DATA WAREHOUSING ARCHITECTURE

What is Data warehouse?

Data warehouse is an information system that contains historical and commutative data from single or multiple sources. It simplifies reporting and analysis process of the organization. It is also a single version of truth for any company for decision making and forecasting.

Types of architecture

- Single tier architecture
- Two tier architecture
- Three tier architecture

Single-tier architecture

The objective of a single layer is to minimize the amount of data stored. This goal is to remove data redundancy. This architecture is not frequently used in practice.

Two-tier architecture

Two-layer architecture separates physically available sources and data warehouse. This architecture is not expandable and also not supporting a large number of end-users. It also has connectivity problems because of network limitations.

Three-tier architecture

This is the most widely used architecture. It consists of

- Top tier
- Middle tier
- Bottom Tier.

Bottom Tier

The database of the Data warehouse servers as the bottom tier. It is usually a relational database system. Data is cleansed, transformed, and loaded into this layer using back-end tools.

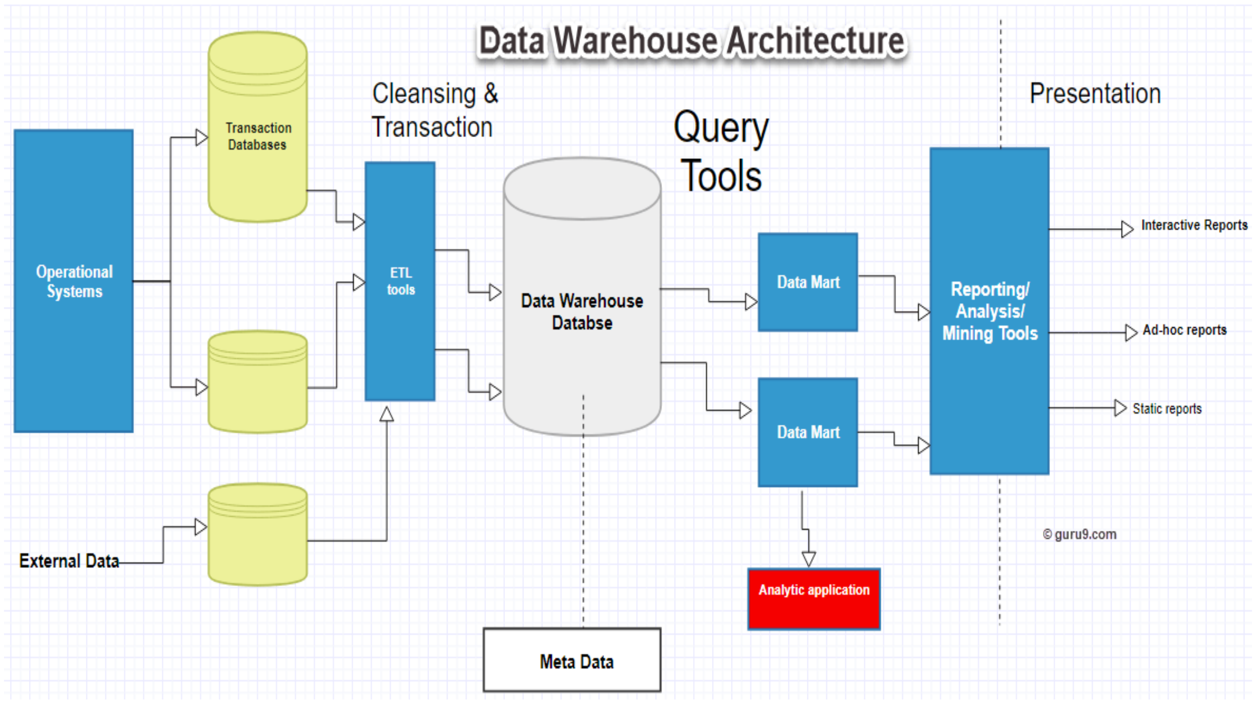
Middle-Tier

The middle tier in Data warehouse is an OLAP server which is implemented using either ROLAP or MOLAP model. For a user, this application tier presents an abstracted view of the database. This layer also acts as a mediator between the end-user and the database.

