Data Aggregation

Data aggregation is the process where data is collected and presented in summarized format for statistical analysis and to effectively achieve business objectives. Data aggregation is vital to data warehousing as it helps to make decisions based on vast amounts of raw data. Data aggregation provides the ability to forecast future trends and aids in predictive modeling. Effective data aggregation techniques help to minimize performance problems.

Types of aggregation with mathematical functions:

Sum— adds together all the specified data to get a total.

Average— computes the average value of the specific data.

Max— display the highest value for each category.

Min— displays the lowest value for each category.

Count— counts the total number of data entries for each category.

Summary tables

Summary tables, also known as aggregate tables that store data at higher levels than it was stored when the data was initially captured and saved.

Summary tables are an important part of creating a high-performance data warehouse. A summary table stores data that has been aggregated in a way that answers a common (or resource-intensive) business query. Summary tables are all about speed. They're smaller than fact tables, which means they generally respond more quickly (fewer rows to query), and they deliver answers without calculating every result from scratch.

Points to remember about summary information.

- Summary information speeds up the performance of common queries.
- It increases the operational cost.
- It needs to be updated whenever new data is loaded into the data warehouse.
- It may not have been backed up, since it can be generated fresh from the detailed information

Data Marts

Data mart can be defined as the subset of data warehouse of an organization which is limited to a specific business unit or group of users. It is a subject-oriented database and is also known as High Performance Query Structures (HPQS).

E.g., Marketing, Sales, HR or finance. It is often controlled by a single department in an organization.

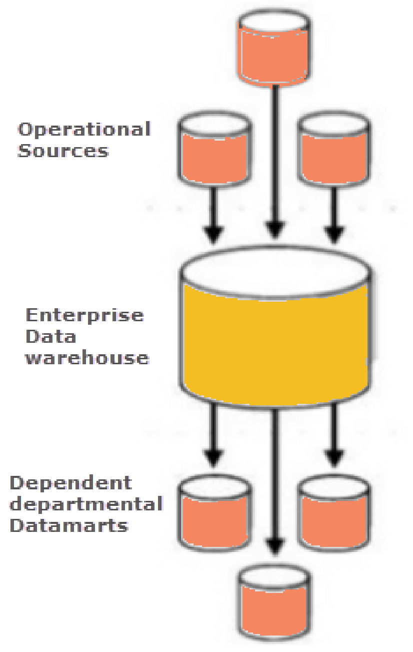
Data Mart usually draws data from only a few sources compared to a Data warehouse. Data marts are small in size and are more flexible compared to a Data warehouse.

Type of Data Mart

There are three main types of data marts are:

1. **Dependent**: Dependent data marts are created by drawing data directly from operational, external or both sources.
2. **Independent**: Independent data mart is created without the use of a central data warehouse.
3. **Hybrid**: This type of data marts can take data from data warehouses or operational systems.

Dependent Data Mart



A dependent data mart allows sourcing organization's data from a single Data Warehouse. It offers the benefit of centralization. If you need to develop one or more physical data marts, then you need to configure them as dependent data marts.

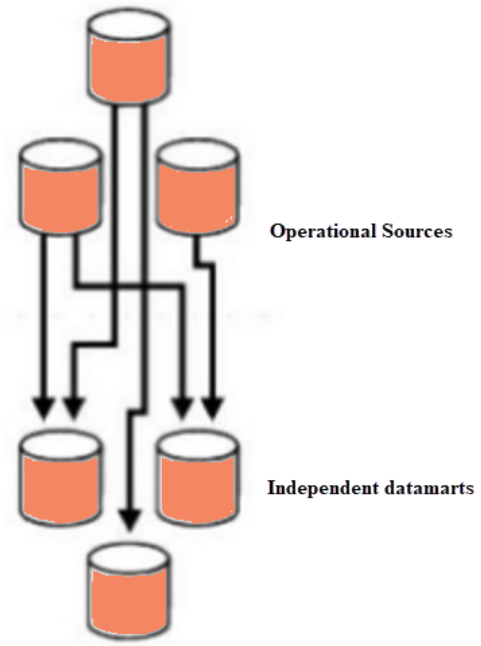
Dependent data marts can be built in two different ways. Either where a user can access both the data mart and data warehouse, depending on need, or where access is limited only to the data mart. The second approach is not optimal as it produces sometimes referred to as a data junkyard. In the data junkyard, all data begins with a common source, but they are scrapped, and mostly junked.

Independent Data Mart

An independent data mart is created without the use of central Data warehouse. This kind of Data Mart is an ideal option for smaller groups within an organization.

An independent data mart has neither a relationship with the enterprise data warehouse nor with any other data mart. In Independent data mart, the data is input separately, and its analyses are also performed autonomously.

