ABC of Data Warehousing and Business Intelligence

By Rajesh Kumar V.S
Aroha Technologies
Bangalore.

Introduction

This book is aimed to enhance your knowledge on Data Warehousing subject.

By reading this book

> You can answer why we need a data warehouse
> Where it fits in the business / organization
> How effectively it is used in an organization
> When should we provide DSS solution to customers?
> What components make this Data Warehouse?
> What are the different tools available in the market in order to create Data Warehouse etc?

This book also will give an idea and the differences between transactional system and Data Warehousing system. After reading this book, you should be able to explain what Data warehousing is all about, different phases with in the system and its general architecture.

First a problem arises, then we provide a solution, if you have more than one solution you will list the advantages and disadvantages and pick the best one based on the scenario. This book explains things pretty much the same way i.e. provide a problem, provide a solution and link the concept / logic in different phases of DW. We call this type of learning as ***Scenario based learning*** which helps you to correlate things when you are attending interviews or in a meeting with your customers.

If you want to excel in Data Warehouse area you should possess good RDBMS concepts and SQL knowledge. If you are a Business Analyst / business marketing executive just concentrate on the points where you can discuss with your customer.

Good luck in reading ABC of Data warehousing and Business Intelligence.

Before we begin

Before we get into Data warehousing, we need to know about the following fundamentals.

What is Data?
What is the Database?
What is RDBMS?
What is a Data Model?
Why Normalization?
Why De-normalization?
What is OLTP System?
Application layer, Data Layer and Reporting Layer

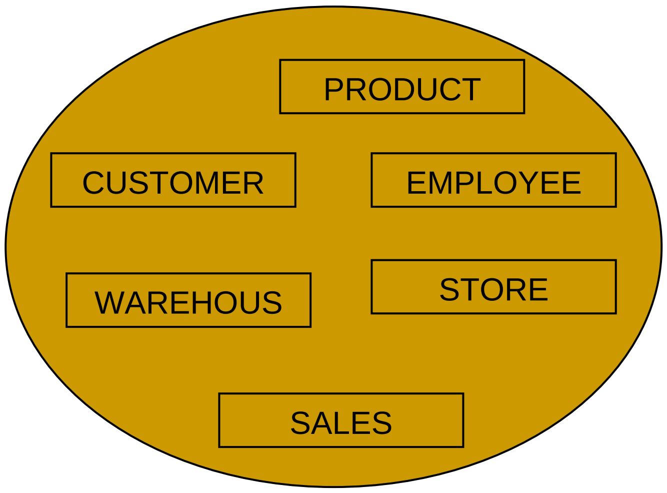
Data

Data can be defined as, any information which helps to run your business. In an HR application data is the employee, his salary information, employee name and the location where he/she works in the organization. In RDBMS, DATA is stored in the form of rows and tables. Following table contains the department info we see in an organization.

DepartmentNo	DepartmentName	Location	Country	Geo
1000	Development	Bangalore	India	APAC
1001	R&D	New York	USA	Americas
1002	Sales and Mktg	Boston	USA	Americas
1003	HR	Bangalore	India	APAC

Database

Software which is used to maintain the data an organization has. We can have multiple tables which contains many rows. Databases organize the data in such a way that it is very easy to maintain. You can visualize the database is nothing but a place with some tables with in like product, employee, warehouse information etc which is used to store the data for an organization.



RDBMS

There are three different types of Databases in the market. They are RDBMS, NDBMS and HDBMS.

RDBMS stands for Relational Database Management system; it works based on the set theory, entities and its relationship. An entity is a real world thing, when you define a business, you deal with customer, products, sales etc, and these are called entities. All the entities have its own properties which we call as attributes / columns in the database. In employee entity (Table) we store all the information about employees (Name, Address, salary, manager, etc). In a business, we deal with more entities which gets connected in various ways, per example one customer can have multiple accounts in a bank. So, we related multiple entities together in order to run the business.

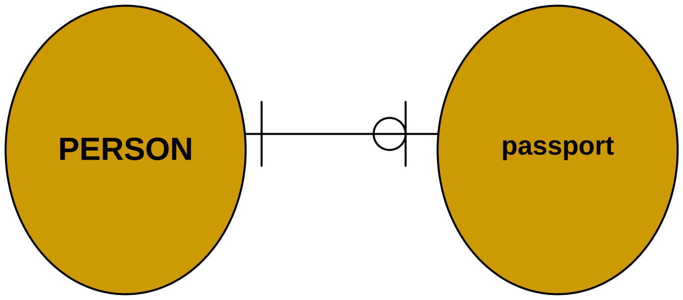
There are only three types of relationships existing between two entities

- One to One
- One to Many
- Many to Many

Only in RDBMS we can have many to many relationships between two entities.

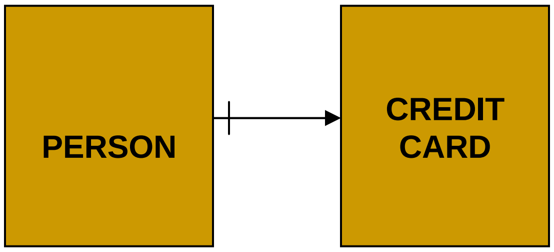
<u>One to One</u>

Example of one to one relationship is a person and passport entity. A person has his own properties like First name, last name, middle name, PAN number, Father name, Mother's name, date of birth etc. Passport contains the passport number, person name, issue date, expiry date, place of issue etc. What makes it one to one relationship is, one person can have only one passport and one passport can be shared by only one person (for the country which does not support dual citizenship as an example).



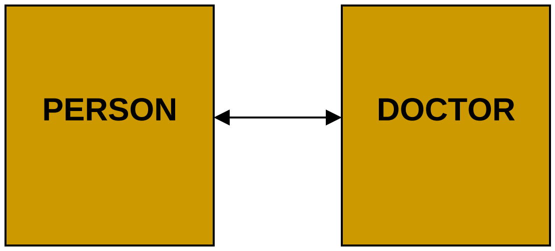
In the above example, one person may or may not have the passport, for example not every person must have the passport. So we must a have person to have a passport. We refer it as one to zero or one

<u>One to Many</u>
Example of one to many relationships is a person and credit cards. One person can have multiple credit cards from various banks where as one credit card can be associated to only one person. Always you have to make sure you read the relationship from both the aspects not just from one side and make sure you identify the correct relationship.



These relationships are defined by business. We will take another example of Department and Employee entities. If business decides one person can work in only one department then the relationship between Department and Employee is one to many. If business decides one person can work in multiple departments then this relationship cannot be one to many. So bottom line is based on how business runs, we model our data base to hold the same.

<u>Many to Many relationships</u>
Example for M:M relationship is a person and a doctor. One person can visit many doctors and one doctor can attend to many persons.



RDBMS provides us the flexibility to change the model as business changes the way it works. It is a multi-user environment with greater security for the data it stores and gives the mechanism to back and restore the data.

Data Model

It's a pictorial representation of the entities, relationships between entities and the attributes which reflects how the data is stored to run the business. In software development life cycle, after the requirement gathering phase, designing the data model is done. As a part of the project, we develop conceptual model, logical model and physical model.

Conceptual Model

It is a very high level view (bird's view) of how the business runs. We cross verify the model with the business to make sure we are on the right track. In this phase, we identify most of the entities which are required by business. You will see lot of many to many relationships between the identified entities.

Logical Model

In this model we actually complete the model with all the details of entities including the attributes, domains and its relationships. We resolve all the many to many relationships using associative tables in this phase. You can completely visualize the data movement with in the business using this model. Every project must have a logical model even if the project is very small.

