Design Tip #105 Snowflakes, Outriggers, and Bridges

By Margy
Ross

Students often blur the concepts of snowflakes, outriggers, and bridges. In this Design Tip, I'll try to reduce the confusion surrounding these embellishments to the standard dimensional model.

When a dimension table is snowflaked, the redundant many-to-one attributes are removed into separate dimension tables. For example, instead of collapsing hierarchical rollups such as brand and category into columns of a product dimension table, the attributes are stored in separate brand and category tables which are then linked to the product table. With snowflakes, the dimension tables are normalized to third normal form. A standard dimensional model often has 10 to 20 denormalized dimension tables surrounding the fact table in a single layer halo; this exact same data might easily be represented by 100 or more linked dimension tables in a snowflake schema.

We generally encourage you to handle many-to-one hierarchical relationships in a single dimension table rather than snowflaking. Snowflakes may appear optimal to an experienced OLTP data modeler, but they're suboptimal for DW/BI query performance. The linked snowflaked tables create complexity and confusion for users directly exposed to the table structures; even if users are buffered from the tables, snowflaking increases complexity for the optimizer which must link hundreds of tables together to resolve queries. Snowflakes also put burden on the ETL system to manage the keys linking the normalized tables which can become grossly complex when the linked hierarchical relationships are subject to change. While snowflaking may save some space by replacing repeated text strings with codes, the savings are negligible, especially in light of the price paid for the extra ETL burden and query complexity.

Outriggers are similar to snowflakes in that they're used for many-to-one relationships, however they're more limited. Outriggers are dimension tables joined to other dimension tables, but they're just one more layer removed from the fact table, rather than being fully normalized snowflakes. Outriggers are most frequently used when one standard dimension table is referenced in another dimension, such as a hire date attribute in the employee dimension table. If the users want to slice and-dice the hire date by non-standard calendar attributes, such as the fiscal year, then a date dimension table (with unique column labels such as Hire Date Fiscal Year) could serve as an outrigger to the employee dimension table joined on a date key.

Like many things in life, outriggers are acceptable in moderation, but they should be viewed as the exception rather than the rule. If outriggers are rampant in your dimensional model, it's time to return to the drawing board given the potentially negative impact on ease-of-use and query performance.

Bridge tables are used in two more complicated scenarios. The first is where a many-to-many relationship can't be resolved in the fact table itself (where M:M relationships are normally handled) because a single fact measurement is associated with multiple occurrences of a dimension, such as multiple customers associated with a single bank account balance. Placing a customer dimension key in the fact table would require the unnatural and unreasonable divvying of the balance amongst multiple customers, so a bridge table with dual keys to capture the many-to-many relationship between customers and accounts is used in conjunction with the measurement fact table. Bridge tables are also used to represent a ragged or variable depth hierarchical relationship which cannot be reasonably forced into a simpler fixed depth hierarchy of many-to-one attributes in a dimension table.

In these isolated situations, the bridge table comes to the rescue, albeit at a price. Sometimes bridges are used to capture the complete data relationships, but pseudo compromises, such as including the primary account holder or top rollup level as dimension attributes, help avoid paying the toll for navigating the bridge on every query.