars architectures, [KRRT98] suggests a "Data Warehouse Bus Architecture", which offers "wire-dimensions" where facts can be plugged. Analysis dimensions are conformed in order to be shared by facts. This is a simple solution to the problem of integrating DMs, which helps to develop DMs at different times, by different teams.

3 Multi-star conceptual schemas

A *Fact* 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 in 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`.

We define a *Dimension* as 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 *Levels*). An in depth explanation of *Dimension*s is in [ASS01a]. Each *Level* corresponds to a granularity in the *Dimension*, and has attributes that allow 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.

If users are only interested in a given *Cell*, they just need to access a *Fact*. However, sometimes, they could desire to relate data in different stars, and OLAP tools should 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 piece of space in a given *Cube*, and want to view the corresponding space in a different one. At first glance, it is 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, but exist some semantic relationship between the *Dimension*s and/or *Fact*s in the stars.

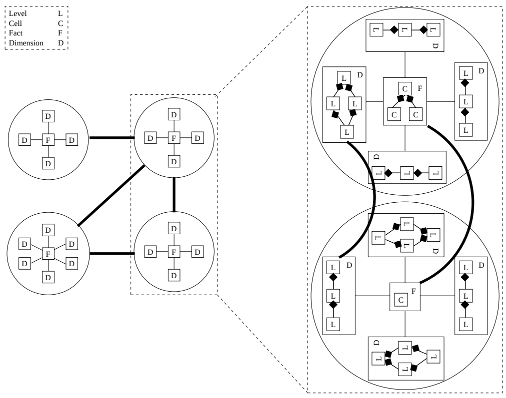

Fig. 3. Multi-star schema

Existing multidimensional models do not consider *semantic domains*, i.e. domains reflecting the conceptualization of values in the mind of the designer. Actually, the analysis dimensions used on drilling across need not exactly coincide in both *Cube*s. We argue that they just should be defined on a common semantic domain. Thus, *Drill-across* between different *Cube*s is performed thanks to semantic relationships between stars (like those drawn with thick lines in figure 3). Two stars can be related whether their *Fact*s, or their *Dimension*s are. We will distinguish four kinds of relationships in this paper, i.e. *Derivation*, *Generalization*, *Association*, and *Flow* (in UML terminology, as defined in [OMG99]).

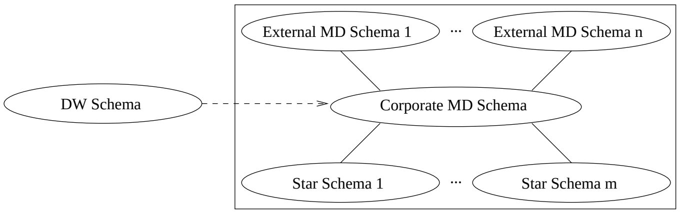

Fig. 4. 3-levels multidimensional schemas architecture

From our point of view, multidimensional data need an integrated access, to be able to drill across through inter-stellar relationships. In this sense, we propose the usage of a 3-level schema architecture like that shown in figure 4 (based on the ANSI/SPARC architecture in [BFJ+86]). The architecture facilitates the usage of semantic relationships at conceptual level, while allows to store independent stars in different DMs. Other levels could also be added to the architecture, as in [SL90], if we were dealing with heterogeneous DMs and multidimensional models.

The unique *Corporate Multidimensional Schema* (CMDS) contains all data in the individual DMs, related by inter-stellar relationships (as sketched in figure 3). Thus, we propose to obtain the CMDS from the DW schema (by means of translation operations between the DW model and the multidimensional model), and later on, store its instances in the DMs. Notice that the possibility of integrating the DMs if they were not defined from a unique DW, or, at least, developed based on a common plan, will be much harder. As pointed out in [IIS98], building independent DMs directly from operational applications is a poor idea. Thus, in this architecture, DMs are obtained by translating the DW to a multidimensional model, and then, implement the different stars found.