Physical Model

We create the tables in their respective databases. Here DBA will take a call on identifying certain changes to logical model to show better performance while running the application. DBA's will be good in the respective databases and its utilities so that they can take a call on how best they can deploy the data model. At the logical level we call entities and attributes where as at physical level we call it as tables and columns.

Why Normalization?

Normalization helps to remove redundant data resulting in using less storage space which in turn makes our transaction to run faster.
For example, let's take an employee's information. We decided to have a table called

employee with the department name in the same table, say we have 100 employees who work in department called **SUPPORT,** business decides to change the department name as **SOFTSUPPORT,** and in this case we have to change 100 records. If we apply Normalization principles as the same value repeats we could have built a table called **department** which holds the name. In that case we would have updated only one record. This way we could have speed up the transaction we do. So, by splitting the tables to smaller, the inserts, updates and deletes run faster where as select statements run slower as we join many tables (this is the disadvantage).

OLTP System

Online Transaction Processing, Example of OLTP systems are any customer support applications where we are online with the representative and the transaction based on your requirement gets completed, another classic example is ATM transactions, railway reservation, airline reservation, flower shop website where you place orders etc.
The transactions you are trying to accomplish should happen ASAP. In OLTP system we cannot wait for 5 or 7 minutes to complete the transaction. The transaction should complete with in seconds. Usually when there is a requirement to stream line the business line then we start creating the OLTP system. First figure out what steps you manually follow to complete the task and convert that as an application. With out OLTP system the business won't run in a more effective way. To stream line the business we create OLTP applications.

Application Layer, Data Layer and Reporting Layer

Usually OLTP systems are used by operational resources. Operational resources use the screens / windows to interact with data. Say, if we are developing shipment entry screen which is used in courier services, the screen will be used by the operational resource (end user) and the data gets stored in the database for further processing.

Data Layer is nothing but the database where we store the incoming data to the application. The screen which is part of application layer interacts with the data layer. The data layer some time can be of any format; some times it can be a flat file, XML file or an RDBMS. Usually many organizations consider RDBMS as the data layer because of various advantages.

Reporting layer is used to generate reports for the application. Typical OLTP reporting tools are Actuate, Reporting Services, Crystal reports etc. Some times, using the screens they display the reports instead of developing the reports using the tool.

Problem Statement

In order to understand why we need a data warehouse we need to know, when and how business or yourself will suggest a data warehouse, what are the advantages of data warehouse etc., In order to make it very easy we will take an example to explain the same.

We will take a company which deals with distribution of goods (Electronic) to various customers as an example.

Company XYZ starts the distribution of electronics items from company 'ZOOM Computers' which manufactures the computers and electronic gadgets. The company 'XYZ Distributors' started with 10 employees with a turn around of say 500K $. At this stage the management had an idea of where the money spent and how the funds are utilized in the finger tips. At this stage they have only two applications which keep track of supplier and customer information.

As the time passes say about 5 years, the same XYZ Distributors deals with different electronic items (2000 items) from different suppliers (150). As the business grow, the number of OLTP applications now currently stands at 75 with an employee strength of 3000 and the turn over is over 750 million $. At this stage usually it's very difficult to pull the information from various source systems to answer some of the business questions. We can pull but the turn around time more (some times it may take weeks as well). IT team needs very specialized skills to make this happen and as the number of ad hoc requirements become more from management side, it's very difficult to answer those with in the period they expect. Some times IT team pulls data from oracle, Sybase, SQL Server and push the data in to a common database to integrate the data and run reports. In an average this type of work takes any where between 1 to 7 days some times even more which is based on the complexity of the requirement.

This is the time where we have to think about creating a data repository which consolidates the data from various applications in one place so that management can get 360 degree of the corporate view.

In the above scenario, say we have order management system, where we deal with orders placed by customers, rejections, canceling orders, we have SMG related activist such as campaign management, HR application to take care of employees, movement of them to different grades, salary information etc.

Management wants to know who are the customers who usually reject the orders they placed, because as a company we are spending our profits as a considerable amount. Want to compare which customer grades places the cancelled orders.

Management wants to know the percentage of income by products. Management wants to know who is the best sales person in a week, month, quarter and year.
Management wants to know if we outsource this operation what is the impact on the revenue.

Management wants to know the least profit making regions and the amount we spend on the same so that they can take a decision whether to continue with the region or how can we improve the business by promoting our products in the region.

To answer most of the management questions, we have to integrate the data across multiple source systems to make some sense of the data.

If management wants see the trend analysis from last three years if OLTP system is storing only 6 months to 1 year then we wont be able to do the analysis

In order to address above mentioned requirements in the industry we create a data warehouse or what we call as DSS (Decision Support System)

Even in OLTP we take decision. Example: You are a customer to a bank, you have 5K as the balance, say you are trying to with draw 10K in this scenario teller takes a decision not to give the money as the balance is less than what you requested for.
So, what's the different?

When you take a decision in an OLTP application, the impact to the business is almost zero. When you take a decision using Data Warehousing application the impact to the business is fairly large. Example, business identified certain product is not moving fast and more over it generates less than .05% of profit of the total revenue. In this case management can decide to continue with the product or not.

Bottom line is, data warehouse is a place to do analysis for an organization where as OLTP system is mainly to run the business.

With out DW you take decisions which will impact the company big time because of assumptions. With DW every decision which you make is based on the past and with the valid industry standard analysis. So, with DW any decision you take will be informed decision.

What is Data Warehouse?

The data repository of an organization which helps you to analyze (slice and dice) the data to make strategic decisions.

Warehouse – where we place lot many things which we deal in the business (check with answers.com) for definition.

So, data warehouse is a place where we have ALL the DATA in ONE place.

Data Warehouse is a place where we store history data. We use history data to see how the company perform over the time and also predict the future (History repeats…..). History data is used for trend analysis in a bigger way in any organization.

As we discussed before in the problem statement, there should be a need to build a data warehouse. If business does not require a DW, then we wont be able to provide the ROI from the data warehouse we built. Just technology wont change the users mind set, so lot of time and energy must go towards explaining the pro's of using the same.

DW is a subject oriented, integrated, time variant and non volatile data which helps to take an organizational decision (Bill Inmon)

Each property given by Bill Inmon gives us more understanding of the Data warehouse. The **subject oriented** approach gives us ONE VERSION OF TRUTH in our data warehouse. The **integrated** approach gives us the ability to get 360 degree view of the organization, **time variant** gives us the power of more analysis and the last one **non volatile** tells us not to change the data quite often in the DW.

So we can consider DW is not just a project which solves the business process. It's an infrastructure where we can do analysis. Its not one particular technology, it's a blend of technologies which includes ETL (Extract, Transform and Load) and RDBMS database (Oracle, Teradata , SQL Server etc) and Business Intelligence (Business Objects, Cognos etc…)

Usually we discover, design, develop and deploy. We can develop data warehousing in the above mentioned steps.

Discovery> where business really finds the requirement of data points to run more effectively

Design> Based on what information business needs design your system to consolidate the data for analyis.

Develop> Develop the code.

Deploy> Deploy the code to the production for the analysis.

In DW, enhancement never stops because once the business analyst understands the data what DW has, analyst thinks in different ways to do analysis and get to the point where it needs to add the data to data warehouse.

