Logical design

The logical desing refers to the relational model ROLAP which is based on different rules (data redundancy and tables non in normal form) with respect to the standard logical design. The principal schemas are:

. *star schema;*

. *snowflake schema.*

The *star* schema uses *reletional tables* (also called *dimensional tables*) to represent each dimension and each instance of a dimension is a record into the relational table. Automatically a surrogate primary key is generated which is a foreign key for the *fact table*; this table is unique for each fact schema and its own primary key is composed by all foreign keys of relational tables. The star schema is the logical representation for multidimensional model usually describe by the cube and it can be obtained joined all relational tables. Attributes of reletional tables may have functional dependance between them: since someone may not be the primary key the table is not in normal form so some data redundancy it is present.

The *snowflake* schema separates functional dependecies by partitioning dimensions in more than one dimensional table: the partition is realized cutting the hierarchy on a given attribute when a functional dependance is present; in that way data redundancy is avoided because all tables are in normal form. Notice that for each new table created a foreign key is needed in order to correlates all tables of the dimension: infact it is possible to rebuild the entire dimension simply joined all tables. This is a disadvantage because in order to rebuild the cube more joins have to be performed respect to the star schema.

The most used schema is the star schema because the snowflake reduce the storage space but not enought so the principal advantage of that schema is not so much useful; snowflake can be used in case of the same part of the hierarchy is shared among different dimensions.

Multiple edges

Multiple edges represent a *many* to *many* relationship among two attributes; there are two possible realizations:

. *brigde table* (most used);

. *push down.*

In the first one a new table is created: it is called *bridge table* because conjuncts the two tables involved. The most important thing to remind is

that the bridge table has a new attribute which represent the weight of the contribution of tuples in the relationship.

The *push down* method integrates multiple edges in the fact table: this can increase a lot the size of the fact table because increase redundancy and the other disadvantage is that also the weight have to be encapsulated into the fact table.

Possible kinds of queries which can be performed on multiples edge relations are:

. weighted query: the weight of contribution is considered;

. impact query: the weight is not considered.

The second kind of query is very difficult to perform with the *push down* method because there is not a specific attribute who shows it as the *bridge table* method has. Moreover the query execution time for push down takes care of the following aspects:

. less joins;

. higher cardinality.

Degenerate dimensions

This kind of dimensions are dimensions with a single attribute: if for them the reletional model is consider the resultant table will have the surrogate key plus the attribute. It means that manage it will be more expensive than other methods although surrogate keys have small size. Possible implementations are:

. direct integration into the fact table;

. junk dimension: a single dimension is able to contain several degenerate dimension without functional dependance among their attributes (it means that all possible combinations of attributes are allowed so it generates the same size of a cartesian product).

Materialized views

A *view* is a query which can be called using a name: in order to reduce future operations and speed up query execution it is possible to precompute summaries of the fact table and materialized view will be exploited looking at aggregations already done. In the set of possible *best* materialized views the best one is selected taking into account the following costs:

. query execution cost;

. view maintainance cost (update).

Constraint that have to be respected are:

. avaiable space;

. time window for update (remember that to update a data warehouse the system is taken off so nobody can access to it);

. response time;

. data freshness.