Database Object and Naming Conventions

Kirk Wilson

This is a sample document I created for a client setting up a brand new data warehouse on Snowflake. The primary data source was Viewpoint Vista, which is a construction-focused ERP that runs on a SQL Server backend. Alteryx was also used primarily for ETL in this scenario.

The client had minimal “traditional” BI experience, so I decided to stick with the basics and use a very traditional star schema model that would fit their needs (and relatively small data volumes) very well.

I created an automated script which could largely automate the process of generating ETL packages that makes these standards easy to adhere to. In my experience, human mistakes are by far the biggest problem with having standards that are consistently followed.

- General
    - Favor readability over brevity, and do not use abbreviations or contractions as part of object names
        - Example: Choose "AR_INVOICE" instead of "ARINV"
    - Only use industry standard acronyms, and only when the acronym is a more common term than its full counterpart or will be used frequently.
        - NOTE: This only applies when creating a brand new data model that does not directly reference existing objects. Vista for example heavily uses acronyms in column names ("ProjectMgr" for example), and due to the volume, industry-specific, and sometimes abstract nature of these abbreviations they will be left as-is.
        - Example: Choose "AP" instead of "ACCOUNTS_PAYABLES" since there will likely be multiple "AP"-related objects, it is used interchangeable with "Accounts Payables", and there is a significant shortening of the overall object name.
        - Example: Choose "AMOUNT" over "AMT" since "AMT" is not an official standard and is not a significant length savings over "AMOUNT".
    - Do not use hyphens or any other non-alphanumeric characters
    - Query formatting
        - The purpose of these standards is not to control every element of coding style. Elements such as tabs vs. spaces, commas at the end or start of a line, lowercase vs. uppercase vs. mixed case, etc. in common usage are left up to the individual author, with the assumption that general readability will still be maintained (i.e., use line breaks for a query of any significant length rather than spanning the query across a single line).
- Capitalization
    - Objects created in Snowflake will not be quoted. As a result, all object names will be case-insensitive.
- Spaces / Underscores
    - Underscores will be used consistently in all database object names to improve readability.
    - Since objects will not be quoted, spaces are not applicable for object names.
- Table Types
    - REF - Reference

        - Contains lookup data that by itself provides no value.
        - Generally does not contain a surrogate key to another table and instead leverages a natural key.
        - Examples:
            - Table that contains country code translations (CAD = Canada, MEX = Mexico, …).
            - Table that contains currency exchange rates over time.
    - DIM - Dimension
        - Contains descriptive attributes that can be used to further slice/categorize transactional data.
        - One Dimension table will generally relate to multiple Fact tables via a single surrogate key.
        - Generally does not contain metrics/measures that would be reported on.
        - All dimensions shall be conformed, meaning dimension tables can be shared across modules.
            - For example, there will be a single Customer table that can relate to both financial and project data (and all other applicable functional areas), as opposed to having separate "Financial Customers" and "Project Customers" tables.
        - All dimensions will be Type 1 by default, but a SCD architecture can be implemented on an as-needed basis.
    - FACT - Fact
        - Contains transactional data (either at a detail or aggregated level of granularity).
        - Generally contains at least one metric/measure (such as cost or quantity).
        - Generally relates to one or more dimension tables.
        - Types of FACT tables:
            - DFACT (Detail Fact)
                - Contains data that has not been aggregated in any way and is at its naturally lowest level of granularity.
            - AFACT (Aggregate Fact)
                - Contains data that has been aggregated in some way.
            - SFACT (Snapshot Fact)
                - Contains snapshots (data as it was at a specific point in time) of data taken from a DFACT or AFACT table. Includes timestamp of when the snap was taken.
            - BFACT (Bridge Fact)
                - Used as an intermediary table to handle a many-to-many relationship between other Fact tables.
    - MD - Metadata
        - For internal use only. Created on an as-needed basis and do not need to conform to any specific convention.
- Data Types
    - Numbers
        - Snowflake: NUMBER(38,s)
            - Always use 38 for the precision, and the scale should be the same as what is defined in the source system.
                - In Vista, reference NUMERIC_SCALE in INFORMATION_SCHEMA.COLUMNS