DW goes through so many stages. When first develop usually we have lot of static reports (What happened?), slowly analyst will find the requirement for Ad hoc queries (Why happened?), then analyst interest in what will happen (what if analysis), then overseeing the operational data (Operationalizing) to make sure the current operations are going on fine and finally it transforms to Activie Data Warehousing (Event driven decisions) or Business Activity monitoring which monitors the data, based on certain identified event (positive or negative) it triggers an email or SMS to the respective resources (Pro active way of identifying instead of reacting after the fact). By proactive decisions business makes sure the business use every bit of opportunity to maximize the returns.

Differences between OLTP and Data Warehousing

OLTP systems were designed to run the business. It takes care of day today activities of the business. Raising invoices, receiving the payments, clearing your dues to your suppliers, running pay roll for the organization, maintaining the employees (adding new employees, updating the info, deleting emp) are considered to be OLTP applications.

We will take an example of a bank where branch deals with the retail banking like maintaining savings account, current account and fixed deposits.

If you focus only on savings account, a customer comes at 10 AM, depsoits 10K, around 11 AM deposited another 10K, with drawl of 5K and at around 2 PM there was 5K with drawl and 40K deposits. The transactions what customer makes are very frequent and the balance amount what the customer has changes based on any given transactions. So in this case we do lot of inserts and handful of updates to existing information. So the nature of this system is lot of changes in the data ie volatile in nature. In OLTP all the DML statements (insert, update, delete, selects) are active. OLTP applications we develop are based on the normalized data model (usually we go up to 3rd normal form) to have more flexibility, to reduce the redundant data, this in turn increases the transactions throughput (800 txns per minute…. etc). We don’t store too much of history data in OLTP applications, because if we store more data the transactions, through put becomes very less (performance of the system becomes slower). Usually in OLTP systems, we store six months of data to 1 year data. In times based on the government regulations in the banking / telecom industry forcefully stores around 2 years of data.

In OLTP systems, we don’t bring in too much of data from the database. We always query based primary key and foreign key while we retrieve the data. We use point queries in the OLTP applications. If you take a scenario of customer service application, when a customer calls a telephone cust service representative you have to give telephone number or account number so that we pull data in the minimum possible timing. The system won’t be able to pull the data if you give the city because OTLP systems are designed to run the point queries.

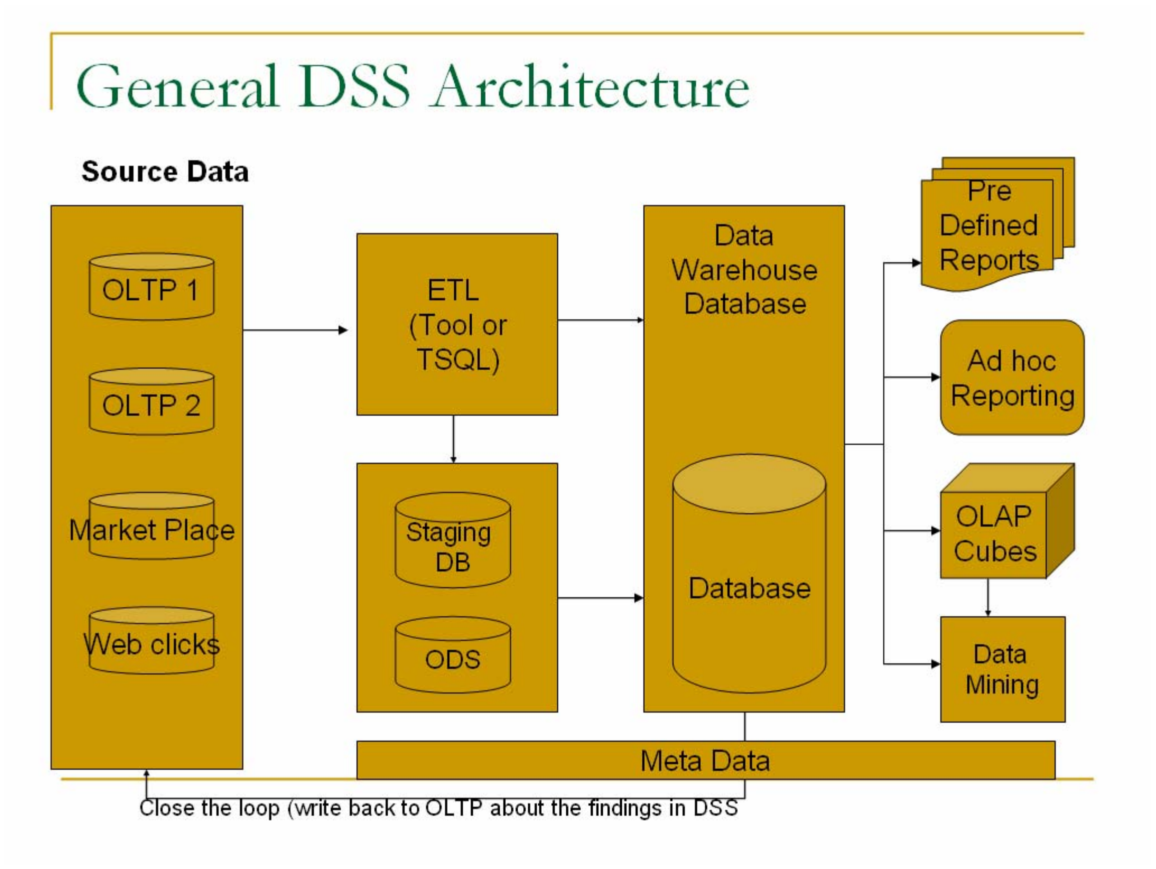
In case of Data warehouse we don’t change the data, thus we store the history of data. With out history we won’t be able to do the value trend analysis. We do analysis using the data warehousing data. Way before we used to call DW as EIS (Executive information systems) as the name explains the data what we have in DW affects the way company executes takes decision based on EIS. As the DW is mainly for analysis, the data DW is summarized. The summarized data have more value for analysis. We design the DW in a subject oriented manner. We always design the data warehouse based on how the business is viewed at an organizational level.

As we do analysis, the queries usually pulls more data. We execute range queries in the data warehouse say if you want to get total revenue break up by customer type in 2005 then we are accessing all the data which we store the entire year 2005. We design the DW database in a de-normalized model to speed up the queries. In some RDBMS we can still design the DM on 3rd normal form still yet get better performance.

General Architecture of Data Warehousing

In general every Data warehousing follow the same principle of collecting the data from across your organization and put it one place to get 360 degree of organization or department.

So the general architecture of DW as follows.



It usually consists of 4 parts, Source system, ETL component, Database Component where we store the data for analysis and the BI component through which our end users do analysis on. We will see the details of each of these components.

Source System

The system which feeds the data to DW, we call it as the source system. In a typical ware house we deal with multiple sources of information to populate the data warehouse. An enterprise can have lots of OLTP applications to take care of multi geo and departmental needs. As the no of source systems increase the complexity of ETL becomes more. Source system can be a database, an excel, XML, file etc. You can even consider a file from external source if you are getting that information from the third party customer

(usually market data)

What is ETL?

ETL stands for

Extract
Transform and
Load

It's a process where we clean the incoming data. To transfer the data from source system to DW database we use this layer. Extract can be done using the SQL or file read based on the type of source system. Say, we have a bank which has 100 different branches which does lot of transactions over the ATM, teller, online banking etc. Operational data is scattered over 5 different OLTP applications. End of the day we want to collect those transactions into the data warehouse, so we run some select statements to the sources system to pull that days ytransactions. This type of getting into from the database we call it as pull mechanism. In some cases what happens in OLTP systems generate a file for DW and places it in a specific lication so that DW can read from that, we call this as push and pull as the source pushes the data and DW pulls the records.

