SSAS Cube Finishing

Version 1.0

Table of Contents

Database Framework

Surrogate and Natural Keys (pg. 25, 26, No DateTime as Key, Use Int, or BigInt <YYYYMMDD>)

Unknown Members, Key Errors, and NULLability (pg. 27)

Handle Unknown Members in ETL if possible. *Dimensions:* set UnknownMember and UnknownMemberName properties and NullProcessing (ErrorConfiguration) for the key attribute (leave default).

No NULLable Foreign Key Columns.

Use SQL Server Foreign Keys when testing and debugging, but not in Prod for faster ETL.

There should be no key errors.

Indexes in the Data Mart (pg.31)

Dims Primary Clustered Key of int, or bigint as the surrogate key

Unique, Non Clustered Indexes on the natural key to speed up the ETL for SCDs (natural key + SCD insert date)

Facts Primary Clustered Key of int, or bigint to identify a row when we need to check its value, or update it

If partitioned by date, the primary key would be the date + the integer surrogate key

If a column is a distinct count column then it may be useful in the clustered index so SSAS will order by

Other Other indexes for data mart reporting with reporting services may be required for performance tuning

Schemas and Naming Conventions (pg. 33, 47)

Schemas

Use schemas to divide the data warehouse into subject areas, or organizational structure (what if the organizational structure changes?) Subject areas typically include sales, accounts, warehouses, suppliers, personnel, emergency department, inpatient services, RAD, RTI, EmSuite, EmCare, Envision, and common.

Naming Conventions

Each entity on a data model represents a person, place, thing, or concept about which the company stores information. Each entity must be uniquely labeled in clear concise terms that are meaningful to the business user of that information. Naming conventions for entities are not as precise as those for data elements, but here are some general guidelines worth following.

The data entity name must be unique across all data models. If the information to be stored in the entity is about a student, there is only one repository in the logical model for student information -- the Student entity. If the chosen name already exists, either the two entities should be combined or one of them should be renamed.

Use singular nouns to name data entities. If occurrences of an entity represent employees, name the entity Employee, rather than the plural Employees; Student, rather than Students. This minimizes vagueness about what a single entity occurrence represents.

Create names as discretely and precisely as possible. For example, if the system scope addresses a specific type of student, such as only foreign students, use Foreign Student, rather than just Student. Never use phrases like data or information in an entity name; they do not adequately describe what a given entity represents.

Use names that reflect the business users' terminology. Where different groups of users have different terminology, select the more widely used or accepted term to name the entity. Include any synonyms to that terminology in the entity definition.

Keep the name as brief as possible. Use only a single noun where possible; under all circumstances, limit the name to 30 characters. Use alphabetic characters only; avoid using special characters like "/?!@#$%^&*()+-=',." or numbers.

Reflect the thing the entity represents, rather than any medium on which it is handled. For example, when naming an entity to describe the event oriented data associated with the receipt of a complaint from a student, use something like Student Complaint, rather than Student Complaint Letter.

BI developers are familiar with basic dimensional modeling concepts. They will expect to see dimensions and facts. Name your objects accordingly. Use a _FACT or _DIM suffix (some prefer prefix) to indicate the object orientation from a dimensional perspective. If you model an object as both (does that really happen often?), indicate that (_DIM_FACT). Use this same concept to indicate additional concepts such as aggregates (_AGG). Remove this type of naming convention in the cube and use friendly names.

Avoid using acronyms and abbreviations if possible. If your RDBMS has restrictions on object names, and you have to shorten names, seek report developers and business users advise on appropriate short names to use. If you can't do that, look to avoid cutting off long words, instead use shortening techniques that use letters from the beginning, middle and end of the word you seek to shorten. If you don't have enough space to spell out CATEGORY, consider using CATEG, instead of CAT.

Use a convenient and easy to follow naming convention for Primary and Foreign key column names. This is especially critical if you do not plan to enforce data integrity at the database level. Examples are Category_ID, EncounterID, or PatientID_SK. Be clear about surrogate keys and business keys. Examples are EncountedID_BK and EncounterID_SK. Remove this type of naming convention in the cube and use friendly names, or hide from the users.

Perhaps the most important of all is BE CONSISTENT!