At the bottom level of the architecture, we have the different star schemas, which could be stored in either Relational OLAP (ROLAP), Multidimensional OLAP (MOLAP), Object-Oriented OLAP (O3LAP), or any other kind of multidimensional system. These DMs optimize accesses to multidimensional data, and it does not matter whether they were independently defined, or built from a huge, common DW, because we still have a common view of all them at conceptual level, which ensures conformation of data and integrated access. $StarSchema_i$ in figure 4 represents the logical schemas of the different DMs (in the data model corresponding to the chosen multidimensional system).

At top level, we can define *External Multidimensional Schemas* (EMDS), to cover the needs of different users or groups of users. These subsets of information would contain different inter-related *Cube*s, that users could successively visit. Furthermore, since different people may view things in a different way, these schemas offer the possibility to rename concepts, or even define some derived data that was not physically stored in the DMs. View definition mechanisms should be adapted here to a multidimensional model.

4 Inter-stellar relationships

We are interested in providing multidimensional schemas with more than one *Fact*. However, having several facts in the same schema is absolutely useless, if it contains isolated stars. They need to be related in some way to allow *Drill-across*. 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 deep into the capture of semantic relationships between different stars, that allows 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, and vice versa.

4.1 Dimension-Dimension

[KRRT98], [Gio00], as well as some multidimensional tools stress the importance of having "conforming dimensions". In order to drill across different *Cube*s, the corresponding schemas must share analysis dimensions so that their instances exactly coincide. We argue that this position unnecessarily restricts the usage of *Drill-across*. Actually, it is just needed that the selected instances of the *Dimension*s of the origin *Cube* determine instances in the *Dimension*s of the destination *Cube*. Thus, domains used in both *Dimension*s must be related in some way, but *Dimension*s could still be absolutely different. We are going to show four kinds of O-O relationships between *Dimension*s that allow to drill across their *Cube*s.

***Derivation* relationships** Firstly, we could find that the same concept has different names depending on the subject we are. Therefore, the same *Dimension*, with exactly the same instances, will need a different name depending on the context we are going to use it. Moreover, this *Dimension* could not play the same role for different facts. `Product`, may be considered `RawMaterial` in a different context. It is not enough to say they are "synonyms" (like in [Kim96]), 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 considers 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 (ex. letters or numbers).

Sometimes, some *Dimension* instances in a star are only considered grouped in another star, because of lack of interest in the individuals, confidentiality issues, or space problems to keep information at maximum detail. In [Kim96], 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, a *Cube* could keep data by hour, while another one 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 instances. Hence, they will be comparable.

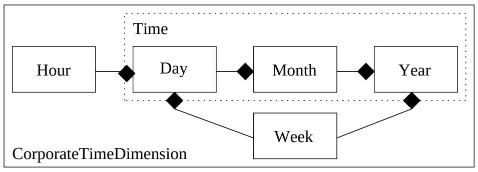

Fig. 5. Example of containment of *Dimension*

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 5 shows how the `Time` *Dimension*, previously presented in figure 2, is included in a more general *Dimension* `CorporateTimeDimension`. `Time` does neither provide `Hour` nor `Week` *Aggregation Levels*, because they are not of interest for the `ProductSales` *Cube* where it is used.

In terms of O-O, considering *Derivability* or *Point-of-view* would allow us to reflect that two *Dimension*s are derived both from a common concept, in spite of the fact that they look different, like in the mentioned examples. Thus, we find that two *Dimension*s can be related by *Derivation*, and users can drill across from a *Cube* to another one through it. The *Dimension* is not shared, because it appears different in each star. By means of this kind of relationship, we can offer users the desired view while they are still able to drill across.