In OLTP system, say customer did 5 transactions on a day, we have minimum of 5 different records. In the DW may store only one record (consildated for the day level) , so what we do we summarize the data of OLTP system. So we are transforming the data which we read from source from 5 records to one record in this scenario. We will take another example, to explain the transform we are getting only state_id from OLTP system where as we have state name in the target database, in this case we search the state name which is associated to that state_id and populate the anem instead of id. Some times we reject the incoming data as the data is in incomplete state. All these process we call it as TRANSFORM.

Once you transform the data based on the business rules and target data model, then we LOAD the data into the data warehouse.

Data warehouse Database

The actual data we store for the anlysis. To tart with you may have one part of your organization data in the DW, as the time passes by we will try to consolidate the entire organizational data to view 360 degree of the organization. To create the DW database either we follow dimensional modeling or 3rd nor,mal form ET modeling. Which is good is based on the amoint of data, what RDBMS yis in use to store the data and lot other parameters.

Teradata is an RDBMS which is specifically designed for the DW implementations, which recommends to desing the data mdel in the 3rd NF so that you have more flexibility at the same time as the RDBMS can pull the data in a faster way the end users don't face the perfrormance problem.

Say weare implementing the DW in a EDBMS like SQL Server or Oracle, as the database size increases the query performance goes down. In this case we prefer for dimensional modeling because the query performance is better as we deal with less number of tables, more over for business users its easy to understand the model. DW data should be designed to store ONE VERSION OF TRUTH. DW database must support lot of users and processes as it should be big power house by itself.

Business Intelligence

This is the layer through which DW end users view the data in the form of reports, graphs etc. Its used to view the data in different formats / ways based on the end users visualization. Here they view the data in a multi dimensional way (OLAP reporting). OLAP stands for Online Analytical Processing, this means if an analyste wants to find top 10 customers based on the one product sale in 8 different countries then we should a provide a mechanism / tool so that analyst can do it immediately. The type of reporting is called Ad hoc reporting.

The way it evolve in real time scenario is, to begin with lot of managers may ask certain static reports or canned reports to do their job. As the become more familiar with the data and process they want more information in the form of drill down, dripp up so that they can do analysis on their own. As the end users become more and more aware of data they want to see the sales information through time or through customer or product wise. This type of reporting is called multi diemsional reporting or slice and dice.

All the BI tools should support the Ad hoc queries, other wise end users should wait few days so that IT department develops the report. This will impact the right decision at the right time time concept of DW.

In BI now we have Data mining and Business Activity monitoring as part of BI. Usually BAM is a proactive way of identifying and even and react based on that.

Data Warehouse / Data Mart

A data warehouse has all the organizational data to view 360 degree of an organization. Usually in the industry we call it as an Enterprise Data Warehouse (EDW). If an org is very large and has different line of business, then for each line of business we may create a DW which is huge by itself and an Enterprise data warehouse which gets data from the datawarehouses. If you visualize GE as an org, as they dealwith various different line of businesses they may follow this. In some cases each geo will have a DW and there will be DW which gets data from all the regions.

Usually Enterprise data warehouse will have all the data integreated in one place like material management, manufacturing data, supplier data, customer data, sales data, asset information, HR information etc etc which in turn gives you the 360 degree of the company.

In a nut shell EDW is the ALL DATA in ONE database. As the networking capabilities are scaling to new heights every day, any where from the world analysts will access the same EDW data for the analysis.

In contrast a Data Mart contains the departmental data. It gives the clear picuture (360 degree) for a department for analysis. You can say a data mart is a subset of datawarehouse. If an org does not have a centralized data store, the department which needs to do analysis may start creating a Data mart which pulls the data from the respective OLTP systems. Some times you can consider this as the initiation point to build the DW in some cases.

Later we will see the different approaches of building data warehouses, in that we will see the pros and cons of creating the Datawarehouse Vs DataMarts.

Data Granularity

In the OLTP systems we store all the details of the business and its transactions. We follow application oriented design to develop the application. We implement the business rules for specific application, say we cannot do a transaction until he / she becomes a vendor. To become a vendor you must be having TAX ID, bank account, wire transfer facility, minimum number of employees etc. This type of imposing rules applies only to OLTP applications.

As the DW end users want more of summary data to do analysis, we may desing the DW to hold the daily level or weekly level or monthly level summary data. The advantage of doing this we hold less no of records when compared to OLTP system with out loosing the information. The important step hile designing the DW is fixing the granularity o the data warehouse. So granularity is defined as the level of detailed information you want to store in the data warehouse.

In some cases the DW team wants to store all the dtails as same as OLTP, in this case the granularity is same for both the system. As the DW team wants more and more info, we need more detailed data in the DW. As we have more detailed infor in the DW, the complexity of handling data is very high as the data volume is huge.

Approaches to Build a Data Warehouse

We have three different types of approaches to build the datawarehouses.

- Top down approach
- Bottom up approach
- Hybrid approach

Top down approach

In this approach of building the DW, the high level committee which includes the representatives from each department will come together and define what type of analysis they wish to have to run the business. By collecting the requirements the team will desing the model through which the org can do the analysius in a cross functional way. In the industry we also call this as big bang approach. Uusally this type of implementation has much risk as we see everything at once or nothing at all. The pro's of this type of implementation is completeness, if we implement this way we wont miss the bigger picure and the required data sets from various OLTP. The con's of this of implementation takes longer time as there are lot of stakeholders. This model of implementation produces the SINGLE VERSION OF TRUTH data warehouse for an organization.

Bottom up approach

In this approach every department which needs the analysis to be done, they create the data mart of their own which suffice their needs. The turn around time is less as we have less number of stakeholders. In this type every department will start creating marts and at the end you will have many data marts which has its own standards. The disadvantage of this type of implementation is, we will end up having redundant data across the data marts, doing the same process in multiple places. The goal of DW is providing single version of truth may not be met with this type of implementation. End of day you will have to maintain different data marts which will increase the cost of the maintenance.

Hybrid Approach

In this approach we start with a data mart and slowly we add data sets which will end up having a data warehouse for the entire organization. Identify which department needs the analytics the most, start creating the mart keeping in mind that you will enhance this database to hold the entire organization data in the long run. Identify the next department add the new data sets needed and reuse the existing data so that all the data is in one place. In this approach you use the confirmed dimensions so that we reuse the dimensional data. Later when we talk about dimension and facts we will explain the confirmed dimension concept. This gives the immediate ROI and also the team will learn through each cycle and build the DW in a nicer way. This type of implementation process has less risk for the organization.

Data Modeling for Data Warehouses

Data Model can be defined as the pictorial representation of your business entities and attributes and its relationships. We follow normalization process while creating the data model for OLTP applications. The advantage of normalized model is remove redundant data, increase the flexibility and faster transaction through put. The disadvantage is the queries which run against the normalized tables are quite complex and degrades the performance of the system.

Data warehouse is a system where lot of resources queries the data, so select statements are very active, there over the data what they get is huge. As the selects are very active we can de-normalize the table to improve the performance. Usually we follow dimensional modeling concepts while create the data model for DW, but its not necessary to create the same. Teradata RDBMS recommends to create the DW model in 3rd normal form so that you get lot of flexibility and more over teradata can retrieve the data in a reasonable time with out compromising the performance.

Why Denormalization?

As you know, as you normalize the database more and more, the performance of queires become worse if we join many tables. In Data warehousing we usually store more tables as we take care of entire organization especially in EDW kind of data warehousing. As we do run lot of queries while analyzing the data as well as we require more data (like years worth of data) some times especially for trend or comparative analysis. Its better to think about the denormalization process. Some of the RDBMS like TERADATA can handle huge amount of data.

