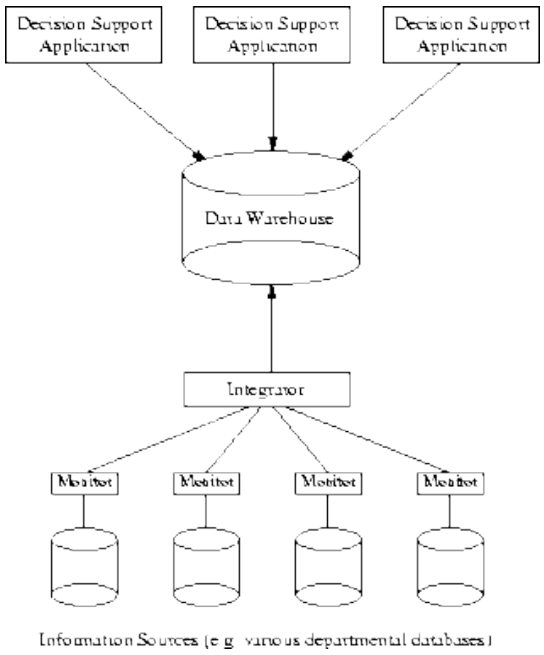
Introduction to temporal Data warehouse: General Concepts, Temporality Data Types, Synchronization and Relationships, Temporal Extension of the Multi Dimensional Model, Temporal Support for Levels, Temporal Hierarchies, Fact Relationships, Measures, Conceptual Models for Temporal Data Warehouses : Logical Representation and Temporal Granularity

Temporal databases and data warehousing are two separate areas which are strongly related: data warehouses are the commercial products that require temporal database technology. Naturally, most other database products are amenable to temporal database technology, too. Regarding the market perspectives, however, one has to assume that it will be mainly data warehouses that adopt the techniques that have been and that will be developed by temporal database researchers. In this section, we want to elaborate the connection between data warehousing and temporal databases in some more detail.

A *data warehouse* (DW) integrates information from many, possibly heterogeneous, databases into a physically separated database and makes this information available to analysis [Inmon, 1996]. Figure 2.5 illustrates this concept. The purpose of the analysis might be, for example, to provide the management of a company with information on trends and facts that are required for taking strategic decisions.

Figure: The concept of a data warehouse.



Trend analysis can go along many dimensions, the most important of which is *time*. It is used to detect certain characteristics in the evolution of data, e.g. over time or over various geographic regions or over product lines

A **temporal database** stores data relating to time instances. It offers temporal data types and stores information relating to past, present and future time. Temporal databases could be uni-temporal, bi-temporal or tri-temporal.

More specifically the temporal aspects usually include valid time, transaction time or decision time

- **Valid time** is the time period during which a fact is true in the real world.
- **Transaction time** is the time period during which a fact stored in the database was known.
- **Decision time** is the time period during which a fact stored in the database was decided to be valid.

Temporal Data Types

Teradata provides temporal table support at the data type level with period data types. A period is an anchored duration that represents a set of contiguous time granules within the duration. It has a beginning bound (defined by the value of a beginning element) and an ending bound (defined by the value of an ending element). Beginning and ending elements can be DATE, TIME, or TIMESTAMP types, but both must be the same type.

The duration that a period represents starts from the beginning bound and extends up to, but does not include, the ending bound.

- The DATE data type
- The TIMESTAMP data types:
    - TIMESTAMP
    - TIMESTAMP WITH TIME ZONE
    - TIMESTAMP WITH LOCAL TIME ZONE
- The INTERVAL data types:
    - INTERVAL YEAR TO MONTH
    - INTERVAL DAY TO SECOND

The DATE Data Type

Oracle's DATE data type holds date as well as time information. Regardless of the date format you use for display purposes, Oracle stores dates internally in one standard format. Internal to the database, a date is a fixed-length, 7-byte field. The seven bytes represent the following pieces of information:

- The Century
- The Year
- The Month

- The Day
- The Hour
- The Minute
- The Second

The TIMESTAMP Data Types

To provide support for fractional seconds along with date and time data, and also to provide support for time zones, Oracle9i introduced the following temporal data types:

- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE

The INTERVAL Data Types

Date and time interval data are an integral part of our day-to-day life. Common examples of interval data are the age of a person, the maturity period of a bond or certificate of deposit, and the warranty period of your car. Prior to Oracle9i Database, we all used the NUMBER data type to represent such data, and the logic needed to deal with interval data had to be coded at the application level. Oracle9i Database introduced two new data types to handle interval data:

- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND

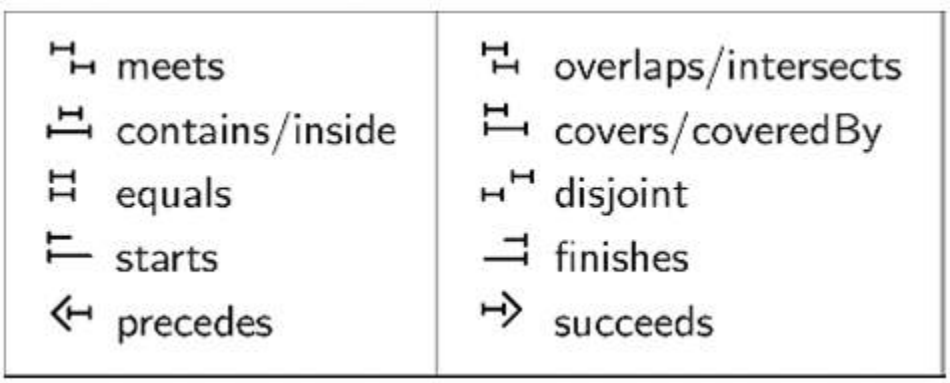
Synchronization Relationships

Synchronization relationships specify how two temporal extents relate to each other. They are essential in temporal applications, since they allow one to determine, for example, whether two events occur simultaneously or whether one precedes the other.

The synchronization relationships for temporal data correspond to the topological relationships for spatial data. They are defined in a similar way, on the basis of the concepts of the boundary, interior, and exterior.

The exterior of a temporal value is composed of all the instants of the underlying time frame that do not belong to the temporal value. On the other hand, the interior of a temporal value is composed of all its instants that do not belong to the boundary. The boundary is defined for the different temporal data types as follows. An instant has an empty boundary. The boundary of an interval consists of its start and end instants. The boundary of a ComplexTime value is

(recursively) defined by the union of the boundaries of its components that do not intersect with other components.



synchronization relationships

meets: Two temporal values meet if they intersect in an instant but their interiors do not. Note that two temporal values may intersect in an instant but do not meet.

overlaps: Two temporal values overlap if their interiors intersect and their intersection is not equal to either of them.

contains/inside: contains and inside are symmetric predicates: a contains b if and only if b inside a. A temporal value contains another one if the interior of the former contains all instants of the latter.

covers/coveredBy: covers and coveredBy are symmetric predicates: a covers b if and only if b coveredBy a. A temporal value covers another one if the former includes all instants of the latter. This means that the former contains the latter, as defined above, but without the restriction that the boundaries of the temporal extents do not intersect. As a particular case, the two temporal values may be equal.

disjoint/intersects: disjoint and intersects are inverse temporal predicates: when one applies, the other does not. Two temporal values are disjoint if they do not share any instant.

equals: Two temporal values are equal if every instant of the first value belongs also to the second and conversely.

starts/finishes: A temporal value starts another if the first instants of the two values are equal. Similarly, a temporal value finishes another if the last instants of the two values are equal.

precedes/succeeds: A temporal value precedes another if the last instant of the former is before the first instant of the latter. Similarly, a temporal value succeeds another if the first instant of the former is later than the last instant of the latter.

Overview of the Model

The MultiDim model allows users to determine which temporal data they need by including in the schema the symbols of the corresponding temporality types. For example, in the schema in Fig. users are not interested in keeping track of changes to clients' data; thus, this dimension does not include any temporal support. On the other hand, changes in the values of measures and changes in data related to products and stores are important for analysis purposes, as indicated by the various temporality types included in the schema. the MultiDim model allows both temporal and nontemporal attributes, levels, parent-child relationships, hierarchies, and dimensions. The definitions of the nontemporal elements of the model remain the same.

we define a temporal level as a level for which the application needs to store the time frame associated with its members. The schema in Fig includes four temporal levels (as shown by the LS symbol next to the level name): Product, Category, Store, and Sales district. This allows users to track changes in a member as a whole, for example inserting or deleting a product or a category. The usual nontemporal levels are called conventional levels e.g. Client.

A temporal attribute is an attribute that keeps track of the changes in its values and the time when they occur. For instance, the valid time support for Size and Distributor in the Product level indicates that the history of changes in these attributes will be kept. As in the spatial case, in our model we adopt an orthogonal approach where a level may be temporal independently of the fact that it has temporal attributes. A temporal parent-child relationship keeps track of the time frame associated with the links relating a child and a parent member. For example, in Fig. The symbol LS in the relationship linking Product and Category indicates that the evolution in time of the assignments of products to categories will be stored.