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

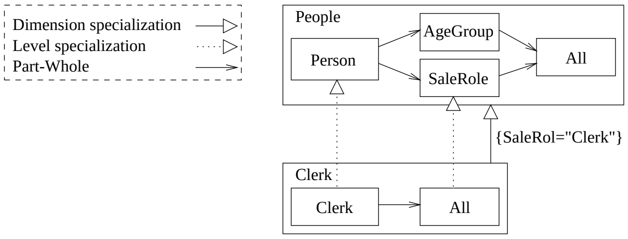

Fig. 6. Example of *Generalization* between *Dimension*

As outlined in [Gio00], using superclasses and subclasses in star schemas, we gain better understanding. Consequences of two *Dimensions* being related by specialization was already studied in [ASS01a]. There, we can see that, as exemplified in figure 6, we should speak of specializing a *Dimension* at a *Level*, rather that just specializing a *Dimension*. If we specialize `People` dimension at `SaleRole` level (solid arrow) to have `Clerk` *Dimension*, this specialization contains a level (i.e. `Clerk`) with each person acting as clerk, and another one with only one element 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 could obtain `Customer` dimension.

In our example, `AgeGroups` aggregation level is not of interest in `Clerk` dimension. Notice that if it would, it would not be specialization of the homonym level in `People` dimension, since its instances would represent different sets of people. For instance, not everybody between twenty and thirty years is a clerk. Therefore, the instance corresponding to this age group in `People` *Dimension* would represent more people than the instance representing the same age group in `Clerk`.

Sharing *Dimensions* would not be enough in this case, neither. `Clerk` has more attributes and much less instances than `Customer`. Therefore, sharing `People` *Dimension* in both two stars would generate lots of undesirable null values.

***Association* relationships** 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 *Cubes*. Clerks use to be assigned to stores. Thus, `Clerk` would be associated with `Store` *Dimension* (maybe multivalued).

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 *Dimensions*, we loose the aggregate one. For example, `Colors` *Dimension* could be used to define `ColoredProduct`. Dissociating `Colors` from `ColoredProduct` means we do not have colored products any more.

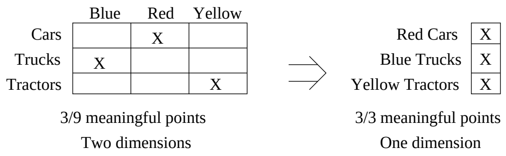

Fig. 7. Example of correlated *Dimension*s

If `ColoredProduct` would be represented as two separate dimensions (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 7). 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 values are meaningful.

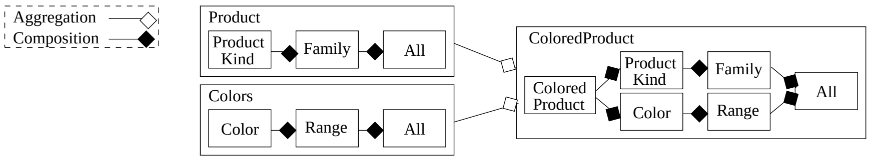

Fig. 8. Example of *Aggregation* between *Dimension*s

Whether it is modeled as two independent *Dimension*s, or only one, 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 8). Relating *Dimension*s by *Aggregation* also has consequences in the aggregation hierarchies. Figure 8 shows how the hierarchy of the aggregated dimension contains those hierarchies in the *Dimension*s we are aggregating (as it is explained in [ASS01a]). However, 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.

Again, this offers navigation possibilities that just sharing *Dimension*s do not offer. A psychological study could include `Colors` *Dimension*. `ColoredProduct` cannot be shared with the star corresponding to that study, because it was about colors, and did not considered products at all. However, analysts will probably be interested on navigating from one star to the other. Thus, it should be allowed through the *Association* between `Colors` and `ColoredProduct` *Dimension*s.

***Flow* relationships** Because of the long periods of interest in analysis tasks and how fast business change nowadays, it is expected that analysis dimensions in our multidimensional schema evolve. Due to the importance of time, it is not acceptable to throw away old dimensions. 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 old data with regard to those attributes, because their values at the time data was collected is 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 he/she is interested on. As studied in [EK01], 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 that 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 dimensions 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 *Cube*, we will be able to select the corresponding set in a related *Cube*. This means that we can also use relationships between *Fact*s to navigate. The relationships between the structures of two related *Fact*s were already studied in [ASS01b].

1. 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 benefits of some products sold (the best sales, sales involving relevant products, etc.). Most *Dimension*s will be likely shared by both *Cube*s (i.e. `EmployeeEfficiency`, and `ProductSale`). However, we could also travel between them due to data in some cells are obtained by operating 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 *Cube*s belong to different subjects, and selection of *Cell* instances is not performed by means of aggregation hierarchies.