So, again its not necessary to de-normalize the tables for DW, its up to the technical team to evaluate the RDBMS they are using, evaluate the advantages and disadvantage to decide the best suitable one.

Advantages of de-normalization is, we will have less number of tables to deal with at the same time the queries will run faster as the number of tables are less in our joins.

So take away is either you create the data model in 3rd normal form or in dimensional modeling. In dimensional modeling we have two types

Star Schema
Snow flake schema

Dimensional Modeling

The methodology we follow to create a de-normalized data model for DW implementations. Before we get more into details of dimensional modeling we will talk about Dimensions and Facts.

Dimension table
A category through which you measure the performance of a business / organization.
You measure the revenue by year, qq, month etc
You measure revenue by customers
You measure revenue by products
You measure revenue by location

In the above example we are looking / analyzing the revenue by different categories like customers, products or locations. These are called dimensions. 80% of the tables in the DW are dimensions. So the way you model the dimensions (how to store the changes of the dimension data) in the Data modeling represents the dimensional modeling.

You can decide to store only current or to store complete history of changes in the dimension table. We will discuss about slowly changing dimensions. Take an example of any organization, employee is a dimension. Over the period of time, employee information like grade, salary, job role etc changes. Think an employee 'TIM' joined in XYZ corporation in Year 2002 as a software engineer, after two years he became Project lead i.e. in Year 2004 and now (current year) he is Project Manager. If we are using the HR OLTP application we can see TIM is currently working as Project Manager, but we won't be able to see what his previous job role was as we usually update the information in employee table. In data warehousing we store time varying data, so for the above example we were discussing in DW we will three records as TIM changed his job role from software engineer to Project lead to project manager. So in the oldest record for TIM in data warehousing contains Software engineer as his role and the latest record for TIM in data warehousing contains Project Manager as his role. This way we can analyze employee TIM for better than just knowing what role is playing. More detailed information gives us the ability to take the data driven decisions.

Attributes of Dimension table

Define business in terms already familiar to users
Wide rows with lots of descriptive text
Small tables (about a million rows)
Joined to fact table by a foreign key
Heavily indexed
Typical dimensions are time period, geographic region, products etc

Fact table

The fact table contains only the numerical values which helps to measure the performance. The value or the number of visits to the website, any thing to deal with money where we can find sum, avg etc we folded into the fact table.

Attributes of Fact table

Mostly raw numeric items
Narrow rows, a few columns at most
Large number of rows (millions to a billion)
Access the data via dimensions

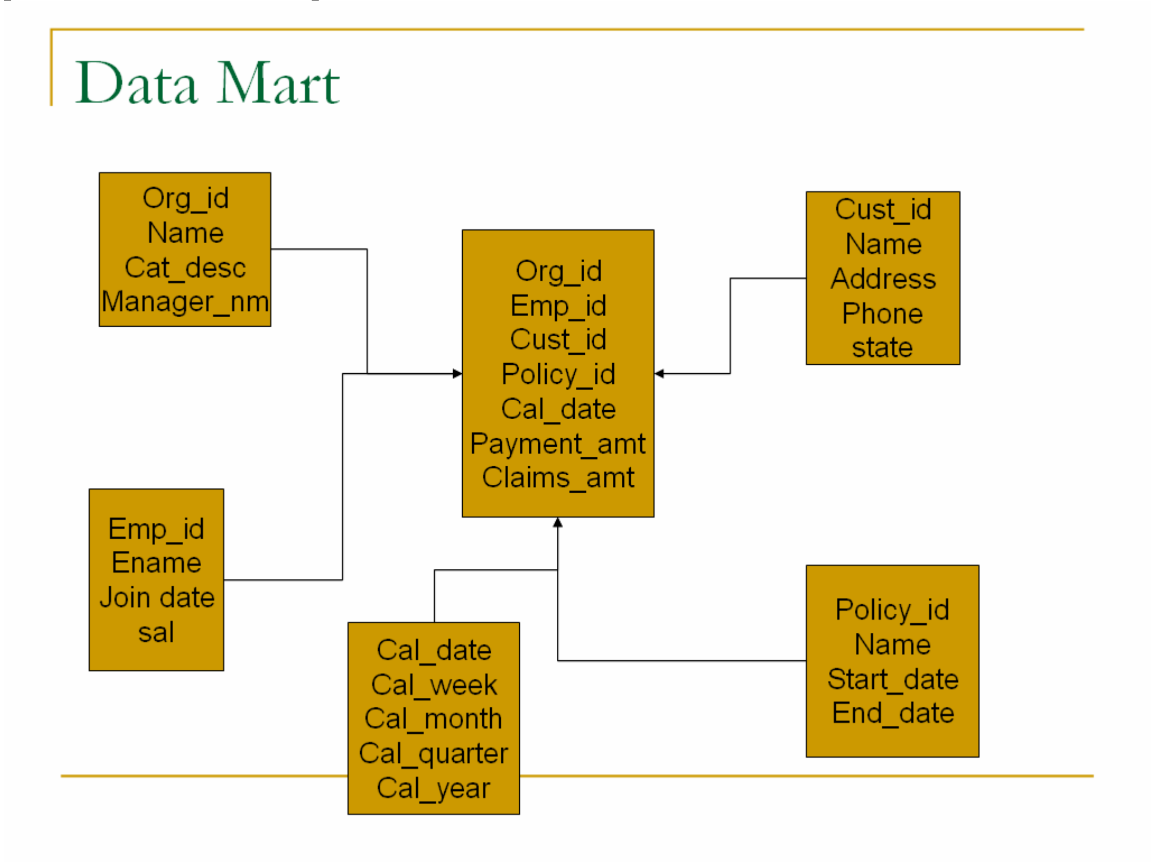
As we know what a dimension and fact table is, now we will get into star schema and the snow flake schema.

Star Schema

Is the type of modeling where we follow the complete de-normalization process. It's easy to understand because it reflects the model based on how your business works. It can be understandable even for the business user.

A star schema contains multiple dimensions which share the same fact table. Say per example in the sales analysis star schema, the actual sales goes into fact and the customer, product, date, sales force information goes as dimensions.