Cardinalities in parent-child relationships constrain the minimum and the maximum number of members in one level that can be related to a member in another level. Temporal support for parent-child relationships leads to two interpretations of cardinalities. The instant cardinality is valid at every time instant, whereas the lifespan cardinality is valid over the entire member's lifespan. The instant cardinality is represented using the symbol for the temporality type (for example, LS in Fig. next to the line for the parent-child relationship, whereas the lifespan cardinality includes the symbol LS surrounded by an ellipse. When the two cardinalities are the same, this is represented using only one cardinality symbol. the instant cardinality between Store and Sales district levels is one-to-many, while the lifespan cardinality is many-to-many. These cardinalities indicate, in particular, that a store belongs to only one sales district at any time instant, but may belong to many sales districts over its lifespan, i.e., its assignment to sales districts may change. On the other hand, the instant and lifespan cardinalities between the Product and Category levels are both one-to-many. They indicate, in particular, that products may belong to only one category over their lifespan.

We define a temporal hierarchy as a hierarchy that has at least one temporal level or one temporal parent-child relationship. Thus, temporal hierarchies can combine temporal and nontemporal levels. Similarly, a temporal dimension is a dimension that has at least one temporal hierarchy. The usual nontemporal dimensions and hierarchies are called conventional dimensions and conventional hierarchies.

A temporal fact relationship is a fact relationship that requires a temporal join between two or more temporal levels. This temporal join can be based on various synchronization relationships: an icon in the fact relationship indicates the synchronization relationship used for specifying the join. For example,the temporal fact relationship Sales in Fig. relates two temporal levels: Product and Store. The overlaps synchronization icon in the relationship indicates that users focus their analysis on those products whose lifespan overlaps the lifespan of their related store. If a synchronization icon is not included in a fact relationship, there is no particular synchronization constraint in the instances of that relationship. In our example, this could allow users to analyze whether the exclusion of some products from stores affects sales.

Temporality Types

They are several ways of interpreting the time frame associated with the facts contained in a temporal database. These interpretations are captured by several temporality types.

Valid time (VT) specifies the period of time in which a fact is true in the modeled reality; for example, it allows one to capture when a specific salary was paid to an employee. The valid time is usually supplied by the user.

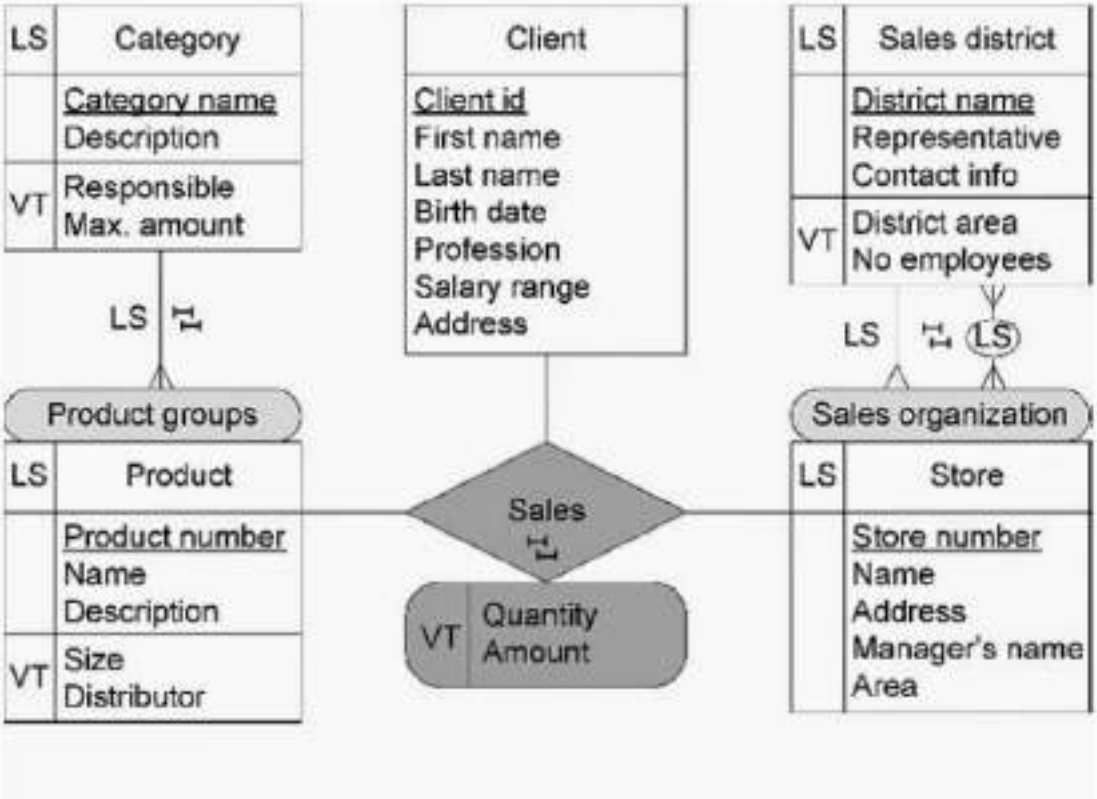
Transaction time (TT) indicates the period of time in which a fact is current in the database and may be retrieved. The transaction time of a fact begins at the time when it is inserted or updated and ends when the fact is deleted or updated. The transaction time is generated by the database system.