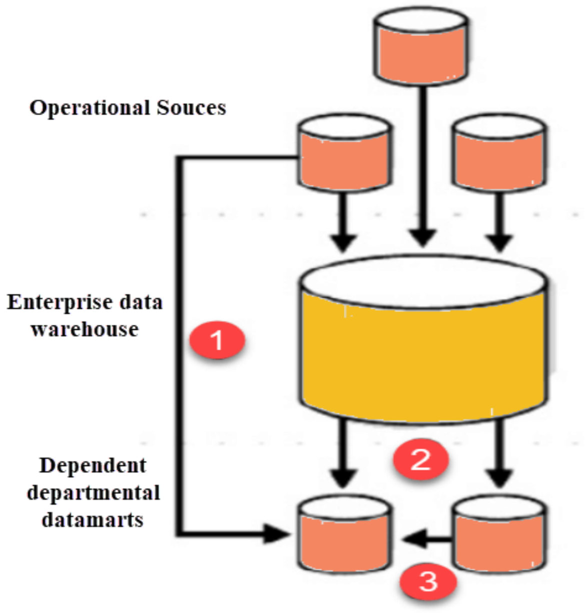
Implementation of independent data marts is antithetical to the motivation for building a data warehouse. First of all, you need a consistent, centralized store of enterprise data which can be analyzed by multiple users with different interests who want widely varying information.



Hybrid data Mart:

A hybrid data mart combines input from sources apart from Data warehouse. This could be helpful when you want ad-hoc integration, like after a new group or product is added to the organization.

It is best suited for multiple database environments and fast implementation turnaround for any organization. It also requires least data cleansing effort. Hybrid Data mart also supports large storage structures, and it is best suited for flexible for smaller data-centric applications.



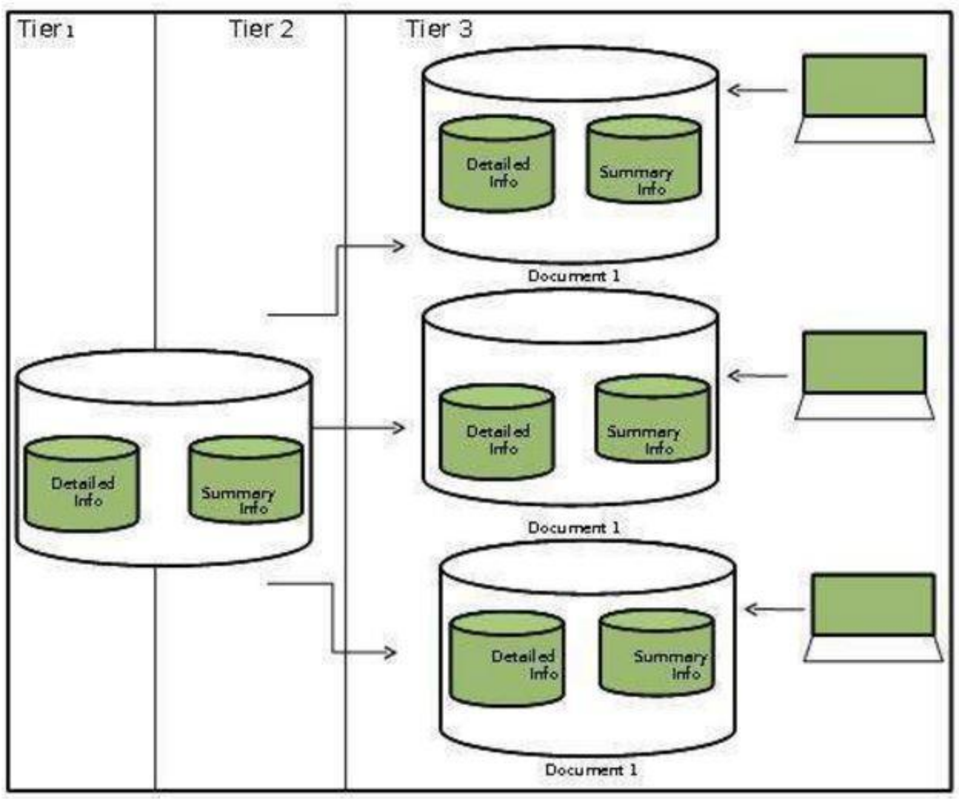
Benefits of Data Marts

- Allows the data to be accessed in lesser time
- Cost-efficient alternative to the bulky data warehouse
- Easy to use as designed according to the needs of specific user group
- Fastens the business processes.