Views versus the Data Source View (pg. 33-36)

Use SQL Views over making changes in the Data Source View (DSV) whenever possible.

Reasons for SQL Views:

- Change column names easier
- Perform Simple Calculations
- Reduce Exposed Columns, Grouping and Banding
- Provide Default Values
- Can expose a star schema for a complex relational model
- Views can be optimized and Quickly Updated
- Changes in the Cube (DSV) are Isolated and Hidden from the database and other cubes

Dimensions

Dimension Health Check all Dimensions (BIDS Helper)

Properties for all Attributes of Dimensions

Key Column, Name Column
Attribute Hierarchy Enabled* (Is a hierarchy created? False for email, phone #, social #)
Attribute Hierarchy Optimized State (If above* is True, do we create an index?)
Order By, Order by Attribute (Numeric Order Column in Dimension to use as a Key of an Attribute)
Attribute Hierarchy Ordered (If no order makes sense, set to False to save processing time)
Is Aggregatable, Default Member (If above* is True, True displays the All Member… always set True)

Dim Date (pg. 50)

Set dimension type property to Time.
Set the attributes type property to Years, Months, etc.
Define Key Columns, Name Columns, and Value Columns especially month (Year, Month collection key)
Set Attributes Order By to Key
Define Hierarchies: Year – Quarter – Month – Date, Year – Month – Date,
Year – Quarter -- Month – Day Number of Week – Date
Define Attribute Relationships

Create User Hierarchies (pg. 51)

Take 2 or more attribute hierarchies and combine them into a more complex drill path
Set the individual attribute hierarchies Visible property to False to reduce confusion
Can't put different levels of the same user hierarchy on different axis in a in a MDX query
See Dim Date example above

Configure Attribute Relationships (pg. 52, 67)

They do not have anything to do with how the dimension gets displayed to the end-user like hierarchies
Very important for query performance by building efficient indexes and making use of aggregations!
Use BIDS Helper "Visualize Attribute Lattice" tool to display an easy to understand diagram
The more long chains you see the better in attribute relationship diagrams
User hierarchies which have one-to-many relationships defined between the attributes that make up each level are called natural user hierarchies. User hierarchies that don't are unnatural user hierarchies (amber triangle warning due to lower query performance).
Relationship Type is Flexible if the relationship between any 2 members on 2 related attributes will change
Relationship Type is Rigid if the relationship between any 2 members on 2 related attributes will not change
Attribute Relationships allow you to model one-to-many relationships between attributes. They define functional dependencies between attributes. For example, City -> State and State -> Country where there is one city in the state and one state in the country. If the current city is Seattle, then we know you are in WA, USA. This is helpful in query performance and calculations in terms of selecting the correct query to execute.

Example Query: Select measures.sales on columns from Sales where (Customer.Name.Richard)
Example Queries SSAS can Execute: (measures.sales, Customer.Name.Richard, Customer.Gender.All, Customer.City.All, Customer.State.All, Customer.Country.All) or (measures.sales, Customer.Name.Richard, Customer.Gender.Male, Customer.City.Sammamish, Customer.State.Washington, Customer.Country.USA)

Grouping and Banding (pg. 61)

Grouping

Design into the dimensions themselves and don't use Analysis Services poop functionality
Group members on a long attribute hierarchy to make navigation easier for the end-user
Use a SQL Views (or named calculation) to create new columns using a CASE Statement
May want an additional column to provide a key for ordering

Banding

Create a dimension that acts as a way of grouping measure values on a fact table
Hardcode only the granularity of the banding into the fact table so that if a banding changes the fact won't need to be loaded. The cube only needs to be processed if the banding changes.