2. 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 another star. 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 *Cube* to another one just because the *Association* relationship between the *Fact*s.

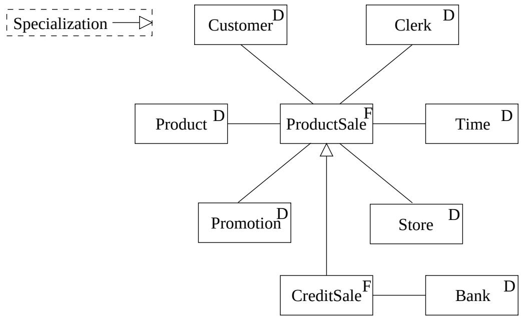

Fig. 9. Example of *Generalization* between *Fact*s

3. 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 9, `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 *Cube*s just because their *Cell*s are specialization one of another. Usually, they will also share most analysis dimensions, but sharing them is not needed in this case to drill across, since *Fact* domains coincide.
4. 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 *Facts* to `Time` *Dimension*, since we actually have different *Cubes* with 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, we should have different stars containing data about the same fact before and after the acceptance of the checks, because the *Cell* structure changed. Analists would be able to relate those data by means of *Flow* relationships between *Facts*.

None of these relationships can be reflected by sharing *Dimensions*. They reflect correspondences among factual information, which is also important to navigate. We can go from a star to another, if cells in the first determine cells in the other.

4.3 Fact-Dimension

The last possibility to navigate through different stars 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 *Cubes* from the same point of view. Rather, we are using results of querying a *Cube* 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 star schemas.

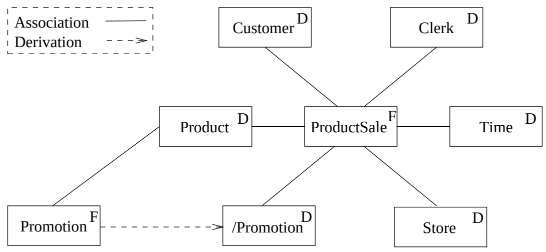

Fig. 10. Example of *Association/Derivation* between *Fact* and *Dimension*

1. 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. *Facts* use to have much more instances than *Dimensions*. Nevertheless, by grouping them, we could obtain coarser aggregation levels of interest. *Measures* use to be numerical, while attributes in *Dimensions* use to be descriptive. Therefore, derivation (exemplified in figure 10) 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.

2. 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, see figure 10. Thus, `Product` *Dimension* could have an attribute defined on domain `Promotion` *Fact* (i.e. association arrow in figure 10). 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

Table 1. Summary table

Relationships	D-D	F-F	F-D	D-F
Derivation	$\surd$	$\surd$	$\surd$	$\times$
Generalization	$\surd$	$\surd$	$\times$	$\times$
Association	$\surd$	$\surd$	$\surd$	$\surd$
Flow	$\surd$	$\surd$	$\times$	$\times$

Table 1 sums up the kinds of relationships we found of interest between different stars. What could attract attention is that we did not find *Generalization*, nor *Flow* relationships between a *Fact* and a *Dimension*. This is because we consider that a temporal transformation can convert neither *Fact*s into *Dimension*s, nor vice-versa; and they are different enough not to be related by *Generalization*. 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 are given information.

Drill-across means use the analysis framework we are using for a given *Cube* to analyze 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 *Dimensions* that would also allow it. If two *Dimension*s are related by *Derivation*, *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 *Association*, instances do not coincide, but an instance of a *Dimension* determines instances in the other.

Actually, it is not necessary the *Dimensions* in the destination *Cube* 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 *Facts* are related in some way. Thus, we just need to substitute *Measures* of one cell, by those of its counterpart in the other *Cube*. In this way, we are also able to study data in two different *Cubes* whose *Facts* are related.

Relationships between a *Fact* and a *Dimension* do not allow proper *Drill-across*. However, if selected cells determine a set of points in a *Dimension*, these can be used in the analysis of another *Cube*.