<u>Design Data Mart</u>

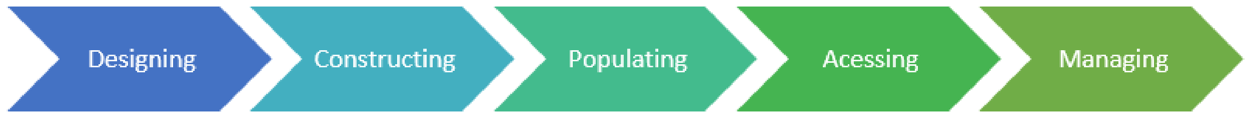
Data Mart should be a designed as smaller version of snowflake schema, within the data warehouse and should match the database design of data warehouse.

It helps maintaining control over database instances.



The summary table helps to summarize the data mart in the same way of data warehouse.

Steps in Implementing a Data Mart



Simply stated, the major steps in building a data mart are:

- Designing
- Constructing
- Populating
- Accessing
- Managing

- **Designing**

Designing is the first phase of Data Mart implementation. It covers all the tasks between initiating the request for a data mart to gathering information about the requirements. Finally, we create the logical and physical design of the data mart.

The design step involves the following tasks:

- Gathering the business & technical requirements and Identifying data sources.
- Selecting the appropriate subset of data.
- Designing the logical and physical structure of the data mart.

Data could be partitioned based on following criteria:

- Date
- Business or Functional Unit
- Geography
- Any combination of above

Data could be partitioned at the application or DBMS level. Though it is recommended to partition at the Application level as it allows different data models each year with the change in business environment.

- **Constructing**

This is the second phase of implementation. It involves creating the physical database and the logical structures.

This step involves the following tasks:

- Implementing the physical database designed in the earlier phase. For instance, database schema objects like table, indexes, views, etc. are created.
- **Storage management:** An RDBMS stores and manages the data to create, add, and delete data.
- **Fast data access:** With a SQL query you can easily access data based on certain conditions/filters.
- **Data protection:** The RDBMS system also offers a way to recover from system failures such as power failures. It also allows restoring data from these backups incase of the disk fails.

- **Multiuser support:** The data management system offers concurrent access, the ability for multiple users to access and modify data without interfering or overwriting changes made by another user.
- **Security:** The RDMS system also provides a way to regulate access by users to objects and certain types of operations.

➢ **Populating:**

In the third phase, data in populated in the data mart.

The populating step involves the following tasks:

- Source data to target data Mapping
- Extraction of source data
- Cleaning and transformation operations on the data
- Loading data into the data mart
- Creating and storing metadata

➢ **Accessing**

Accessing is a fourth step which involves putting the data to use: querying the data, creating reports, charts, and publishing them. End-user submit queries to the database and display the results of the queries

The accessing step needs to perform the following tasks:

- Set up a Meta layer that translates database structures and objects names into business terms. This helps non-technical users to access the Data mart easily.
- Set up and maintain database structures.
- Set up API and interfaces if required

➢ **Managing**

This is the last step of Data Mart Implementation process. This step covers management tasks such as-

- Ongoing user access management.
- System optimizations and fine-tuning to achieve the enhanced performance.
- Adding and managing fresh data into the data mart.
- Planning recovery scenarios and ensure system availability in the case when the system fails.

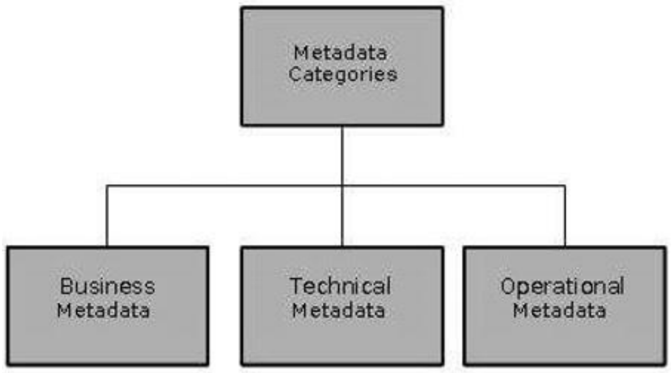
Metadata

Metadata is simply defined as data about data which defines the data warehouse. It is used for building, maintaining and managing the data warehouse. The data that is used to represent other data is known as metadata.

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

For example, the index of a book serves as a metadata for the contents in the book. In other words, we can say that metadata is the summarized data that leads us to detailed data. In terms of data warehouse, we can define metadata as follows.

