**Test Knowledge of BIDS for Developing Cubes**

1. Ask the candidate to explain all the steps they need to complete in BIDS to create and publish a cube from scratch. For simplicity sake, I usually ask them to assume they have a Kimball method data warehouse on one SQL Server that has 2 fact tables and 5 dimension tables.

Most candidates who claim to have SSAS experience can explain the life cycle of building a cube, but rarely can they actually explain the steps to build a cube correctly. Experienced users should talk about setting up the databse connection, creating a DSV, generating a cube, generating dimension tables or modifying dimension tables created by the cube, defining attribute relationships for dimensions, defining relationships in the cube between fact and dimension tables, deploying the cube, etc. Candidates should know the terminology inside and out.

1. If the candidate describes the top-line process for building cubes in BIDS, then drill into details about the DSV. What are named queries? What are advantages and disadvantages of named queries? Should you link directly to tables, views, or named queries? Do views have any advantages over direct links to tables?
2. Ask the candidate to describe in detail how they would add a new attribute to a dimension. Assume for simplicity sake that someone has already added the column to the underlying database table and you now need to adjust the cube definition and deploy the changes.
3. Ask the candidate how cubes are maintained from day to day. Ask about the differences between fully processing cubes and dimensions versus partially processing cubes. Ask about what happens if a customer cancels an order and how that should propagate through the data warehouse. See if the candidate talks about ledger-style transactions versus status changes and how this impacts processing the fact table. Ask about how partitions are used, how they are defined, when you should use them, and when you shouldn't use them.
4. Ask detailed questions about advantages and disadvantages of date dimensions, time dimensions, how the dimensions should be maintained to handle new dates, etc. The candidate should explain an automated method for maintaining dates except for holidays.
5. Ask how changes to the cube are tested before publishing the changes to end users. I once interviewed a candidate who answered most of the technical questions about how to build a cube in BIDS correctly, but then couldn't explain to me how to test the cube. The candidate simply said he would publish the changes and then his manager would take care of everything. When I asked how he would test drill through actions, slicing behavior, etc., it became clear that the "architect" had no idea how any of this actually worked.
6. Ask how the candidate troubleshoots performance issues. Good answers should talk about SQL Profiler, testing MDX queries directly in Management Studio, monitoring key perfmon statistics, redefining attribute relationships and cube relationships, loading data into cleansed tables instead of using raw source tables, isolating analysis services performance from other application or sql server services, etc.

**Test Knowledge of MDX**

1. Ask the candidate some basic MDX questions. Ask questions like "I have a cube called new\_cube and it has a products dimension and a orders fact table. Tell me roughly how you would filter this down to 3 orders." If the candidate can only explain how to do it in a GUI such as int Excel or SSRS, then ask some deeper questions about returning nulls, returning all records regardless of nulls, or returning non-null values.
2. Ask the candidate about when they actually code MDX versus just use a GUI. Ask about which tools the candidate used to interact with the data. If it is Excel, then ask if they have used the olap extensions or data mining extensions. Ask what they can see in SQL Server Mangement Studio. If it is Excel, then ask how they handled refreshing data between months without having to change parameters. If it is SSRS, then ask how they handled multivalue parameters or changing dates for subscriptions. If they did most of their work in Management Studio, then ask questions about syntax and different methods for limiting data to a subset of users, orders, or dates.

**Test Knowledge of Data Warehousing Design Principals**

1. Ask questions about Kimball method data warehouses, star schemas, snowflake schemas, degenerate dimensions, data dimensions, time dimensions, surrogate keys, etc.
2. Ask questions about SQL Server database design pricipals such as the differences between indexes, non-clustered indexes, clustered indexes, composite indexes, CTEs, table-value functions, looping over data, fizzbuzz test, creating and managing SQL Server Agent Jobs and schedules, how to troubleshoot slow-performing queries, etc. An excellent SSAS architect should be an expert SQL DBA from a data warehousing perspective. Don't ask questions about replication, log shipping, mirroring, clustering, etc., since this is usually outside of the pervue of data warehousing SQL DBAs.
3. Ask questions about SSIS. An excellent SSAS architect must understand how to build complex SSIS packages including importing a filtered list of changing files from a directory, pulling data in via data flows, explain how to use fast load options for bulk inserts, talk about script components as sources or transformations, etc.