Slowly Changing Dimensions (pg. 18, 19)

Type I SCDs (pg.65) – No special design is needed in SSAS. Process Update will update any changes. If a change is to an attribute relationship that is rigid processing will fail.
Set Dimension Property MDXMissingMemberMode to Error from Ignore to alert if a query references a changed unique name that no longer exists.
Type II SCDs (pg. 67) – No explicit support for different versions of the same dimension in SSAS.
Use the business key, to group the many surrogate keys that are created for the same entity like customer.
Carefully define attribute relationships on Type II SCDs (pg. 67).
Include Member Status (hide from users) in the dimension to filter an attribute on current row (pg. 70)
For start and end dates use the same integer surrogate key as in the Date Dimension to provide the option for role playing dimensions based on the time dimension which then can be joined to the SCD on these columns with w referenced relationship to filter entities like customer to return members within a certain date range.
Type III SCDs (pg. 70) – Results in 2 different sets of attributes for the same entity like customer (current city and previous city). Group these different sets of attributes into separate folders using the AttributeHierarchyDisplayFolder property.
2012 Change Data Capture (this happens upstream in the ETL, but will it impact this?)

Junk Dimensions (pg. 71)

Groups of attributes that don't belong to specific dimensions.
Generally columns from fact tables that represent flags, or status indicators.

Degenerate (Fact) Dimensions (pg. 16, 17)

Columns on the Fact Table that we want to use for analysis but do not relate to any dimension.
They are almost always the cardinality of the fact table like Transaction Number in a Sales Data Mart.
Useful for Total Sales (Sold) in one (1) transaction measure. Useful for Drill Down to OLTP data.
Should only be displayed in specially designed reports that limit data set sizes.
Not for ad hoc navigation of sales. Sales Dim may include both degenerate attributes (Transaction Number) and standard attributes (POS Number)

Ragged Hierarchies (pg. 72)

Parent Child Hierarchies (pg. 72, Avoid Setting Is Aggregatable to False pg. 74, Avoid Using pg. 75)
Can't build aggregations at the intermediate levels if the parent child hierarchy.

- Build a regular hierarchy and use Hide Member If (pg. 75)
- Set HideMemberIf= OnlyChildWithParentName to hide members with the same name as their parent

Measures and Measure Groups

Measure Group Health Check all Measure Groups (BIDS Helper)

Properties for all Measure of Measure Groups (Fact Tables)

Aggregate Function (pg. 80, 83) – How the measure aggregates up through each hierarchy in the cube.

Sum – Most common, Values for this measure will be summed up.

Average – Divide a measure with the aggregate function Sum by one with aggregate function Count.

Count (Number of Rows with Row Binding, Number of Non-NULL Values with Column Binding)

Min / Max – Returns the minimum and maximum measure values.

Distinct Count – Number of distinct values in a column in a fact table. In its own measure group.

None – No aggregation takes place on the measure.

Semi-Additive Aggregation Types – Useful for snapshot data.

Average of Children – Average of all the values at the lowest level of granularity

First Child – First child at the lowest level of granularity

Last Child – Last child at the lowest level of granularity

First Non Empty – First child at the lowest level of granularity, that is not empty

Last Non Empty – Last child at the lowest level of granularity, that is not empty

Format String (pg. 80)

% - Value multiplied (*) by 100

#,#.00;#,#.00;0;\N\A – Displays the string NULL values as NA… does not affect performance like MDX

Be careful using Currency since it will change the currency symbol for the locale specified in the cube

Use a format string instead of currency… '$#,#.00'

http://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx

http://technet.microsoft.com/en-us/library/ms146084.aspx

Display Folders (pg. 81)

Folder names are case sensitive. You can make a measure appear under multiple folders by entering a ; delimited list of names as follows: Folder1;Folder2;Folder3 or Folder1;Folder2\Folder3

By Account (Chart of Accounts pg. 88, Not Often Used pg. 89)

Chart of Accounts Dim

Attribute to Flag which Members on the Main Hierarchy will use which Semi-Additive Aggregation Type

Select Following Properties

On the Dimension, Set Type Property to Accounts

On the Main Hierarchy, Set Type Property to Account

On the Account Type Attribute, Set Type Property to Account Type

Define How Each Account Type Should Be Aggregated

Dimension Calculations (pg. 89)

Define other ways to rollup measures (Aggregate Functions) besides using MDX… see below to Dimension Usage…

Unary Operators and Weights (pg. 90)

Control how members on a hierarchy (usually parent child) aggregate up to their parents

Define a column in dim table to hold operator and set the attribute's UnaryOperatorColumn property

Custom Member Formulas (pg. 91)

Assign and MDX expression to calculate the value for each member on the attribute hierarchy