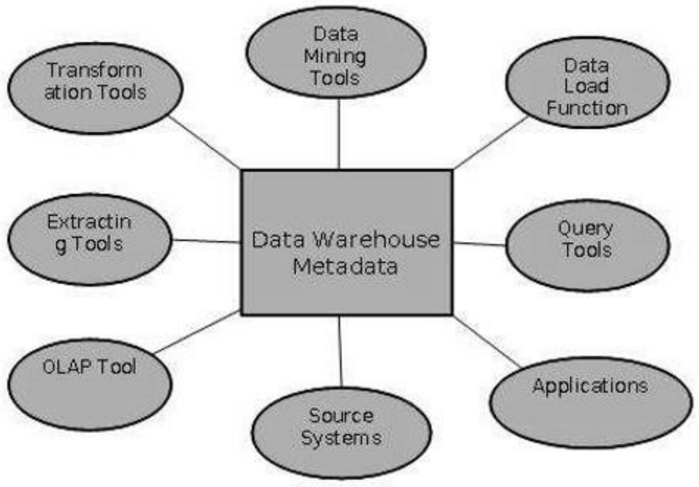
- Metadata is the road-map to a data warehouse.
- Metadata in a data warehouse defines the warehouse objects.
- Metadata acts as a directory. This directory helps the decision support system to locate the contents of a data warehouse.



Categories of Metadata

Metadata can be broadly categorized into three categories −

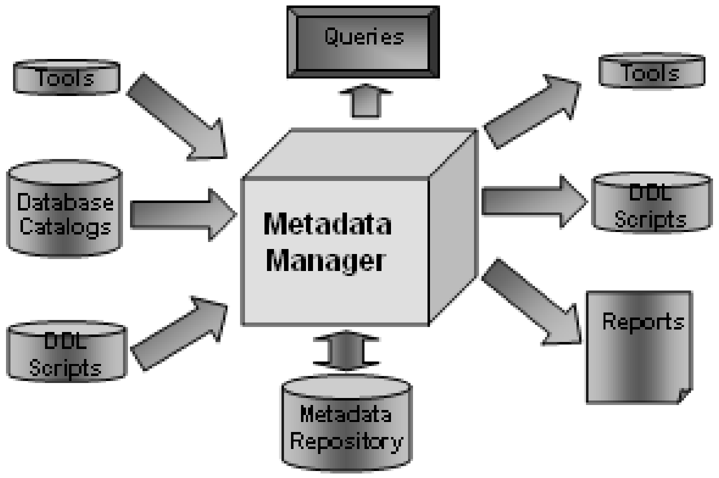
- **Business Metadata** − It has the data ownership information, business definition, and changing policies.
- **Technical Metadata** − It includes database system names, table and column names and sizes, data types and allowed values. Technical metadata also includes structural information such as primary and foreign key attributes and indices.
- **Operational Metadata** − It includes currency of data and data lineage. Currency of data means whether the data is active, archived, or purged. Lineage of data means the history of data migrated and transformation applied on it.



Role of Metadata

Metadata has a very important role in a data warehouse. The role of metadata in a warehouse is different from the warehouse data, yet it plays an important role. The various roles of metadata are explained below.

- Metadata acts as a directory.
- This directory helps the decision support system to locate the contents of the data warehouse.
- Metadata helps in decision support system for mapping of data when data is transformed from operational environment to data warehouse environment.
- Metadata helps in summarization between current detailed data and highly summarized data.
- Metadata also helps in summarization between lightly detailed data and highly summarized data.
- Metadata is used for query tools.
- Metadata is used in extraction and cleansing tools.
- Metadata is used in reporting tools.
- Metadata is used in transformation tools.
- Metadata plays an important role in loading functions.



Metadata Repository

Metadata repository is an integral part of a data warehouse system. It has the following metadata-

- **Definition of data warehouse** − It includes the description of structure of data warehouse. The description is defined by schema, view, hierarchies, derived data definitions, and data mart locations and contents.
- **Business metadata** − It contains has the data ownership information, business definition, and changing policies.
- **Operational Metadata** − It includes currency of data and data lineage. Currency of data means whether the data is active, archived, or purged. Lineage of data means the history of data migrated and transformation applied on it.
- **Data for mapping from operational environment to data warehouse** − It includes the source databases and their contents, data extraction, data partition cleaning, transformation rules, data refresh and purging rules.
- **Algorithms for summarization** − It includes dimension algorithms, data on granularity, aggregation, summarizing, etc.

Data Transformation

Data transformation is the process of converting data from one format or structure into another format or structure.

Transformation refers to the cleansing and aggregation that may need to happen to data to prepare it for analysis. Architecturally speaking, there are two ways to approach in transformation:

- **Multistage data transformation –** This is the classic extract, transform, load process. Extracted data is moved to a staging area where transformations occur prior to loading the data into the warehouse.
- **In-warehouse data transformation –** In this approach, the process flow changes to something. Data is extracted and loaded into the analytics warehouse, and transformations are done there.