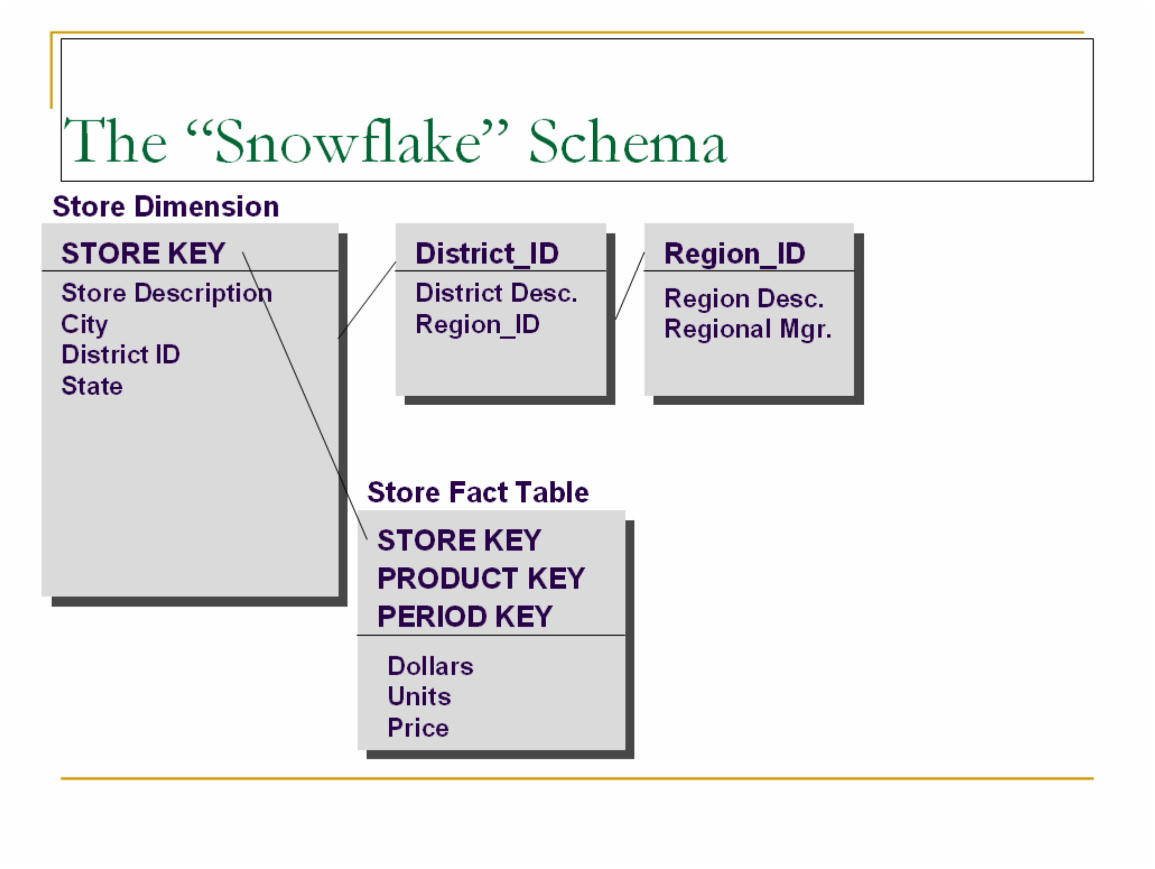
The following star schema is based on the lay man insurance business which deals with customer, policies, period and employees and organization. Using this, insurance company can do analysis on their revenues based on organizational unit, employees, policies, customers and period.


Data Mart
Org_id Name Cat_desc Manager_nm
Org_id Emp_id Cust_id Policy_id Cal_date Payment_amt Claims_amt
Cust_id Name Address Phone state
Emp_id Ename Join date sal
Cal_date Cal_week Cal_month Cal_quarter Cal_year
Policy_id Name Start_date End_date

The visual of the model imitates the star, so we call it as star schema. In case of start schema all the information about customer being stored in the same table. So usually the dimensions in star schema are wider in nature meaning lot of columns. So your dimension will have lot of redundant information.

Snow Flake

In order to strike a balance between 3rd normal form data model and star schema, we have snow flake schema which is more of normalized star schema. Even in snow flake we will have the fact and dimension only big difference is the dimension tables are normalized. So the above data model (Star) will looks like the following model in the snow flake model.



So, now you can visualize a snowflake does not have redundant data in the dimension tables. We will end up having lot of look up tables for the dimension table. As the no of tables become more the query performance goes down. So you have to take a decision on whether to go for star or snow flake depending on the data volume and the user performance requirements etc. Usually if we end up having lot of aggregate tables in the database, then snowflake is more handy.

Slowly Changing Dimension

This is the method through which we store or capture the history data. Example, we have customer who is with us from last 3 years. The customer would have changed the address, phone, city etc. If business wants to store the changes happened to the customer from last 3 years then we store all the changes in the data warehouse. Business can also say in certain dimensions they don't want history because it does not add any value for analysis.

Per example, DATE / PERIOD dimension values are static in nature. So, take away is not all the dimensions necessarily to store the history.

Over the period of time, the dimension data tend to change and how you want to capture the changes is what slowly changing dimension is all about.

There are three different ways we capture the history

Slowly Changing Dimension 1 (SCD1 / TYPE1)
Slowly Changing Dimension 2 (SCD2 / TYPE2)
Slowly Changing Dimension 3 (SCD3 / TYPE3)

TYPE1 / SCD1:
When we use this TYPE1, we don't store any history. Only the current information is stored in this type of dimension. So if your business says, we don't need history, then we can opt for TYPE1 / SCD1. If your organization is dealing financial services, then you SMG can say for the campaign management system, we do analysis based on the current information of customer rather than previous data then we can choose TYPE1.

TYPEII / SCD2:
We use this method to store the complete history. In this case if customer information got changed thrice in the OLTP application, the DW will have 3 records. We use an id called surrogate ID in TYPE II because the customer ID can exists more than once, where as in TYPE1 you will have only one record per customer, when there is a change we update the data thus we maintain only the current in TYPE1. Usually we store customer, product dimensions in TYPEII model as we can do better analysis like while this was selling at 'X'$ we use to sell 100 qty where as when are selling at 'Y'$ we are selling 150 qty etc. If we don't store the history you cannot get this kind of information from DW. TYPEII enables your DW to track the changes happens to data even when your OLTP does not maintain.

TYPE III / SCD3:
In this method we store current and previous values of the dimension data. Say per example we want to track the history on city column of a customer then we design the table in the following way.

CUST_ID	CUST_NM	PHONE	CITY	PRE_CITY
1000	HSBC	8909173763	JAIPUR	MUMBAI
1001	CITI	8903-2039	MUMBAI	NULL

You will have only one record per cust_id we will have one more column on which you want to store history and call it as pre_colname. Typically in many implementations we don't use TYPE 3 dimension type.

Now we will take the example of a source system data and will run through the TYPE1, TYPE2 and TYPE3 examples.

SOURCE TABLE

EMPL_MSTR (employee master table)

Emp_id	Ename	Sal	Deptno	City
4000	Ravi Kumar	9000	10	CHENNAI

TYPE1 Dimension (EMP_DIM)

EMP_ID	ENAME	SAL	DEPTNO	CITY
4000	Ravi Kumar	9000	10	CHENNAI

TYPE II Dimension (EMP_DIM)

EMP_SUR_ID	EMP_ID	ENAME	SAL	DEPTNO	CITY
100	4000	Ravi Kumar	9000	10	CHENNAI

TYPE III Dimension (EMP_DIM)

EMP_ID	ENAME	SAL	PRE_SAL	DEPTNO	PRE_DEPTNO	CITY	PRE_CITY
4000	Ravi Kumar	9000	NULL	10	NULL	Chennai	NULL

As a wrap up on dimensional modeling, we can have either create the data model using 3rd normal form, Star schema or snow flake schema for the data warehouse.

The pros of 3rd normal form is flexibility, the con's is difficult to understand for the businessusers and the SQL's will run slower as we have lot of tables. In some implementations we simulate star schema for business users over the 3rd normal form data model using VIEWS. By doing this writing the SQL become easier but under the hood database takes the hit while executing the same.

The pros of having star schema, easy to understand, reflects the business in terms of dimensions and facts. SQL's are faster. The disadvanatage is if we want to add new dimension to the existing star schema then you have either reload the fact table which takes longer time to load.

Snow flake, it's a balance between two. Pro's are less no tables when compared to 3rd normal form and more no tables when compared to star schema. So based on the dimension table size, available storage space you decide what kind of model suits the organization better to implement.

As we completed explaining about all the components in Data Warehousing architecture we will go deep into ETL process and BI reports.

Extract, Transform and Load

As you are aware of ETL layer in DW, now we get into more details of ETL like different type of extracts, transformations and loading process.