At the end of all of this, you should be able to determine if the user is a SSAS architect, a wannabe SSAS architect who has lots of SQL DBA data warehousing architecture experience, a SSAS report writer in Excel, SSRS, or other BI platform, a report writer who doesn't really understand what's happening under the covers, a newbie, or a faker. Keep in mind that a lot of really good data warehouse architects don't have much SSAS experience. If you are looking for an experienced SSAS architect, then they basically have to be able to do the entire Microsoft BI stack. Anyone else fits into some other category.

----------------------

**Problem**

Measures and KPIs are very important aspects of an OLAP/SSAS solution from an end user standpoint, and in general, an important aspect of any Business Intelligence application. Storage is another important aspect of SSAS from an engineering standpoint. Hence it is essential to have a fair understanding of these aspects.  Check out these SQL Server Analysis Services (SSAS) interview questions.

**Solution**

In the [first](http://www.mssqltips.com/sqlservertip/2637/ssas-interview-questions-part-i-questions-on-basic-concepts-data-sources-and-data-source-views/), [second](http://www.mssqltips.com/sqlservertip/2662/sql-server-analysis-services-interview-questions-part-ii--dimensions/), and [third](http://www.mssqltips.com/sqlservertip/2683/sql-server-analysis-services-interview-questions-on-dimensions-hierarchies-and-properties/) tips of this series, I have covered questions on basic concepts, data sources, data source views, dimensions, dimension types, and components associated with dimensions like attributes, hierarchies, and some of the dimension properties. In this fourth tip, I will be covering some of the questions on measures, KPIs, actions, partitions, storage modes, etc. within [SQL Server Analysis Services (SSAS)](http://www.mssqltips.com/sql-server-tip-category/28/analysis-services/).

**What are Measures and Measure Groups? What is the difference between them?**

A Measure is any numeric quantity/value that represents a metric aligned to an organization's goals. This is the value which the business users are interested in, and are interested in viewing these values from different angles and different granularity levels. A measure is also commonly called a fact. The term "measures" and "facts" are used interchangeably.

A Measure Group is a collection/group of measures which belong to the same underlying fact table. In SSAS, typically each Measure Group is tied to each one of the underlying fact tables.

A Measure is single numeric value whereas a Measure Group is a collection of measures.

**What are the different types of Measures? Explain each one of them with an example.**

Below are the most common types of measures/facts:

* **Fully Additive Facts:**These are facts which can be added across all the associated dimensions. For example, sales amount is a fact which can be summed across different dimensions like customer, geography, date, product, and so on.
* **Semi-Additive Facts:**These are facts which can be added across only few dimensions rather than all dimensions. For example, bank balance is a fact which can be summed across the customer dimension (i.e. the total balance of all the customers in a bank at the end of a particular quarter).  However, the same fact cannot be added across the date dimension (i.e. the total balance at the end of quarter 1 is $X million and $Y million at the end of quarter 2, so at the end of quarter 2, the total balance is only $Y million and not $X+$Y).
* **Non-Additive Facts:**These are facts which cannot be added across any of the dimensions in the cube. For example, profit margin is a fact which cannot be added across any of the dimensions. For example, if product P1 has a 10% profit and product P2 has a 10% profit then your net profit is still 10% and not 20%.  We cannot add profit margins across product dimensions. Similarly, if your profit margin is 10% on Day1 and 10% on Day2, then your net Profit Margin at the end of Day2 is still 10% and not 20%.
* **Derived Facts:**Derived facts are the facts which are calculated from one or more base facts, often by applying additional criteria. Often these are not stored in the cube and are calculated on the fly at the time of accessing them. For example, profit margin.
* **Factless Facts:**A factless fact table is one which only has references (Foreign Keys) to the dimensions and it does not contain any measures. These types of fact tables are often used to capture events (valid transactions without a net change in a measure value). For example, a balance enquiry at an automated teller machine (ATM). Though there is no change in the account balance, this transaction is still important for analysis purposes.
* **Textual Facts:**Textual facts refer to the textual data present in the fact table, which is not measurable (non-additive), but is important for analysis purposes. For example, codes (i.e. product codes), flags (i.e. status flag), etc.

**What is the purpose of Dimension Usage settings? Explain different types of relationships between Facts and Dimensions.**

The Dimension Usage tab in the Cube Designer in [SQL Server Business Intelligence Development Studio](http://www.mssqltips.com/sqlservertutorial/204/business-intelligence-development-studio-bids/) defines the relationship between a Cube Dimension and a Measure Group (s). A Dimension which is related to one of more Measure Groups, directly/indirectly, is called as a Cube Dimension. A Cube Dimension is an instance of a database Dimension as explained in the [previous tip](http://www.mssqltips.com/sqlservertip/2683/sql-server-analysis-services-interview-questions-on-dimensions-hierarchies-and-properties/).