Top-Tier

The top tier is a front-end client layer. Top tier is the tools and API that you connect and get data out from the data warehouse. It could be Query tools, reporting tools, managed query tools, Analysis tools and Data mining tools.

DATA WAREHOUSE COMPONENTS



Data warehouse Components

The data warehouse is based on an RDBMS server which is a central information repository that is surrounded by some key components to make the entire environment functional, manageable and accessible. There are mainly five components of Data Warehouse.

- Database
- ETL Tools
- Meta Data
- Query Tools
- Data Marts

These are four main categories of query tools

- Query and reporting, tools
- Application Development tools,
- Data mining tools
- OLAP tools

Data Warehouse Database

The central database is the foundation of the data warehousing environment. This database is implemented on the RDBMS technology. Although, this kind of implementation is constrained by the fact that traditional RDBMS system is optimized for transactional database processing and not for data warehousing. For instance, ad-hoc query, multi-table joins, aggregates are resource intensive and slow down performance. Hence, alternative approaches to Database are used as listed below

- In a data warehouse, relational databases are deployed in parallel to allow for scalability. Parallel relational databases also allow shared memory or shared nothing model on various multiprocessor configurations or massively parallel processors.
- New index structures are used to bypass relational table scan and improve speed.
- Use of multidimensional database (MDDBs) to overcome any limitations which are placed because of the relational data model. Example: Essbase from Oracle.

Sourcing, Acquisition, Clean-up and Transformation Tools (ETL)

The data sourcing, transformation, and migration tools are used for performing all the conversions, summarizations, and all the changes needed to transform data into a unified format in the data warehouse. They are also called Extract, Transform and Load (ETL) Tools.
Their functionality includes:

- Anonymize data as per regulatory stipulations.
- Eliminating unwanted data in operational databases from loading into Data warehouse.
- Search and replace common names and definitions for data arriving from different sources.
- Calculating summaries and derived data
- In case of missing data, populate them with defaults.
- De-duplicated repeated data arriving from multiple data sources.

These Extract, Transform, and Load tools may generate cron jobs, background jobs, Cobol programs, shell scripts, etc. that regularly update data in data warehouse. These tools are also helpful to maintain the Metadata. These ETL Tools have to deal with challenges of Database & Data heterogeneity.

Metadata

The name Meta Data suggests some high- level technological concept. However, it is quite simple. Metadata is data about data which defines the data warehouse. It is used for building, maintaining and managing the data warehouse. In the Data Warehouse Architecture, meta-data plays an important role as it specifies the source, usage, values, and features of data warehouse data. It also defines how data can be changed and processed. It is closely connected to the data warehouse. Metadata helps to answer the following questions

- What tables, attributes, and keys does the Data Warehouse contain?
- Where did the data come from?
- How many times do data get reloaded?
- What transformations were applied with cleansing?

Metadata can be classified into following categories:

- **Technical Meta Data**: This kind of Metadata contains information about warehouse which is used by Data warehouse designers and administrators.
- **Business Meta Data:** This kind of Metadata contains detail that gives end-users a way easy to understand information stored in the data warehouse.

Query Tools

One of the primary objects of data warehousing is to provide information to businesses to make strategic decisions. Query tools allow users to interact with the data warehouse system.

These tools fall into four different categories:

- Query and reporting tools
- Application Development tools
- Data mining tools
- OLAP tools

Query and reporting tools

Query and reporting tools can be further divided into

- Reporting tools
- Managed query tools

Reporting tools

Reporting tools can be further divided into production reporting tools and desktop report writer.

- **Report writers**: This kind of reporting tool is tools designed for end-users for their analysis.
- **Production reporting**: This kind of tools allows organizations to generate regular operational reports. It also supports high volume batch jobs like printing and calculating. Some popular reporting tools are Brio, Business Objects, Oracle, Power Soft, SAS Institute.

Managed query tools

This kind of access tools helps end users to resolve snags in database and SQL and database structure by inserting meta-layer between users and database.

Application development tools

Sometimes built-in graphical and analytical tools do not satisfy the analytical needs of an organization. In such cases, custom reports are developed using Application development tools.