In the extract part, either we get all data from source tables or only the new and changed data from the source. We follow the first method where we don't have a mechanism to know what is new record or changed records. If we have audit columns like created_dt, updated_dt, created_by and updated_by in every record in the source then its easy to pull the records which got inserterd / updated based on the time stamp. Some times if we get legacy data we may not have this information, in that case we pull all the data from source and then gigure our what data to be updated / inserted or ignored because the data already exists in the DW. Usually if source system is not able to give only delta then we push all the extracted data into staging database then start the process of identifying the record which exists or not as part of the transformation logic.

As part of transformation, you will convert the incoming data to the respective data which is needed by the DW. If we change the state of the extracted data, we call it as a transformation. Following are examples of different types of transformations.

Lookup
Expression
Aggregate etc

In the look up transformation you pass the code or ID and get the related value from the reference table. (Example: Pass the country_id and get the country_name).

In the expression transformation we may add two column values in a row to get the new column value as output. If you are getting the product cost and the product selling price, you can calculate the margin using the expression transformation.

In the aggregate transformation we summarize the data (GROUP BY clause in SQL). If the granularity of DW is a week basis, in this case you run a SQL which summarizes the amounts, qty columns based on the week and then write it to the DW.

As part of transformation you change the data from one state to the other, ignoring some unwanted records, rejecting the data to the bin for further analysis, aggregate data, scrub the data so that we maintain the quality of the data in the DW.

As the transformation logic becomes more and more complex then we will have staging tables to process the data.

In the industry you will come across various terms we use in ETL process like, Extracting, Conditioning, Scrubbing, Scoring, full load, delta load, de duplication, validating etc etc. We will try to explain each one of this using an example.

Extracting

Pulling data from the source (can be partial or full extract)

Conditioning

Changing data types or formatting the data so that DW wont reject that value while pushing the data to the DW

Scrubbing

Is modifying the data with the correct information, example in developed countries the postel service maintains all the addresses in the country. By paying the price you can cross validate the address what you have is correct or not. Per example if you are sending the address with out zip code, or misspelled a city name, the system returns the data with the ZIP CODE and the correct CITY NAME. In this case we are making sure we have the correct data / filling the data as much as possible so that we can store the complete information.

Scoring

If you have process in DW where you grade the customer based on the sales happened every month based on which customer may upgrade to next level or degrade to the lower level. If you this process as part of the ETL code then we consider as grading process.

Full Load

Every time you populate the data the DW, you truncate / delete the records in the table and then load the data. Usually this takes longer time, if possible we should not follow this methodology at all.

Delta Load (Incremental Load)

Is uploading only the required data (inserted / modified) to DW which got inserted or changed after the last load time of the data warehouse. Your goal should be always we have to do the delta so that you can finish the ETL process in the shorter duration.

De Duplication

Is the process of identifying the duplicate data and load one instance from may. You may get customer from web source, call center source or 3rd party data. In this case the same customer can get from multiple sources, in the process we have identify those duplicates and remove them and pick the ones which has complete information based on the business rules.

Validation

To make sure the data what we are getting is valid or not. Making sure whether all the not null columns have values or not. If there is no value for an some column in the imcoming data, either ignore that record or reject that record or assign a default value. These are part of validation process. Business dictages if the record should be rejected or ignored or assign some default value.

Load Process

Once we transform the data as needed, we can load the data to the Data Warehouse. Some times we may decide to load the data by truncating or delete all the rows. Usually we do this while we are at the starting stage o implementing the DW. Once the team becomes good in identifying the data whether it's a changed or new record based on that we load only the delta or we do only the incremental update. When we are loading the DW for the first time we always use the FULL load. Going forward we will apply the changes in OLTP to DW.

We may face the following issues while we load the data into DW.

Lots and lots of data
Some DW are 24*7 meaning, all the time the system should be up and running
Creating summary tables (applying the new data to existing sum. Tables)
Creating Indexes
Monitoring the database / data warehouse

Important and help full tips to remember

1. Follow same methodology while loading the data in the entire organization
2. Keep track of timings like how much time it takes to extract, transform and loading process
3. How many end users are very active in analysis, have a usage analysis so that you can calculate the ROI.
4. Document the process you used to have before the DW, and explain how many days / hours to generate the same report it used to take
5. Educate the end user on how else they can utilize the DW data in their day to day analysis.

Business Intelligence

Layer through which end users view the data. The data can be viewed in the form of static reports, ad hoc reports, cubes or you use Data mining and active data warehousing or Business activity monitoring for the same.

What is OLAP?

OLAP - Online Analytical Processing. The way analysts slice and dice the DW data. Viewing the fact information through multiple dimensions is called slice and dice. Example: if we see the revenue by product and year they we call the report as multi dimensional reporting.

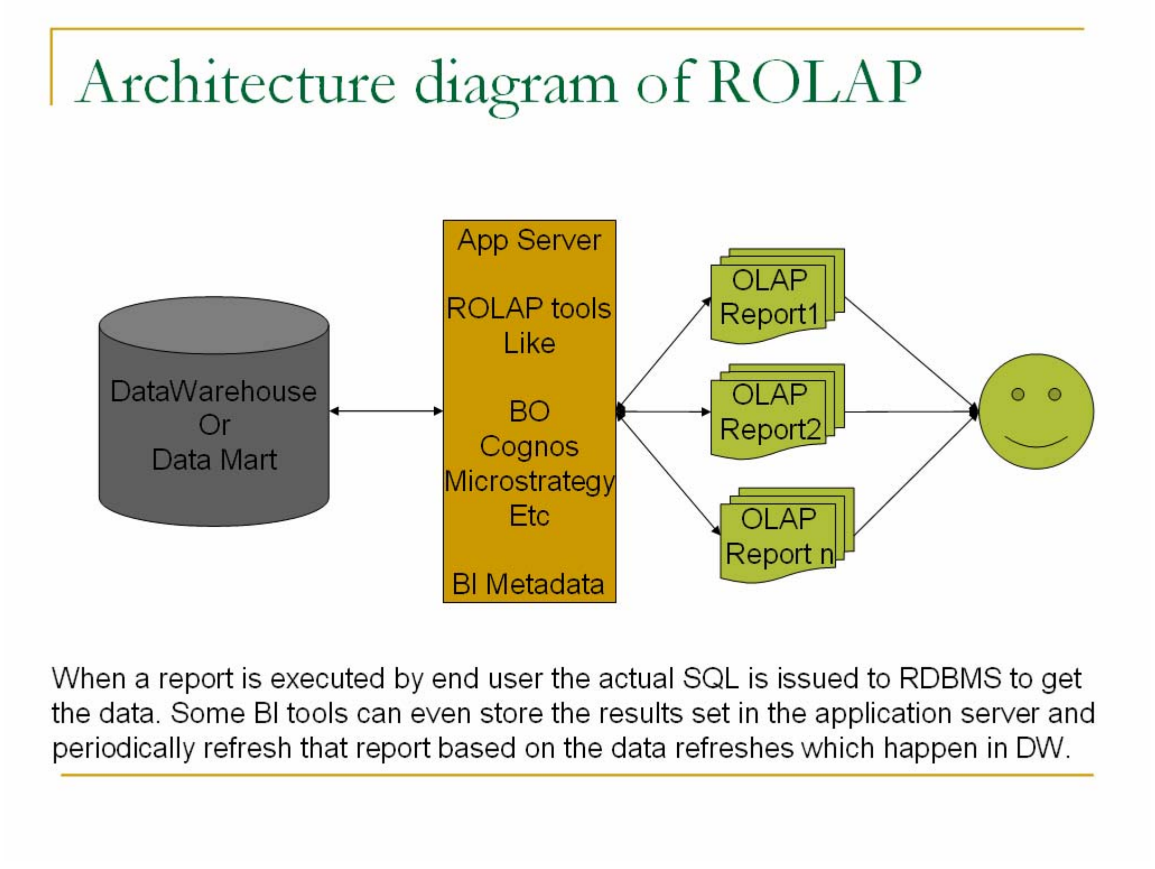
Usually when we implement the DW, we create of static reports; some times we call it as canned reports as well. As users become good in understanding the data, to analyze why such slip or peak happened in specific scenario we will open up for the Ad hoc query. Using Adhoc query facility end users can pull in the data what they need in the way they want to see. Any tool, which exposes the DW model through meta data and helps to create the report then we call it as BI tool.

