DAYHUFFGROUP

Why Star Schemas Work

Connecting Content | Driving Performance

The Need

Business users need to query the data warehouse and to create reports with little or no intervention from IT. Adopting a reporting and on-line analytical processing (OLAP) tool that engages a full semantic layer with cubing capabilities will improve the business user experience and lessen the reliance on IT. OLAP tools, while capable of connecting to a third normal form database, prefer a star schema designed database. The goal of any semantic layer is to produce user defined queries with only inner JOINs. Here are two examples of the expected SQL code. The first is a query against one star and the second of a cross-star join.

Star Schema Data Models

Star schema data warehouse models are designed for analytics. Specifically, a star schema:

- Is very easy to understand, even for non–technical business managers
- Provides better performance and shorter query times
- Is extensible and will handle future changes more easily than the source transactional system

Why It Works

In the star schema design, the fact table is made up of nothing but keys and additive facts, or integers and some decimal numbers. It turns out the database stores these very efficiently. A star schema uses two kinds of database tables: a dimension table and a fact table. Each of these types of tables plays a unique role in the overall design.

Dimensions can be thought of as categories. Dimensions usually have hierarchies within them. For example, a Sales dimension could have Global (all geographies) at the top of the hierarchy. Global can then be broken down into many Geographies and Divisions can then be further broken down into many States. An example would be: Global, Western Geography, Small Business Division, California.

A fact table contains the measurements or metrics or facts of the business processes, in this case student registrations. A measurement of this business process would be student FTE, revenue and headcount. The tuition revenue is captured in the fact table.

Star SQL

Star schemas are extremely user–friendly and allow users to generate SQL code which is efficient and can be processed by a database. Using a star schema and a BI tool, users will generate queries like the sample below. This kind of query technique will allow users to run queries across different units of business. The users are allowed to change the columns that they need and the filtered conditions in the 'where' clause as highlighted below. From an IT perspective it allows users to generate a standard query and from a user perspective it allows them great flexibility.

DAYHUFFGROUP

Why Star Schemas Work

Single Star Query

```
SELECT
        STAR1_DIMENSION_1.COLUMN_1
        STAR1_DIMENSION_1.COLUMN_N
        STAR1_DIMENSION_2.COLUMN_1
        STAR1_DIMENSION_2.COLUMN_N
        STAR1_DIMENSION_N.COLUMN_1
        STAR1_DIMENSION_N.COLUMN_N
        STAR1_FACT.COLUMN_1
        STAR1_FACT.COLUMN_N

FROM
        STAR1_DIMENSION_1
        STAR1_DIMENSION_2
        STAR1_DIMENSION_N
        STAR1_FACT

WHERE
        STAR1_DIMENSION_1.KEY_1 = STAR1_FACT.KEY1_COLUMN
        STAR1_DIMENSION_2.KEY_2 = STAR1_FACT.KEY2_COLUMN
        STAR1_DIMENSION_3.KEY_3 = STAR1_FACT.KEY3_COLUMN
        STAR1_DIMENSION_N.KEY_N = STAR1_FACT.KEYN_COLUMN

Other WHERE clause conditions
```

User Drop and Drag
Users may drop and drag any data elements onto a report

Semantic Layer
All joins are INNER JOINs and managed by the semantic layer.

Tables used is controlled by the semantic layer and updated in only once location.

User Filter
The user enters filter criteria like Year and Tern or Major

Cross-Star Query

```
SELECT
        STAR1_DIMENSION_1.COLUMN_1
        STAR1_DIMENSION_1.COLUMN_N
        STAR1_DIMENSION_2.COLUMN_1
        STAR1_DIMENSION_2.COLUMN_N
        STAR1_DIMENSION_N.COLUMN_1
        STAR1_DIMENSION_N.COLUMN_N
        STAR1_FACT.COLUMN_1
        STAR1_FACT.COLUMN_N

        STAR2_DIMENSION_1.COLUMN_1
        STAR2_DIMENSION_1.COLUMN_N
        STAR2_DIMENSION_2.COLUMN_1
        STAR2_DIMENSION_2.COLUMN_N
        STAR2_DIMENSION_N.COLUMN_1
        STAR2_DIMENSION_N.COLUMN_N
        STAR2_FACT.COLUMN_1
        STAR2_FACT.COLUMN_N
FROM
        STAR1_DIMENSION_1
        STAR1_DIMENSION_2
        STAR1_DIMENSION_N
        STAR1_FACT

        STAR2_DIMENSION_1
        STAR2_DIMENSION_2
        STAR2_DIMENSION_N
        STAR2_FACT
WHERE
        STAR1_DIMENSION_1.KEY_1 = STAR1_FACT.KEY1_COLUMN
        STAR1_DIMENSION_2.KEY_2 = STAR1_FACT.KEY2_COLUMN
        STAR1_DIMENSION_3.KEY_3 = STAR1_FACT.KEY3_COLUMN
        STAR1_DIMENSION_N.KEY_N = STAR1_FACT.KEYN_COLUMN

        STAR2_DIMENSION_1.KEY_1 = STAR2_FACT.KEY1_COLUMN
        STAR2_DIMENSION_2.KEY_2 = STAR2_FACT.KEY2_COLUMN
        STAR2_DIMENSION_3.KEY_3 = STAR2_FACT.KEY3_COLUMN
        STAR2_DIMENSION_N.KEY_N = STAR2_FACT.KEYN_COLUMN

        STAR1_DIMENSION_1.KEY_1 = STAR2_DIMENSION_1.KEY_1
        STAR1_DIMENSION_2.KEY_2 = STAR2_DIMENSION_2.KEY_2
        STAR1_DIMENSION_N.KEY_N = STAR2_DIMENSION_N.KEY_N

Other WHERE clause conditions
```

Star 1
All columns and facts fully qualified
By TABLENAME.COLUMNNAME

Star 2
All columns and facts fully qualified
By TABLENAME.COLUMNNAME

Star 1
All tables

Star 2
All tables

Star 1
All joins fully qualified
By
TABLENAME.COLUMN

Star 2
All joins fully qualified
By
TABLENAME.COLUMN

Cross Star
All joins fully qualified
By
TABLENAME.COLUMN

Other conditions
Fully qualified by
TABLENAME.COLUMNNAME