Transformation types

Regardless of where in the process transformation takes place, it's an important step in the analytic workflow. Transformations prepare the data for analysis. Here are some of the most common types:

- **Basic transformations:**
  - **Cleaning:** Mapping NULL to 0 or "Male" to "M" and "Female" to "F," date format consistency, etc.
  - **Deduplication:** Identifying and removing duplicate records
  - **Format revision:** Character set conversion, unit of measurement conversion, date/time conversion, etc.
  - **Key restructuring:** Establishing key relationships across tables

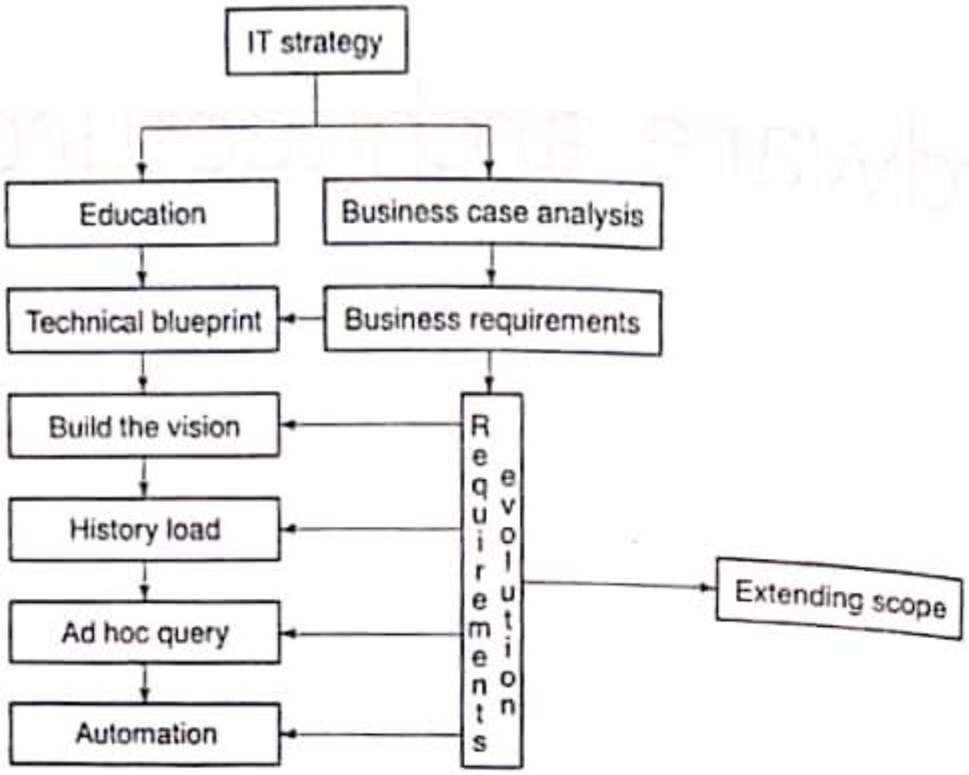
- **Advanced transformations:**

  Derivation applying business rules to your data that derive new calculated values from existing data – for example, creating a revenue metric that subtracts taxes

  - **Filtering:** Selecting only certain rows and/or columns
  - **Joining:** Linking data from multiple sources – for example, adding ad spend data across multiple platforms, such as Google Adwords and Facebook Ads
  - **Splitting:** Splitting a single column into multiple columns
  - **Data validation:** Simple or complex data validation – for example, if the first three columns in a row are empty then reject the row from processing
  - **Summarization:** Values are summarized to obtain total figures which are calculated and stored at multiple levels as business metrics – for example, adding up all purchases a customer has made to build a customer lifetime value (CLV) metric
  - **Aggregation:** Data elements are aggregated from multiple data sources and databases
  - **Integration:** Give each unique data element one standard name with one standard definition. Data integration reconciles different data names and values for the same data element.

Hardware Architecture

Process- The hardware architecture of a data warehouse is defined within the technical blueprint stage of the process. The business requirements stage should have identified the initial user requirements and given an indication of the capacity- planning requirements. The hardware architecture is determined once a broad understanding of the required technical architecture has been achieved. The backup and security strategies are also determined during the technical blueprint phase.



Stages in the Process

Server Hardware

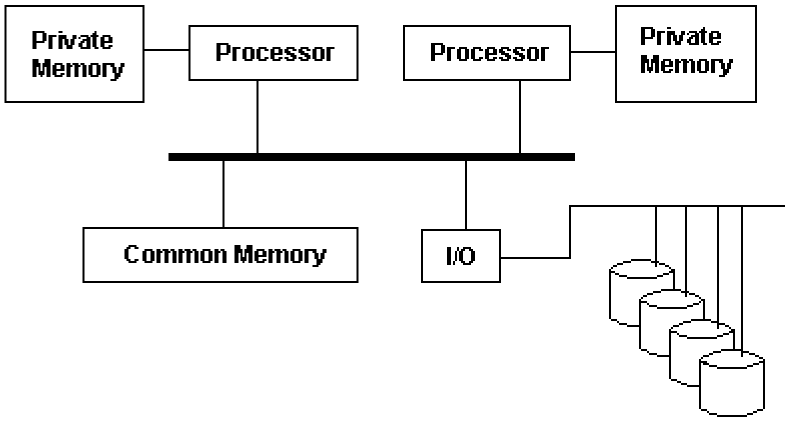
The server is a crucial part of the data warehouse environment. To support the size of the database, and the ad hoc nature of the query access, warehouse applications generally require large hardware configurations. There are a number of different hardware architectures in the open systems market, each of which has its own advantages and disadvantages. The different architectures are discussed below.