In the industry we have three types of OLAP.

ROLAP – Relational OLAP.
MOLAP – Multidimensional OLAP
HOLAP – Hybrid OLAP.

ROLAP tools in the industry are Business Objects, Microstrategy, Cognos Report net etc. We call this tools as ROLAP because the data what you see in the reports are stored in the relational databases where as we display the data in a multidimensional way from the reporting tools. Following is the architecture diagram of ROLAP.

ROLAP Architecture

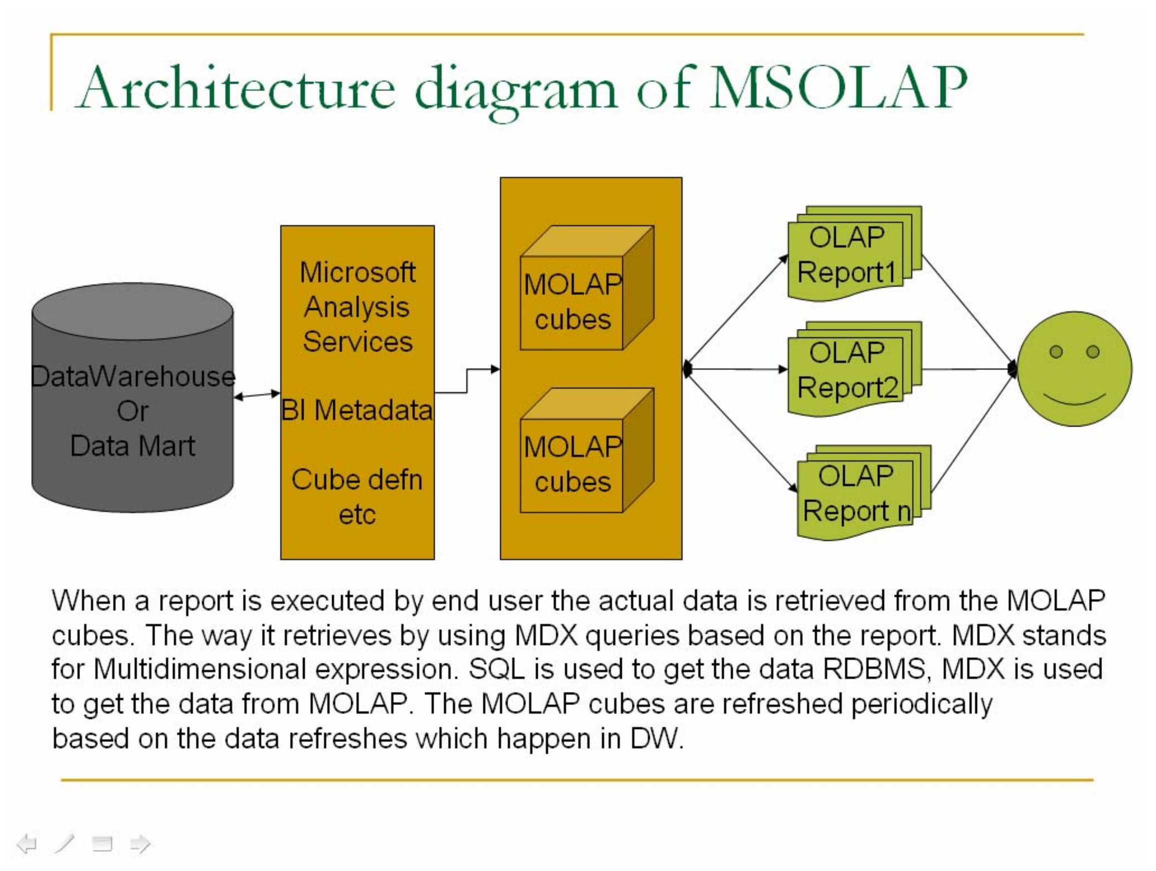


Architecture diagram of ROLAP

When a report is executed by end user the actual SQL is issued to RDBMS to get the data. Some BI tools can even store the results set in the application server and periodically refresh that report based on the data refreshes which happen in DW.

MOLAP tools in the industry are Microsoft Analysis Services, Cognos Power Play, Essbase etc. We call this tools as MOLAP because the data what you see in the reports are stored in the multi dimensional databases. Following is the architecture diagram of MOLAP. Usually we start the reporting for any organization as ROLAP system, as the time goes by the data in the data warehouse become more. In this scenario, the reports start running slow, in this case we can suggest MOLAP tools to run reports faster for our end users.

Architecture of MOLAP



Architecture diagram of MSOLAP

When a report is executed by end user the actual data is retrieved from the MOLAP cubes. The way it retrieves by using MDX queries based on the report. MDX stands for Multidimensional expression. SQL is used to get the data RDBMS, MDX is used to get the data from MOLAP. The MOLAP cubes are refreshed periodically based on the data refreshes which happen in DW.

Mostly top level management wants to see the summary data. CEO may wants the profit level by Geo's, Geo's manager may want to see the profit by countries, country manager wants to see the profit by states, state manager wants to see the profit by city, city manager wants to see the profit by customer. This is the usual way every one in the organization takes care of the business. End of the day the sales rolls up to worldwide sales.

To do the above said process we create hierarchies based on a dimension. Say we have customer dimension, the hierarchy of customer dimension may be Geo, Country, State, City and customer name. So, hierarchies are nothing but the navigation path in the reports through which management can easily locate the data they are looking for.

Some dimensions can have multiple hierarchies as well. For example if we consider period or date dimension you may have physical calendar information as well as financial calendar information. If you consider the financial year in India its between March to March. So, 10-JAN-08 belongs to financial year 2007 where as for the same date belongs to physical year 2008. Finance department wants to do analysis by financial year where as the other departments like HR and SMG may be looking the data in terms of physical year.

In the above scenario, **period dimension** table in DW looks like
Date_id
Day_date
week
month
quarter
year
fin_week
fin_month
fin_quarter
fin_year.

When it comes to the design of hierarchies it will look
Physical Calendar Hierarchy
Year
Quarter
Month
Week
Day_date

Financial Calendar hierarchy
Fin_Year
Fin_Quarter
Fin_Month
Fin_Week
Day_date

Usually if we do analysis in One Year you will have multiple Quarter, One Quarter you will have multiple Months, in one month you will have multiple weeks, in one week you will have multiple days.

We have to learn what Drill down, Drill up and Drill through is all about in OLAP. Drill down mechanism helps us to get more information from one level.

Conclusion

We hope by now you know what is data warehouse and different components in the warehouse and usage of warehouse.

Build data warehouses for people who utilize the data and improve the way they do operations. Otherwise we won’t be able to provide the ROI.