Following are the four different types of relationships between a Cube Dimension and a Measure Group:

* **Regular:**In a Regular relationship, primary key column of a dimension is directly connected to the fact table. This type of relationship is similar to the relationship between a dimension and a fact in a [Star Schema](http://www.mssqltips.com/sqlservertutorial/2004/creating-a-star-schema-using-a-data-source-view/), and it can be based on either the physical primary key-foreign key relationship in the underlying relational database or the logical primary key-foreign key relationship defined in the Data Source View.
* **Referenced:**In a Referenced relationship, primary key columns of a dimension is indirectly connected to the fact table through a key column in the intermediate dimension table. This type of relationship is similar to the indirect relationship between a dimension and a fact, through an intermediate dimension, in a Snowflake Schema.
* **Fact:**In a Fact relationship, the dimension table and the fact table are one and the same. Basically a [Fact Dimension or Degenerate Dimension](http://www.mssqltips.com/sqlservertip/2662/sql-server-analysis-services-interview-questions-part-ii--dimensions/)is created using one or more columns from the fact table and this degenerate dimension is used while defining/establishing the relationship in case of a fact relationship.
* **Many-to-Many:**In a Many-to-Many relationship, a dimension is indirectly connected to a Measure Group through an intermediate fact table which joins with the dimension table. It is analogous to a scenario, where one project can have multiple project managers and one project manager can manage multiple projects.

**What are Calculated Members? How do they differ from Measures?**

[Calculated Members](http://www.mssqltips.com/sqlservertutorial/2013/developing-a-calculated-measure/) are members of a measure group and are defined based on a combination of one or more base measures, arithmetic/conditional operators, numeric values, and functions, etc. For example, profit is a calculated member/calculate measure, which is defined based on various base measures like selling price, cost, price, tax amount, freight amount, etc.

The value of a measure (base measure) is stored in a cube as part of the cube processing process. Whereas the value of a calculated member/measure is calculated on the fly in response to a user request and only the definition is stored in the cube.

**What are Named Sets? What are the two types of Named Sets?**

A [Named Set](http://www.mssqltips.com/sqlservertutorial/2014/developing-named-sets/) is a set of dimension members (usually a subset of dimension members) and is defined using MDX (a Multidimensional Expression). Often Named Sets are defined for improved usability by the end users and client applications. Apart from that, they can also be used for various calculations at the cube level. Similar to calculated members/measures, named sets are defined using a combination of cube/dimension data, arithmetic operators, numeric values, functions, etc. Some of the examples of Named Sets are top 50 customers, top 10 products, top 5 students, etc.

Named Sets are of two types: Static Named Sets and Dynamic Named Sets.

Static Named Sets, when defined in cube, are evaluated during cube processing process. Dynamic Named Sets are evaluated each time the query is invoked by the user.

**What are KPIs? What are the different properties associated with a KPI?**

KPI stands for [Key Performance Indicator](http://www.mssqltips.com/sqlservertip/2398/ssrs-2008-r2-kpis-with-bullet-graphs/). A KPI is a measure of an organization's performance in a pre-defined area of interest. KPIs are defined to align with the pre-defined organizational goals and help the business decision makers gain insights into their business performance.

Often KPIs have the following five commonly used properties:

* **Name:**Indicates the name of the Key Performance Indicator.
* **Actual/Value:**Indicates the actual value of a measure pre-defined to align with organizational goals.
* **Target/Goal:**Indicates the target value (i.e. goal) of a measure pre-defined to align with organizational goals.
* **Status:**It is a numeric value and indicates the status of the KPI like performance is better than expected, performance is as expected, performance is not as expected, performance is much lower than expected, etc.
* **Trend:**It is a numeric value and indicates the KPIs trend like performance is constant over a period of time, performance is improving over a period of time, performance is degrading over a period of time, etc.

Apart from the above listed properties, most of the times, KPIs contain the following two optional properties:

* **Status Indicator:**It is a graphical Indicator used to visually display the status of a KPI. Usually colors like red, yellow, and green are used or even other graphics like smiley or unhappy faces.
* **Trend Indicator:**It is a graphical indicator used to visually display the trend of a KPI. Usually up arrow, right arrow, and down arrow are used.