Architecture Options

There are two main hardware architectures commonly used as server platforms in data warehousing solutions: symmetric multi-processing (SMP), and massively parallel processing (MPP). There is a lot of confusion about the distinction between these architectures, and this is not helped by the existence of hybrid machines that use both.

The primary distinguishing feature between SMP and MPP is as follows. An SMP machine is a set of tightly coupled CPUs that share memory and disk. An MPP machine is a set of loosely coupled CPUs, each of which has its own memory and disk.

Symmetric Multi-Processing

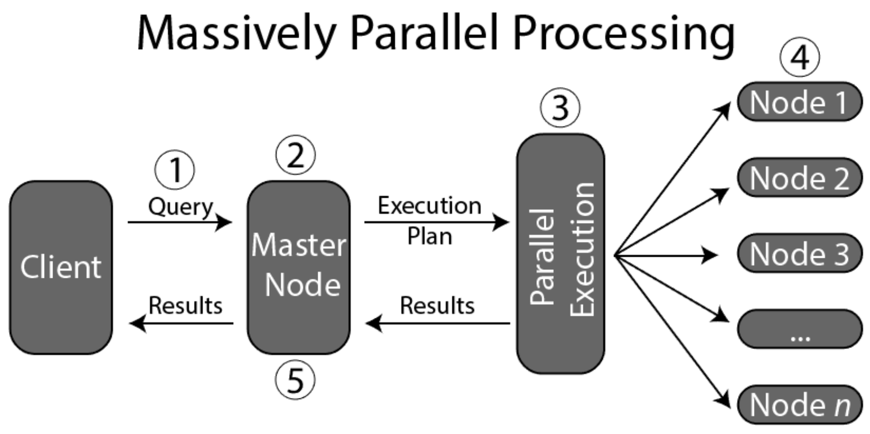


An SMP machine is a set of CPUs that share memory and disk. This is sometimes called a **shared-everything environment.** Unlike some of the earlier multi-CPU machines, where there existed a master CPU with a number of slave CPUs, the CPU in an SMP machine are all equal. Having multiple CPUs available allows operations to be processed in parallel.

Cluster Technology

A cluster is a set of loosely coupled SMP machines connected by a high-speed interconnect. Each machine has its own CPUs and memory, but they share access to disk. Thus these systems are called **shared-disk systems.** Each machine in the cluster is called a **node.** The aim of the cluster is to mimic a single larger machine. In this pseudo single machine, resources such as shared disk must be managed in a distributed fashion. A layer of software called the **distributed lock manager** is used to achieve this.

Massively Parallel Processing



An MPP machine is made up of many loosely coupled nodes. These nodes will be linked together by a high-speed connection. Each node has its own memory, and the disks are not shared, each being attached to only one node. However, most MPP systems allow a disk to be dual connected between two nodes. This protects against an individual node failure causing disks to be unavailable.

Like a cluster, MPP machines require the use of a distributed lock manager to maintain the integrity of the distributed resources across the system as a whole.

The advantage of MPP systems is that because nothing is shared they do not suffer the same restrictions as SMP and cluster systems.

New and Emerging Technologies

Non uniform memory architecture (NUMA) machines are not a new concept. A NUMA machine is basically a tightly coupled cluster of SMP nodes, with an extremely high-speed interconnect. The interconnect needs to be sufficiently fast to give near-SMP internal speeds.

Another new technology is the **high-speed memory interconnect**, such as the memory channel provided by Digital on its clustered UNIX systems. The memory channel allows a cluster of SMP nodes to act more as though it has one memory address space.

Both of these new technologies overcome bottlenecks in the current cluster systems, thereby allowing much greater scalability of disk, memory and CPU on a clustered system.

Server Management

The systems required for data warehouse environments are generally large and complex. This, added to the changing nature of a warehouse, means that these systems require a lot of management. For systems and database administrators to manage effectively, they require the use of one or more of the increasing number of management and monitoring tools on the market.

These tools allow the automatic monitoring of most if not all the required processes and statistics. Events such as:

- running out of space on certain key disks,
- a process dying,
- a process using excessive resource (such as CPU),
- a process returning an error,
- disks exhibiting I/O bottlenecks,
- hardware failure
- a table failing to extend because of lack of space,
- CPU usage exceeding an 80% threshold,
- excessive memory swapping,
- low buffer cache hit ratios,

can be caught and in many cases fixed automatically. This takes much of the sheer drudgery out of system management, allowing the system administrators to get on with more important, but less immediately critical, work.