Data mining tools:

Data mining is a process of discovering meaningful new correlation, patterns, and trends by mining large amount data. Data mining tools are used to make this process automatic.

OLAP tools

These tools are based on concepts of a multidimensional database. It allows users to analyse the data using elaborate and complex multidimensional views.

Data warehouse Bus Architecture

Data warehouse Bus determines the flow of data in your warehouse. The data flow in a data warehouse can be categorized as Inflow, Upflow, Downflow, Outflow and Meta flow. While designing a Data Bus, one needs to consider the shared dimensions, facts across data marts.

Data Marts

A data mart is an access layer which is used to get data out to the users. It is presented as an option for large size data warehouse as it takes less time and money to build. However, there is no standard definition of a data mart is differing from person to person. In a simple word Data mart is a subsidiary of a data warehouse. The data mart is used for partition of data which is created for the specific group of users. Data marts could be created in the same database as the Data warehouse or a physically separate Database.

Data warehouse Architecture Best Practices

To design Data Warehouse Architecture, you need to follow below given best practices:

- Use a data model which is optimized for information retrieval which can be the dimensional mode, denormalized or hybrid approach.
- Need to assure that Data is processed quickly and accurately. At the same time, you should take an approach which consolidates data into a single version of the truth.
- Carefully design the data acquisition and cleansing process for Data warehouse.
- Design a Meta Data architecture which allows sharing of metadata between components of Data Warehouse
- Consider implementing an ODS model when information retrieval need is near the bottom of the data abstraction pyramid or when there are multiple operational sources required to be accessed.

- One should make sure that the data model is integrated and not just consolidated. In that case, you should consider 3NF data model. It is also ideal for acquiring ETL and Data cleansing tools

BUILDING A DATA WAREHOUSE

In general, building any data warehouse consists of the following steps:

- Extracting the transactional data from the data sources into a staging area
- Transforming the transactional data
- Loading the transformed data into a dimensional database
- Building pre-calculated summary values to speed up report generation
- Building (or purchasing) a front-end reporting tool