**What are Actions in SSAS? What are the different types of Actions in SQL Server Analysis Services?**

Actions in SSAS allow us to extend the cube functionality and enable the users to interact with the cube. An Action in simple terms is basically an event, which can be initiated by a user/application and it can take various forms depending upon the type of Action defined.

Actions are primarily of following three types:

* **Drillthrough Actions:**A [Drillthrough Action](http://www.mssqltips.com/sqlservertip/2491/enabling-drillthrough-in-analysis-services/)retrieves the detail level information associated with the cube data based on which the Drillthrough Action is defined.
* **Reporting Actions:**A Reporting Action retrieves an SSRS report which is associated with the cube data. The command which invokes the SSRS report contains the report URL along with the report parameters.
* **Standard Actions:**A Standard Action retrieves the action element associated with the cube data. Standard actions are further categorized into 5 different subcategories and the action element varies for each of these subcategories. The following are the types of Standard Actions:
  + *Dataset Action:*Returns a dataset to the client application and the action content is an MDX expression.
  + *Proprietary Action:*Performs an operation as defined by the client application. The action content for this type of action is specific to the calling client application and the client application is responsible for interpreting the meaning of the Action.
  + *Rowset Action:*A Rowset Action returns a Rowset to the client application. The action content is a command to retrieve the data.
  + *Statement Action:*The action content for this type of Action is an OLE DB command and it returns a command string to the client application.
  + *URL Action:*The Action Content for this type of action is an URL and it returns a URL to the client application which can be opened usually in a web browser. This is the default action.

**What are partitions in cubes? How do they different from table partitions at a SQL Server database level?**

A partition is physical storage space which contains either all or a portion of measure group data. Each measure group in SSAS has one partition by default.

A partition can be either bound to a table in the underlying relational database or a query pointing to the table(s) in the underlying database and has filters in it.

In terms of storage, cube partitions in SSAS and [table partitions](http://www.mssqltips.com/sql-server-tip-category/65/partitioning/) in a database are similar. Both these types of partitions are used to improve the performance. However, partitions in SSAS offer additional benefits including:

* Each partition can be processed separately (i.e. a measure group can be split across multiple partitions, for example one partition for each year). Only the partitions in which data has been modified can be processed thereby improving the processing time of the cube.
* Partitions provide improved manageability by allowing us to define storage mode, aggregation design, etc. at the partition level and these settings can vary between different partitions belonging to the same measure group.

**What are the different Storage Modes supported by Cube Partitions?**

There are primarily two types of data in SSAS: summary and detail data. Based on the approach used to store each of these two types of data, there are three standard storage modes supported by partitions:

* **ROLAP:**ROLAP stands for Real Time Online Analytical Processing. In this storage mode, summary data is stored in the relational data warehouse and detail data is stored in the relational database. This storage mode offers low latency, but it requires large storage space as well as slower processing and query response times.
* **MOLAP:**MOLAP stands for Multidimensional Online Analytical Processing. In this storage mode, both summary and detail data is stored on the OLAP server (multidimensional storage). This storage mode offers faster query response and processing times, but offers a high latency and requires average amount of storage space. This storage mode leads to duplication of data as the detail data is present in both the relational as well as the multidimensional storage.
* **HOLAP:**HOLAP stands for Hybrid Online Analytical Processing. This storage mode is a combination of ROLAP and MOLAP storage modes. In this storage mode, summary data is stored in OLAP server (Multidimensional storage) and detail data is stored in the relational data warehouse. This storage mode offers optimal storage space, query response time, latency and fast processing times.

There are different variations of these Standard Storage Modes. Visit [this msdn article](http://msdn.microsoft.com/en-us/library/ms175646(v=sql.100).aspx) for more details.

**What is proactive caching in SQL Server Analysis Services?**

Proactive caching is an advanced feature in SSAS and it enables a cube to reflect the most recent data present in the underlying database by automatically refreshing the cube based on the predefined settings. This feature allows the users to view the data in near real-time.

Proactive caching can be configured to refresh the cache (MOLAP cache) either on a pre-defined schedule or in response to an event (change in the data) from the underlying relational database. Proactive caching settings also determine whether the data is queried from the underlying relational database (ROLAP) or is read from the outdated MOLAP cache, while the MOLAP cache is rebuilt.

Proactive caching helps in minimizing latency and achieve high performance.

----------------------------------