Network Hardware

The network, although not part of the data warehouse itself, can play an important part in a data warehouse's success.

Network Architecture

As long as the network has sufficient bandwidth to supply the data feed and user requirements, the architecture of the network is irrelevant. It is, however, important at the design stage to consider some aspects of the network architecture.

The main aspects of a data warehouse design that may be affected by the network architecture are:

- user access,
- sources system data transfer,
- data extractions,

Each of these issues needs to be considered carefully.

Data extractions are any requests that cause data to be transferred out over the network.

Network Management

Network management is a black art. It requires specialist tools and lots of network experience. The management of the network has no direct effect on a data warehouse, except for one issue. It is important to be able to monitor network performance. The network may play a key part in data flow through a data warehouse environment. Being able to monitor this part of the data flow is necessary to enable resolution of any performance problems.

Client Hardware

As with the network, clients are external to the data warehouse system itself. There are, however, still aspects that need to be considered during the design phase.

Client Management

Management of the clients is beyond the scope of the data warehouse environment. The dependencies here are the other way around. These responsible for client machine management will need to know the requirements for that machine to access the data warehouse system. Details such as the network protocols supported on the server, and the server's Internet address, will need to be supplied.

If multiple access paths to the server system exist, this information needs to be relayed to those responsible for the client systems.

Client Tools

At the design stage it may be necessary to consider what user-side tools will be used. If these tools have special requirements, such as data being summarized in certain ways, these requirements need to be catered for. Even given this requirement, it is important that the data warehouse be designed to be accessible to any tool. In fact the tool should not be allowed to affect the basic design of the warehouse itself. This protects against changing tools requirements, and is particularly important in the data warehouse arena, where requirements evolve over time.

No one tool is likely to meet all users' requirements, and it is probable that multiple tools will be used against the data warehouse. The tools should be thoroughly tested and trialed to ensure that they are suitable for the users. This testing of the tools should ideally be performed in parallel with the data warehouse design. This will allow any usability issues to be exposed, and will also help to drive out any requirements that the tool will place on the data warehouse.

Data Warehouse Database

The central data warehouse database is the cornerstone of the data warehousing environment. This database is almost always implemented on the relational database management system (RDBMS) technology. However, this kind of implementation is often constrained by the fact that traditional RDBMS products are optimized for transactional database processing. Certain data warehouse attributes, such as very large database size, ad hoc query processing and the need for flexible user view creation including aggregates, multi-table joins and drill-downs, have become drivers for different technological approaches to the data warehouse database. These approaches include:

- Parallel relational database designs for scalability that include shared-memory, shared disk, or shared-nothing models implemented on various multiprocessor configurations (symmetric
- Multiprocessors or SMP, massively parallel processors or MPP, and/or clusters of uni- or multiprocessors).
- An innovative approach to speed up a traditional RDBMS by using new index structures to bypass relational table scans.
- Multidimensional databases (MDDBs) that are based on proprietary database technology; conversely, a dimensional data model can be implemented using a familiar RDBMS. Multi-dimensional databases are designed to overcome any limitations placed on the warehouse by the nature of the relational data model. MDDBs enable on-line analytical processing (OLAP) tools that architecturally belong to a group of data warehousing components jointly categorized as the data query, reporting, analysis and mining tools.

Database Structure

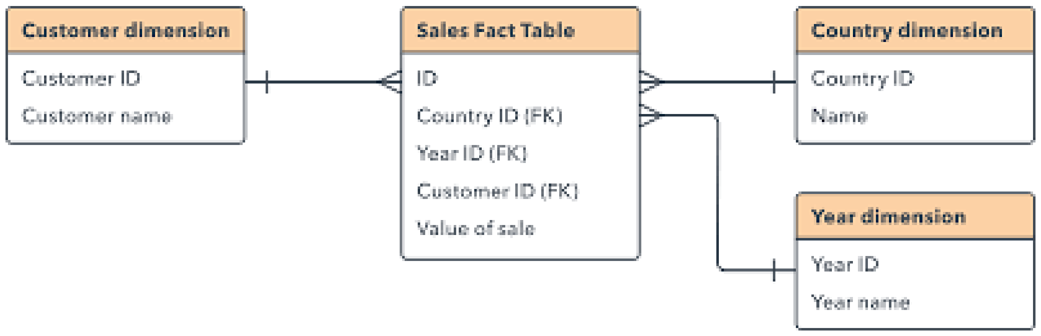
A database is an organized collection of data. Instead of having all the data in a list with a random order, a database provides a structure to organize the data. One of the most common data structures is a database table. A database table consists of rows and columns. A database table is also called a two-dimensional array. An array is like a list of values, and each value is identified by a specific index. A two-dimensional array uses two indices, which correspond to the rows and columns of a table.