    - Characters
        - Snowflake: VARCHAR
            - Do not specify a length. Per the Snowflake documentation, there is no performance or storage difference between VARCHAR(1) and VARCHAR(16777216).
                - It is possible that there would be overhead within Alteryx if it pre-allocates memory based on the data types, but this will be evaluated at a later point in time.
    - Dates
        - Snowflake: DATETIME
- Table/View Names
    - DIM table names will not have a module prefix as they can typically be shared across tables/modules. They will be suffixed with the table type.
    - FACT table names will be prefixed with their source module and suffixed with the table type.
    - Metadata tables names will be prefixed with "MD". They will not have a required suffix.
    - View names will follow table naming conventions, except they will be suffixed with "VIEW"
    - Tables source from Vista will follow naming conventions stored in DDTH.Description.
    - Examples:
        - JCCH (Vista) would be named JC_COST_DETAIL_DFACT
        - CMCO (Vista) would be named COMPANY_DIM
        - A view used as the extracting interface for JCCH would be named JC_COST_DETAIL_VIEW
        - Metadata table that stores a list of source systems would be named MD_SOURCE_SYSTEMS
- Column Names
    - The first column in each table will be a generated surrogate key (populated by a database sequence). This column name will be the same as the table name, with only the suffix changing to "KEY"
        - Examples:
            - The primary key of JC_COST_DETAIL_DFACT will be JC_COST_DETAIL_KEY
            - The primary key of COMPANY_DIM will be COMPANY_KEY
    - Surrogate foreign keys will be used to join tables together, rather than relying on natural keys. In many instances natural keys are not obvious, especially for an outsider who might be wanting to run an ad-hoc query. In addition, a natural key can be comprised of multiple columns which again provides an obstacle to writing simple, user-friendly queries.
    - Since Vista uses case-sensitive camel case conventions, underscores will be inserted between words.
    - Examples:
        - "JobStatus" becomes "JOB_STATUS"
        - "PRLocalCode" becomes "PR_LOCAL_CODE"
- Metadata Columns
    - MD_NATURAL_KEY (VARCHAR)
        - Contains a string representing the record's natural key.
            - If a table has a compound natural key, use two colons :: as a delimiter.

        - Example: Assume a natural key is composed of COLUMN_1 and COLUMN_2, with values 'ABC' and 'DEF' respectively for a record. The value stored in MD_NATURAL_KEY would be 'ABC::DEF'.
    - MD_SOURCE_SYSTEM (NUMBER)
        - Numeric identifier relating to original source of data (Vista, Smartsheet, etc.). There will be a separate metadata table dedicated to storing all source system information.
    - MD_INSERT_DATE (DATETIME)
        - Timestamp when the record was first inserted into the target environment. This is static.
    - MD_UPDATE_DATE (DATETIME)
        - Timestamp when the record was last updated in the target environment.
- Table Relationships
    - All tables can be joined directly on keys using an inner join (except where an outer join is required for business logic).
    - Any foreign key column will be defaulted to -1. In addition, each table will have a "seed" record with a primary key value of -1 so that even if there is no relationship between two tables for a given record, an inner join can still be leveraged for simplicity.
    - Examples:
        - JCCD, JCCH, and CMCO would be joined together in the warehouse as follows:

```
SELECT ...
FROM
JC_COST_DETAIL_DFACT JCCD,
JC_COST_HEADER_DFACT JCCH,
COMPANY_DIM CMCO
WHERE 1=1
AND JCCD.JC_COST_HEADER_KEY = JCCH.JC_COST_HEADER_KEY
AND CMCO.COMPANY_KEY = JCCD.COMPANY_KEY
```