Valid time and transaction time can be combined to define bitemporal time (BT). This indicates both when a fact is true in reality and when it is current in the database.

In many applications it is necessary to capture the time during which an object exists. This is represented by the lifespan (LS) or existence time; for example, it can be used to represent the duration of a project. The lifespan of an object o may be seen as the valid time of the related

fact, “o exists.” Lifespan also applies to relationships, for example, it may be used to capture the time during which an employee has worked for a project. In addition to a lifespan, an object or relationship can also record a transaction time,indicating the time when it is current in the database.

In addition, since data in data warehouses is neither modified nor deleted,we proposed the loading time (LT), which indicates the time since when thedata is current in a data warehouse. This time can differ from the transaction time of the source systems owing to the delay between the time when a change occurs in a source system and the time when this change is integrated into a temporal data warehouse. The loading time can help users to know the time since when an item of data has been available in a data warehouse for analysis purposes.



Conceptual Models for Temporal Data Warehouses

The MultiDim model is a conceptual multidimensional model for data warehouse and OLAP applications. These applications require the presence of a time dimension to track changes in measure values. However, the time dimension cannot be used to represent changes in other dimensions.In this paper we introduce a temporal extension of the MultiDim model. This extension is based on research realized in temporal databases. We allow different temporality types: valid time, transaction time, and lifespan, which are obtained from source systems, and loading time, which is generated in the data warehouse. Our model provides temporal support for levels, attributes, hierarchies, and measures. For hierarchies we discuss different cases depending on whether the changes in levels or in the relationships between them must be

kept. For measures, we give different scenarios that show the usefulness of the different temporality types. Further, since measures can be aggregated before being inserted into data warehouses, we discuss the issues related to different time granularities between source systems and data warehouses. We finish the paper presenting a transformation of the MultiDim model into the entity-relationship and the object-relational models.

Granularity means to uniquely identify the information. It mostly means the level of the information stored in the databases. For example you can identify the single transaction for a given product and can store it at least level of granularity. Same information can also be available at order number level which will not include the products for the same order.

If you store the above information on the order number level and don't include product information . You will miss any business insight available at product level.if business is trying to analyze the sales or orders based on Product, they will not be able to do that without product information available to them

Data warehousing principles recommend to store information at the lowest level and summary can be produced from the least granular level data

All data have different kinds of structure and granularity. Tables are structured to suit end user needs, and granularity defines the level of detail the data provides.

In a data warehouse, On-Line Analytical Processing (OLAP) is the answer engine that provides the infrastructure for ad-hoc user query and multi-dimensional analysis in a data warehouse. Granularity is one of the three critical elements of OLAP design that include: Data Warehouse Design: OLAP Cube Design

- Grouping measures - numerical values you want to analyze such as revenue, number of customers, how many products customers purchase, or average purchase amount.
- Dimension - where measures are stored for analysis such as geographic region, month, or quarter.
- Granularity - the lowest level of detail that you want to include in the OLAP dataset.

The single most important design issue facing the data warehouse developer is determining the proper level of granularity of the data that will reside in the data warehouse. When the level of granularity is properly set, the remaining aspects of design and implementation flow smoothly; when it is not properly set, every other aspect is awkward.

Granularity is also important to the warehouse architect because it affects all the environments that depend on the warehouse for data. Granularity affects how efficiently data can be shipped to the different environments and determines the types of analysis that can be done.

The primary issue of granularity is that of getting it at the right level. The level of granularity needs to be neither too high nor too low.

The trade-off in choosing the right levels of granularity (as discussed in Chapter 2) centers around managing the volume of data and storing data at too high a level of granularity, to the point that detailed data is so voluminous that it is unusable. In addition, if there is to be a truly large amount of data, consideration must be given to putting the inactive portion of the data into overflow storage.

Raw Estimates

The starting point for determining the appropriate level of granularity is to do a raw estimate of the number of rows of data and the DASD (direct access storage device) that will be in the data warehouse. Admittedly, in the best of circumstances, only an estimate can be made.