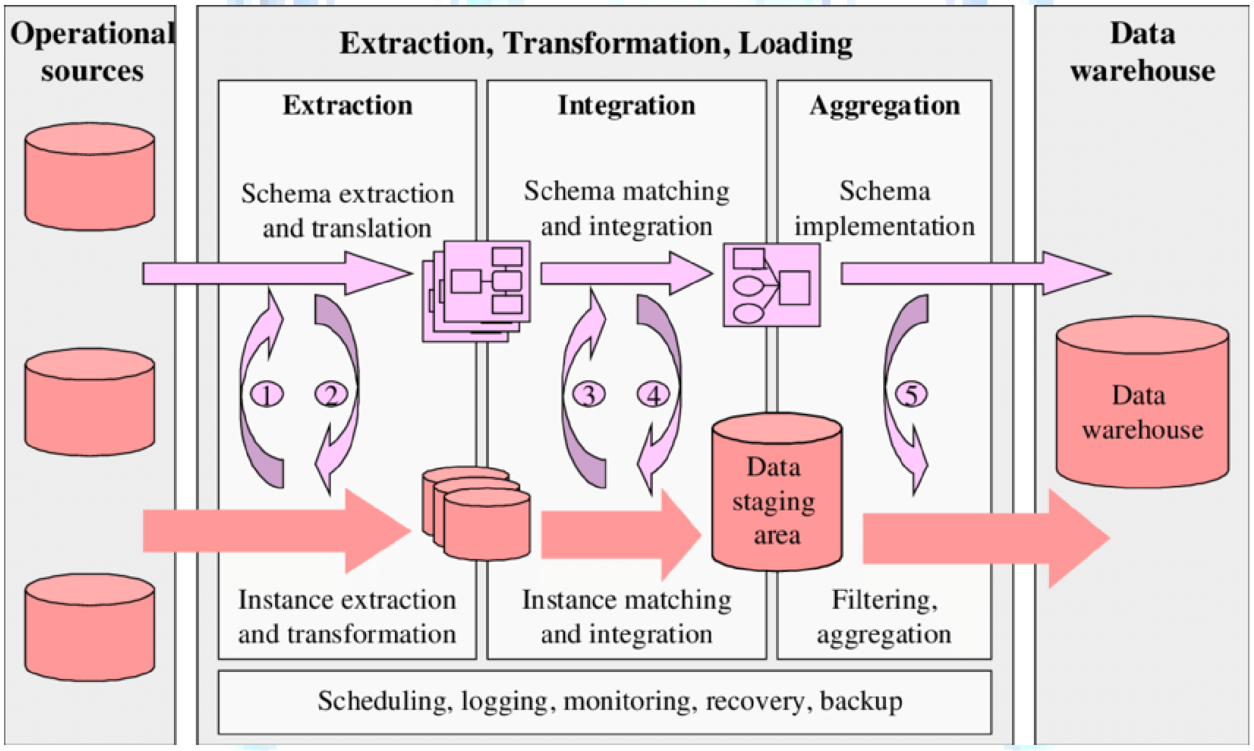

Operational sources
Extraction, Transformation, Loading
Data warehouse
Extraction
Integration
Aggregation
Schema extraction and translation
Schema matching and integration
Schema implementation
1
2
3
4
5
Data staging area
Data warehouse
Instance extraction and transformation
Instance matching and integration
Filtering, aggregation
Scheduling, logging, monitoring, recovery, backup

Diagram for building a data warehouse

Extracting Transactional Data:

A large part of building a DW is pulling data from various data sources and placing it in a central storage area. In fact, this can be the most difficult step to accomplish due to the reasons mentioned earlier: Most people who worked on the systems in place have moved on to other jobs. Even if they haven't left the company, you still have a lot of work to do: You need to figure out which database system to use for your staging area and how to pull data from various sources into that area.

Fortunately for many small to mid-size companies, Microsoft has come up with an excellent tool for data extraction. Data Transformation Services (DTS), which is part of Microsoft SQL Server 7.0 and 2000, allows you to import and export data from any OLE DB or ODBC-compliant database as long as you have an appropriate provider. This tool is available at no extra cost when you purchase Microsoft SQL Server. The sad reality is that you won't always have an OLE DB or ODBC-compliant data source to work with, however. If not, you're bound to make a considerable investment of time and effort in writing a custom program that transfers data from the original source into the staging database.

Transforming Transactional Data*:*

An equally important and challenging step after extracting is **transforming and relating the data** extracted from multiple sources. As I said earlier, your source systems were most likely built by many different IT professionals. Let's face it. Each person sees the world through their own eyes, so each solution is at least a bit different from the others. The data model of your mainframe system might be very different from the model of the client-server system.

Most companies have their data spread out in a number of various database management systems: MS Access, MS SQL Server, Oracle, Sybase, and so on. Many companies will also have much of their data in flat files, spread sheets, mail systems and other types of data stores. When building a data warehouse, you need to relate data from all of these sources and build some type of a staging area that can handle data extracted from any of these source systems. After all the data is in the staging area, you have to massage it and give it a common shape. Prior to massaging data, you need to figure out a way to relate tables and columns of one system to the tables and columns coming from the other systems.

Creating a Dimensional Model:

The third step in building a data warehouse is coming up with a dimensional model. Most modern transactional systems are built using the relational model. The relational database is highly normalized; when designing such a system, you try to get rid of repeating columns and make all columns dependent on the primary key of each table. The relational systems perform well in the On-Line Transaction Processing (OLTP) environment. On the other hand, they perform rather poorly in the reporting (and especially DW) environment, in which joining multiple huge tables just is not the best idea.

The relational format is not very efficient when it comes to building reports with summary and aggregate values. The dimensional approach, on the other hand, provides a way to improve query performance without affecting data integrity. However, the query performance improvement comes with a storage space penalty; a dimensional database will generally take up much more space than its relational counterpart. These days, storage space is fairly inexpensive, and most companies can afford large hard disks with a minimal effort.