Define a column in the dim table to hold the MDX and set the CustomRollUpColumn property

Non-Aggregatable Values / Measures (pg. 92, A Different Approach pg. 99)

Usually the result of some pre-aggregation

Build data warehouse from Additive Data even if volumes are much larger

Dimension Usage (pg. 94)

Define relationships between measure groups and dimensions

Dimension Measure Group Relationships (pg. 103, 123)

No Relationship	No relationship between the Measure Group and the Dim
Regular	Regular Many-To-1 relationship between Measure Group and Dim
Fact	Degenerate Dims are built directly from Fact columns
Referenced	Dim joins to a Measure Group through another Dim (Materialize is checked)
Many-To-Many	Bridge table (links 2 regular dims) to model a M:M relationship with Measure Group
Data Mining	Use to create a special kind of dim called a Data Mining Dim

Measure Groups from Dimension Tables (pg. 96)

Create a measure that counts the number of days in the currently selected time period

So, if you selected year on the time dim hierarchy, the measure would show the number of days in the year

New measure group from time dim containing a new measure with aggFunction Count

(counts number of days as the number of rows in the dim table)

Ave Daily Sales: Create real measures for Sales and Day Count, then a MDX Calculated Measure that divides the former by the later (Sales / Day Count).

Handling Different Dimensionality (pg. 97)

Different measure groups in a cube will almost always have different dims associated with them

If there are measure groups with identical dimensionality, consider combining them into one measure group

What happens when a dim is selected without a relationship to the measure group?

IgnoreUnrelatedDimensions = Fales – Displays a null value for all members

IgnoreUnrelatedDimensions = True – Repeats the value displayed at the root of the dim for every member on the hierarchy of the dim (default)

Handling Different Granularity (pg. 98)

Even when measure groups share the same dimensionality, they may not share the same granularity

When you have a fact table at a higher grain (sales at the quarter level), change the granularity attribute property of the relationship to choose the attribute from the dim you want to join instead (Quarter).

In this case use IgnoreUnrelatedDimensions = Fales, not the default since nulls are preferred by the user.

Linked Dimensions / Measure Groups (pg. 101)

Creating linked dims and measure groups allows you to share the same dims and measure groups across separate SSAS databases and the measure group across multiple cubes. (Run new linked object wizard)

These represent a static snapshot with no updates – there is no option to refresh a linked object.

Transactional Data (pg. 107)

Drill Through (pg. 109)

Actions and Drill Through Actions (pg. 109 - 111)

Drill Through Columns Order (pg. 113)

Drill Through and Calculated Members (pg. 116)

Drill Through Modeling (pg. 117)

Drill Through Using a Transaction Details Dimension (pg. 117)

Drill Through on Alternate Fact Table (pg. 120)

Many to Many Relationships (pg. 122 - 130)

Calculations (pg. 131)

- Query Scoped, Session Scoped, and Globally Scoped Calculated Member (pg. 313 - 132)
- Formatting Calculated Measures (pg. 151)
- Calculation Dimensions (pg. 152 - 155, Best Practices pg. 158, Can have more than one calculation dimension)
- Attribute Overwrite (pg. 156)
- Limitations of Calculated Members (pg. 157)
- Named Sets (pg. 161)
    - Regular
    - Dynamic

Query Performance Tuning (pg. 191)

- Partitioning Measure Groups (pg. 194-199, 269-276)
- Aggregations (pg. 200-205)
    - Aggregation Design
    - Build Manually and Edit Aggregations (pg. 210-213, BIDS Helper)
    - User Based Optimization (pg. 205-208)
    - Aggregation Design Issues (pg. 213, 214)
- Monitoring Partitions and Aggregations (pg. 208-210)
- Tuning MDX Calculations and Queries (pg. 215-222)
- Using Caching to Your Advantage (pg. 222-225)

Securing the Cube (pg. 227)

- Roles
    - Admin
- Dynamic Security

Next Steps

- KPIS
- Actions
- Perspectives
- Translations
- Processing the Cube (pg. 276-284, Errors pg. 284-286, SSIS pg. 286-288, Proactive Caching pg. 289)
- Currency Conversion (pg. 167)
- Monitor Cube Performance and Usage (pg. 295)