Design Pattern – 005 Placeholders

Intent

This Design Pattern documents how to handle situations where there are mismatches with the source business keys, leading to values being not available in some cases. Due to the strict key lookups this would lead to errors in ETL, and this is where placeholders come in. This pattern assumes that source files are always processed against Hub tables first, including loading any transactional tables against the Hubs. In other words, all files that contain business keys are updated against the Hub tables first to eliminate any timing related business key lookup errors.

Motivation

This pattern focuses on processing data from dodgy sources that actually contains NULL business keys. When a business key actually is NULL this should be resolved to a placeholder (dummy Surrogate Key).

The reasoning behind this is to prevent overcomplicated error handling while loading data into the (raw) Data Vault; the goal is to load everything just as the source system provides it, while at the same time preventing losing any records while loading into the (raw) Data Vault.

Also known as

- -1 value.
- Unknown business keys.
- Late or early arriving data.

Applicability

This pattern is only applicable for loading data into the Integration Area tables.

Structure

The Enterprise Data Warehouse architecture specifies that 'hard' business rules are implemented on the way into the Data Warehouse (the process from the Staging Area into the Integration Area) whereas 'soft' business rules are implemented from the Integration Layer to the Interpretation Area and/or the Presentation Layer (on the way out).

Using placeholders is a 'hard' business rule because no-one can interpret the meaning of a NULL value. SQL can't deal with NULLS very well and because of this allowing NULL values increases the complexity of the queries against the Integration Area (potentially using Outer Joins). This is the reason why NULL values are remapped on the way into the Integration Area, and ultimately why this kind of business logic is allowed here.

For example, here are some reasons how NULL values can be presented instead of business keys:

1. The source declares them as optional Foreign Keys; for instance when 'X' is true, then the business key is populated. Otherwise the business key remains NULL.

2. The source declares them as required, but the declaration is broken or not enforced (there is an error in the source application that allows NULLS when it shouldn't).

Implementation guidelines

- NULL/unknown/undefined business key values can be mapped to various placeholder surrogate key values (-1 to -7 surrogate key values) with descriptions like 'Not Applicable', 'Unknown' or anything that fits the business key domain. The taxonomy usable for most situations is (not all values are applicable in all situations):
    1. Missing (-1): the root node and supertype of all 'missing' information, it encompasses:
        1. Missing value (-2): supertype of all missing values. Can be 'Unknown' or 'Not Applicable':
            1. Not Applicable (-3).
            2. Unknown (-4).
        2. Missing Attribute/Column (-5): supertype of all missing values due to missing attributes:
            1. Missing Source Attribute (Non recordable Source) (-6). Used when source fails to supply attribute/column
            2. Missing Target Attribute (Non recordable DWH Attribute) (-7). Used for temporal data that falls before the deployment of the attribute.
- Deciding between the various types of 'unknown' is a business question that is decided based on how the source database works.

Consequences

- The Hubs must be pre-populated with the placeholder values.
- ETL processes loading data into the Integration Area must automatically resolve NULL values to (potentially different) placeholders.
- Implementing a full taxonomy of potential unknown values as hard business rules must be weighed against extra complexity while loading Integration Area tables.

Related patterns

- Design Pattern 007 – Loading Hub tables.
- Design Pattern 008 – Loading Satellite tables.
- Design Pattern 009 – Loading Link tables.

Discussion items (not yet to be implemented or used until final)

None.