The dimensional model consists of the fact and dimension tables. The fact tables consist of foreign keys to each dimension table, as well as measures. The *measures* are a factual representation of how well (or how poorly) your business is doing (for instance, the number of parts produced per hour or the number of cars rented per day). *Dimensions,* on the other hand, are what your business users expect in the reports—the details about the measures. For example, the time dimension tells the user that 2000 parts were produced between 7 a.m. and 7 p.m. on the specific day; the plant dimension specifies that these parts were produced by the Northern plant.

Just like any modeling exercise the dimensional modeling is not to be taken lightly. Figuring out the needed dimensions is a matter of discussing the business requirements with your users over and over again. When you first talk to the users they have very minimal requirements: "Just give me those reports that show me how each portion of the company performs." Figuring out what "each portion of the company" means is your job as a DW architect. The company may consist of regions, each of which report to a different vice president of operations. Each region, on the other hand, might consist of areas, which in turn might consist of individual stores. Each store could have several departments. When the DW is complete, splitting the revenue among the regions won't be enough. That's when your users will demand more features and additional drill-down capabilities. Instead of waiting for that to happen, an architect should take proactive measures to get all the necessary requirements ahead of time.

It's also important to realize that not every field you import from each data source may fit into the dimensional model. Indeed, if you have a sequential key on a mainframe system, it won't have much meaning to your business users. Other columns might have had significance eons ago when the system was built. Since then, the management might have changed its mind about the relevance of such columns. So don't worry if all of the columns you imported are not part of your dimensional model.

Loading the Data

After you've built a dimensional model, it's time to **populate it with the data** in the staging database. This step only sounds trivial. It might involve combining several columns together or splitting one field into several columns. You might have to perform several lookups before calculating certain values for your dimensional model.

Keep in mind that such data transformations can be performed at either of the two stages: while extracting the data from their origins or while loading data into the dimensional model. I wouldn't recommend one way over the other—make a decision depending on the project. If your users need to be sure that they can extract all the data first, wait until all data is extracted prior to transforming it. If the dimensions are known prior to extraction, go on and transform the data while extracting it.

Generating Pre calculated Summary Values

The next step is generating the pre calculated summary values which are commonly referred to as *aggregations*. This step has been tremendously simplified by SQL Server Analysis Services (or OLAP Services, as it is referred to in SQL Server 7.0). After you have populated your dimensional database, SQL Server Analysis Services does all the aggregate generation work for you. However, remember that depending on the number of dimensions you have in your DW, building aggregations can take a long time. As a rule of thumb, the more dimensions you have, the more time it'll take to build aggregations. However, the size of each dimension also plays a significant role.

Prior to generating aggregations, you need to make an important choice about which dimensional model to use: ROLAP (Relational OLAP), MOLAP (Multidimensional OLAP), or HOLAP (Hybrid OLAP). The ROLAP model builds additional tables for storing the aggregates, but this takes much more storage space than a dimensional database, so be careful! The MOLAP model stores the aggregations as well as the data in multidimensional format, which is far more efficient than ROLAP. The HOLAP approach keeps the data in the relational format, but builds aggregations in multidimensional format, so it's a combination of ROLAP and MOLAP.

Regardless of which dimensional model you choose, ensure that SQL Server has as much memory as possible. Building aggregations is a memory-intensive operation, and the more memory you provide, the less time it will take to build aggregate values.

Building (or Purchasing) a Front-End Reporting Tool

After you've built the dimensional database and the aggregations you can decide how sophisticated your **reporting tools** need to be. If you just need the drill-down capabilities, and your users have Microsoft Office 2000 on their desktops, the Pivot Table Service of Microsoft Excel 2000 will do the job. If the reporting needs are more than what Excel can offer, you'll have to investigate the alternative of building or purchasing a reporting tool. The cost of building a custom reporting (and OLAP) tool will usually outweigh the purchase price of a third-party tool. That is not to say that OLAP tools are cheap.

There are several major vendors on the market that have top-notch analytical tools. In addition to the third-party tools, Microsoft has just released its own tool, Data Analyzer, which can be a cost- effective alternative. Consider purchasing one of these suites before delving into the process of developing your own software because reinventing the wheel is not always beneficial or affordable. Building OLAP tools is not a trivial exercise by any means.