Dimension Tables

- A dimension is a structure, often composed of one or more hierarchies, that categorizes data.
- Dimensional attributes help to describe the dimensional value.
- They are normally descriptive, textual values.

Dimension Tables

- Several distinct dimensions, combined with facts, enable you to answer business questions.
- Commonly used dimensions are customers, products, and time.

Dimension Tables

- Dimension data is typically collected at the lowest level of detail and then aggregated into higher level totals that are more useful for analysis.
- These natural rollups or aggregations within a dimension table are called hierarchies.

Hierarchies

- Hierarchies are logical structures that use ordered levels as a means of organizing data.
- A hierarchy can be used to define data aggregation.
- For example, in a time dimension, a hierarchy might aggregate data from the month level to the quarter level to the year level.

Hierarchies

- A hierarchy can also be used to define a navigational drill path and to establish a family structure.
- Within a hierarchy, each level is logically connected to the levels above and below it.
- Data values at lower levels aggregate into the data values at higher levels.

Hierarchies

- A dimension can be composed of more than one hierarchy.
- For example, in the product dimension, there might be two hierarchies--one for product categories and one for product suppliers.

Hierarchies

- Dimension hierarchies also group levels from general to granular.
- Query tools use hierarchies to enable you to drill down into your data to view different levels of granularity.
- This is one of the key benefits of a data warehouse.

Hierarchies

- When designing hierarchies, you must consider the relationships in business structures.
- For example, a divisional multilevel sales organization.
- Hierarchies impose a family structure on dimension values.

Hierarchies

- For a particular level value, a value at the next higher level is its parent, and values at the next lower level are its children.
- These familial relationships enable analysts to access data quickly.

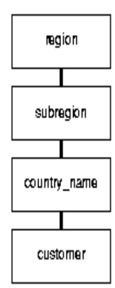
Level Relationships

- Level relationships specify top-to-bottom ordering of levels from most general (the root) to most specific information.
- They define the parent-child relationship between the levels in a hierarchy.

Level Relationships

- Hierarchies are also essential components in enabling more complex rewrites.
- For example, the database can aggregate an existing sales revenue on a quarterly base to a yearly aggregation when the dimensional dependencies between quarter and year are known.

Typical Levels in a Dimension Hierarchy



Unique Identifiers

- Unique identifiers are specified for one distinct record in a dimension table.
- Artificial unique identifiers are often used to avoid the potential problem of unique identifiers changing.
- Unique identifiers are represented with the # character.
- For example, #customer_id.

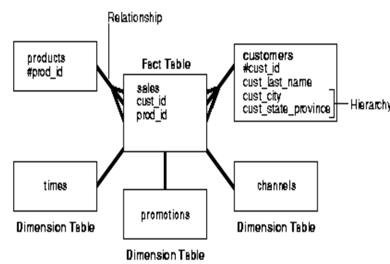
Relationships

- Relationships guarantee business integrity.
- An example is that if a business sells something,
- there is obviously a customer and a product.
- Designing a relationship between the sales information in the fact table and
- the dimension tables products and customers enforces the business rules in databases.

Example of Data Warehousing Objects and Their Relationships

- Figure below illustrates a common example of a sales fact table and dimension tables customers, products, promotions, times, and channels.

Typical Data Warehousing Objects



Physical Design in Data Warehouses

- The following will be discussed
- Moving from Logical to Physical Design
- Physical Design

Moving from Logical to Physical Design

- Logical design is what you draw with a pen and paper or design with Oracle Warehouse Builder or Designer before building your warehouse.
- Physical design is the creation of the database with SQL statements.

Moving from Logical to Physical Design

- During the physical design process, you convert the data gathered during the logical design phase into a description of the physical database structure.
- Physical design decisions are mainly driven by query performance and database maintenance aspects.

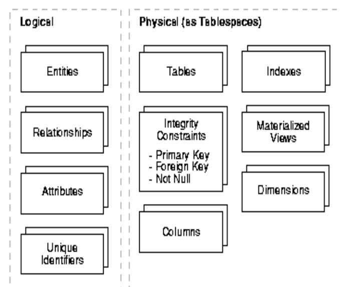
Physical Design

- During the logical design phase, you defined a model for your data warehouse consisting of entities, attributes, and relationships.
- The entities are linked together using relationships.
- Attributes are used to describe the entities.
- The unique identifier (UID) distinguishes between one instance of an entity and another.

Physical Design

- Figure below offers a graphical way of looking at the different ways of thinking about logical and physical designs.

