A lot of analysis of OLAP data is financial in nature and lot of financial analysis turns out to be time based.

We want calculations such as rolling averages, year over year change, quarter to date attainment, and so on

For Example:- On February 15, we want to add the data from January 1 through today. On March 29, it's January 1 through March 29. But on April 1, we just use the data from April.

A painful solution is to write a query that determines the current quarter, determines the first date of the current quarter, and then uses that to restrict the data in a subquery. However, you have to consider that this calculation has to be run for every value or set of values in the report.

OLAP engines provide a time-based analysis capability that makes these types of reports easy.

First, there is the hierarchical aspect of a time dimension: years, quarters,months, days. In addition, the ability to have multiple hierarchies on a single set of dimension members comes into use here; Like, you can have a hierarchy of year-month-day coexist with year-week-day.

More important, you can have multiple calendars: a standard calendar, in which the year runs from January 1 to December 31, and a fiscal calendar, which runs from the start of the fiscal year to the end (each with halves, quarters, weeks, and so forth).

At the root of everything we're trying to do are measures, or the numbers we're trying to analyze.

As stated earlier, generally measures are numbers—dollars, days, counts, and so forth. We are looking at values that we want to combine in different ways (most commonly by adding):

• How many bananas do we sell on weekdays vs. weekends?

• What color car sells best in summer months?

• What is the trend in sales (dollars) through the year, broken down by sales district?

• How many tons of grain do our livestock eat per month in the winter?

In each of these cases, we'll have individual records of sales or consumption, which we can combine. We'll then create dimensions and select members of those dimensions to answer the question.

- At the root of everything we're trying to do are measures, or the numbers we're trying to analyze.
- As stated earlier, generally measures are numbers—dollars, days, counts, and so forth.

We are looking at values that we want to combine in different ways (most commonly by adding) like:-

- How many bananas do we sell on weekdays vs. weekends?
- What color car sells best in summer months?
- What is the trend in sales (dollars) through the year, broken down by sales district?
- How many tons of grain do our livestock eat per month in the winter?

- In each of these cases, we'll have individual records of sales or consumption, which we can combine. We'll then create dimensions and select members of those dimensions to answer the question.

OLAP offers several ways of aggregating the numerical measures in our cube. But first we want to

designate how to aggregate the data—either additive, nonadditive, or semiadditive measures.

Additive

An additive measure can be aggregated along any dimension associated with the measure. When working with our sales measure, the sales figures are added together whether we use the date dimension, region, or product. Additive measures can be added or counted (and the counts can be added).

Semiadditive

A semiadditive measure can be aggregated along some dimensions but not others. The simplest example is an inventory, which can be added across warehouses, districts, and even products. However, you can't add inventory across time; if I have 1,100 widgets in stock in September, and then (after selling 200 widgets) I have 900 widgets in October, that doesn't mean I have 2,000 widgets (1,100 + 900).

Nonadditive

Finally, a nonadditive measure cannot be aggregated along any dimension. It must be calculated independently for every set of data. A distinct count is an example of a nonadditive measure.

The Unified Dimensional Model

A major underlying concept in Analysis Services is the unified dimensional model, or UDM.

We have a staging database (for scrubbing the data), a data warehouse (for aggregating the normalized data), data marts (for severing the data into more manageable chunks), and finally our OLAP store.

SSAS is designed to conceptually unify as much of Figure 3-2 as possible into a single-dimensional model, and as a result make an OLAP solution easier to create and maintain. Part of what makes this possible is the data source view (DSV), which is covered in Chapter 5. The DSV makes it possible to create a "virtual view," collating tables from numerous data sources. Using a DSV, a developer can create multiple cubes to address the various business scenarios necessary in a business intelligence solution.

SSAS is designed to make an OLAP solution easier to create and maintain, using Data Source View(DSV).

The DSV makes it possible to create a "virtual view," collating tables from numerous data sources. Using a DSV, a developer can create multiple cubes to address the various business scenarios necessary in a business intelligence solution.

LOgical Architecture:-

A single server can run multiple instances of Analysis Services, just as it can run several instances of the SQL Server relational engine. (You connect to an Analysis Services instance by using the same syntax: [server name] \ [instance name].) Within each instance is a server object that acts as the container for the objects within.

Each server object can have multiple database objects. A database object consists of all the objects you see in an Analysis Services solution in BIDS (more on that later). The minimum set of objects you need in a database object is a dimension, a measure group, and a partition (forming a cube).

We have grouped the objects in a database into three rough groups:

OLAP objects: Consisting of cubes, data sources, data source views, and dimensions, these are the fundamental objects that we use to build an OLAP solution.

Data-mining objects: This is pretty much the Mining Structure collection and the subordinate object hierarchy. A mining structure contains one or more MiningModel objects, as well as the columns and bindings necessary to map a mining model to the data source view.

Helper objects: Something of an "everything else" catchall. The helper objects consist of a collection of Assembly objects, DatabasePermission objects, and Role objects for managing security. An Assembly object represents a .NET assembly installed in the database.

XMLA

XML for Analysis (XMLA) was introduced by Microsoft in 2000 as a standard transport for querying OLAP engines. In 2001, Microsoft and Hyperion joined together to form the XMLA Council to maintain the standard. Today more than 25 companies follow the XMLA standard.

XMLA is a SOAP-based API (because it doesn't necessarily travel over HTTP, it's not a web service).Fundamentally, XMLA consists of just two methods: discover and execute. All results are returned in XML. Queries are sent via the execute method; the query language is not defined by the XMLA standard.

As I've mentioned previously, SQL Server Analysis Services runs as a single Windows service. The service executable is msmdsrv.exe, the display name (instance name) is SQL Server Analysis Services, and the service name is MSSQLServerOLAPService.

The default path to the executable is as follows:

$Program Files\Microsoft SQL Server\MSAS10.MSSQLSERVER\OLAP\bin

That service has an XMLA listener that handles all communications between the SSAS service and

external applications. The XMLA listener defaults to port 2383, and can be changed either during setup or from SQL Server Management Studio (SSMS). The location of database data files can also be changed in SSMS.

If you've ever had to root around the SQL Server file system, there's some great news with SQL Server 2008. With previous versions of SQL Server, folders for additional services (Analysis Services, Reporting Services, Integration Services) were simply added to the Microsoft SQL Server folder with incrementing suffixes. You would have to open each folder to find the one you were looking for.

You may ask, "What do these object models do for me?" In SQL Server Analysis Services, you can have stored procedures to provide functions that implement business rules or requirements more complex than perhaps SSAS can easily accomplish. Perhaps you need to run a query that calls to a web service and then retrieves a data set from a relational database based on the results of that query. You could create a stored procedure that accepts parameters and returns a data set, and then call that procedure from MDX in a KPI-bound or a calculated measure.