[Kim96] points out that it rarely makes sense to restrict simultaneously two *Dimensions* in the same star by the same condition. We completely agree that it is senseless to apply the same constraint to two sets of instances over different semantic domains. However, Kimball also explains how an analysis dimension can play different roles in the same star (for instance, `People` acting as `Clerk` or `Customer` in our example). If so, it could be constrained for both roles at once. Furthermore, if we would have two semantically related *Dimensions* in the same star, both could also be constrained at the same time, because they would be "type-compatible" in one way or another. For example, we could study clerks that are our customers. Therefore, the relationships found in this work should not only be considered to *Drill-across*, but also for operations that involve isolated star schemas, i.e. *Dice*. A condition could be simultaneously applied to several *Dimensions* defined over the same semantic domain (like `Clerk` and `Customer`). Moreover, instances of a *Dimension* can be selected by selecting instances of a related one.

6 Conclusions

Along this paper we have described relationships we found between star schemas for OLAP tools. We explained how different stars can be related by *Derivation*, *Generalization*, *Association*, or even *Flow* relationships (in UML terminology). We showed how those relationships, between analysis dimensions and the different kinds of cells, could be used to navigate or *Drill-across* between *Cubes*. Moreover, these relationships could also be used in other multidimensional operations like *Dice*.

These relationships are not only useful for analysts, but also for designers. It has been exemplified that semantic relationships between *Dimensions* have nice consequences for aggregation hierarchies inside *Dimensions*, since two related *Dimensions* contain related aggregation hierarchies. The same stands for different *Cells* in related *Facts*. Therefore, relationships between stars can also be used to drive designers work, so that they can detect inconsistences and errors. Conformed stars would also drive users to the usage of data in a uniform way.

Acknowledgements This work has been partially supported by the Spanish Research Program PRONTIC under projects TIC2000-1723-C02-01 and TIC2000-1723-C02-02, as well as the grant 1998FI-00228 from the Generalitat de Catalunya.

References

[ASS01a] A. Abelló, J. Samos, and F. Saltor. Understanding Analysis Dimensions in a Multidimensional Object-Oriented Model. In *Proc. of the 3rd Int. Workshop on Design and Management of Data Warehouses (DMDW)*. SwissLife, 2001.

[ASS01b] 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 a nd OLAP (DOLAP)*. ACM, 2001. To appear.

[BFJ+86] T. Burns, E. N. Fong, D. Jefferson, R. Knox, L. Mark, C. Reedy, L. Reich, N. Roussopoulos, and W. Truszkowski. Reference Model for DBMS Standardization, Database Architecture Framework Task Group (DAFTG) of the ANSI/X3/SPARC Database System Study Group. *SIGMOD Record*, 15(1):19–58, 1986.

[CT98] 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.

[EK01] 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.

[Gio00] W. Giovinazzo. *Object-Oriented Data Warehouse Design*. Prentice Hall, 2000.

[GLK99] 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.

[GMR98] 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.

[IIS98] W. H. Inmon, Claudia Imhoff, and Ryan Sousa. *Corporate Information Factory*. John Wiley & Sons, 1998.

[Inm96] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, second edition, 1996.

[ISO99] ISO. *ISO/IEC 9075:1999: Information technology — Database languages — SQL*. International Organization for Standardization, 1999.

[Kim96] R. Kimball. *The Data Warehouse toolkit*. John Wiley & Sons, 1996.

[KRRT98] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite. *The Data Warehouse lifecycle toolkit*. John Willey & Sons, 1998.

[Leh98] W. Lehner. Modeling large scale OLAP scenarios. In *Advances in Database Technology - EDBT'98*, number 1377 in LNCS, pages 153–167. Springer, 1998.

[MK00] 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.

[OMG99] OMG. *Unified Modeling Language Specification*, June 1999. Version 1.3.

[Ped00] 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.

[PJ99] 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.

[SBHD99] 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.

[SL90] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heteroge neous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[SR91] 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.

[TBC99] 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.