Logical Design Compared with Physical Design



Logical Design Compared with Physical Design

- During the physical design process,
- you translate the expected schemas into actual database structures.
- At this time, you have to map:

Logical Design Compared with Physical Design

- Entities to tables
- Relationships to foreign key constraints
- Attributes to columns
- Primary unique identifiers to primary key constraints
- Unique identifiers to unique key constraints

Physical Design Structures

- Once you have converted your logical design to a physical one,
- you will need to create some or all of the following structures:
- Tablespaces
- Tables and Partitioned Tables
- Views
- Integrity Constraints
- Dimensions

Physical Design Structures

- Some of these structures require disk space.
- Others exist only in the data dictionary.
- Additionally, the following structures may be created for performance improvement:
- Indexes and Partitioned Indexes
- Materialized Views

Tablespaces

- A tablespace consists of one or more datafiles,
- which are physical structures within the operating system you are using.
- A datafile is associated with only one tablespace.
- From a design perspective, tablespaces are containers for physical design structures.

Physical Design Structures

- Tablespaces need to be separated by differences.
- For example, tables should be separated from their indexes and
- small tables should be separated from large tables.

Physical Design Structures

- Tablespaces should also represent logical business units if possible.
- Because a tablespace is the coarsest granularity for backup and
- recovery or the transportable tablespaces mechanism,
- the logical business design affects availability and maintenance operations.

Tables and Partitioned Tables

- Tables are the basic unit of data storage.
- They are the container for the expected amount of raw data in your data warehouse.
- Using partitioned tables instead of nonpartitioned ones addresses the key problem of supporting very large data volumes

Physical Design Structures

- by allowing you to decompose them into smaller and more manageable pieces.
- The main design criterion for partitioning is manageability,
- though you will also see performance benefits in most cases because of partition pruning or intelligent parallel processing.

Physical Design Structures

- For example, you might choose a partitioning strategy based on a sales transaction date and a monthly granularity.
- If you have four years' worth of data, you can delete a month's data as it becomes older than four years with a single, quick DDL statement and
- load new data while only affecting 1/48th of the complete table.

Physical Design Structures

- Business questions regarding the last quarter will only affect three months, which is equivalent to three partitions, or 3/48ths of the total volume.
- Partitioning large tables improves performance because each partitioned piece is more manageable.

Physical Design Structures

- Typically, you partition based on transaction dates in a data warehouse.
- For example, each month, one month's worth of data can be assigned its own partition.

Data Segment Compression

- You can save disk space by compressing heap-organized tables.
- A typical type of heap-organized table you should consider for data segment compression is partitioned tables.
- To reduce disk use and memory use (specifically, the buffer cache),

Data Segment Compression

- you can store tables and partitioned tables in a compressed format inside the database.
- This often leads to a better scaleup for read-only operations.
- Data segment compression can also speed up query execution.

Data Segment Compression

- There is, however, a cost in CPU overhead.
- Data segment compression should be used with highly redundant data,
- such as tables with many foreign keys.
- You should avoid compressing tables with much update or other DML activity.

Data Segment Compression

- Although compressed tables or partitions are updatable,
- there is some overhead in updating these tables, and
- high update activity may work against compression by causing some space to be wasted.

Views

- A view is a tailored presentation of the data contained in one or more tables or other views.
- A view takes the output of a query and treats it as a table.
- Views do not require any space in the database.

Integrity Constraints

- Integrity constraints are used to enforce business rules associated with your database and to prevent having invalid information in the tables.
- Integrity constraints in data warehousing differ from constraints in OLTP environments.

Integrity Constraints

- In OLTP environments, they primarily prevent the insertion of invalid data into a record,
- which is not a big problem in data warehousing environments because accuracy has already been guaranteed.

Integrity Constraints

- In data warehousing environments, constraints are only used for query rewrite.
- NOT NULL constraints are particularly common in data warehouses.
- Under some specific circumstances, constraints need space in the database.
- These constraints are in the form of the underlying unique index.

Indexes and Partitioned Indexes

- Indexes are optional structures associated with tables or clusters.
- In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.
- Bitmap indexes are optimized index structures for set-oriented operations.

- Additionally, they are necessary for some optimized data access methods such as star transformations.
- Indexes are just like tables in that you can partition them,

---

- although the partitioning strategy is not dependent upon the table structure.
- Partitioning indexes makes it easier to manage the warehouse during refresh and improves query performance.

Materialized Views

- Materialized views are query results that have been stored in advance
- so long-running calculations are not necessary when you actually execute your SQL statements.
- From a physical design point of view,
- materialized views resemble tables or partitioned tables and behave like indexes.