In database terminology, each row is called a record. A record is also called an object or an entity. In other words, a database table is a collection of records. The records in a table are the objects you are interested in, such as the books in a library catalog or the customers in a sales database. A field corresponds to a column in the table and represents a single value for each record. A field is also called an attribute. In other words, a record is a collection of related attributes that make up a single database entry.

The database structure imposes certain constraints on the data values, which makes it more reliable. For example, for the phone number, you cannot enter text, since that wouldn't make sense.

While this example is quite simple, you can easily imagine what else could be stored in such a database. For example, you could store the customer's mailing address, billing information, history of past purchases, etc. For an organization with many thousands of customers, this quickly becomes a large database. To use a large database effectively, you can use a database management system (DBMS). A DBMS is specialized software to input, store, retrieve and manage all the data.

<u>**Database Layout**</u>

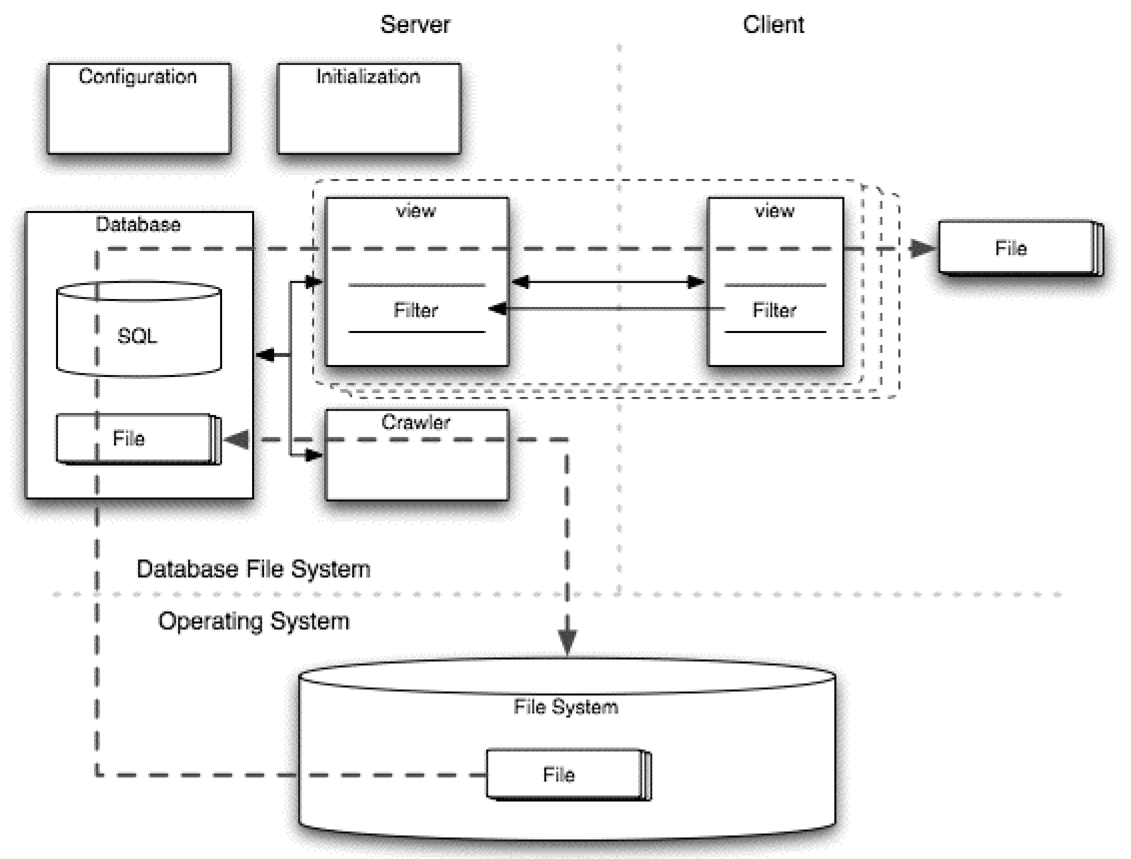


Database layout is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model.

The database layout is also affected because when security measures are implemented, there is an increase in the number of views and tables. Adding security increases the size of the database and hence increases the complexity of the database design and management. It will also add complexity to the backup management and recovery plan.

Database layout involves classifying data and identifying interrelationships. This theoretical representation of the data is called an ontology. The ontology is the theory behind the database's design.

Database File System (DBFS)



Database File System (DBFS) creates a standard file system interface on top of files and directories that are stored in database tables.

Database File System (DBFS) creates a standard file system interface using a server and clients.

It is a new type of file system that does away with places where you store your files.

And while being precise, it is not database system either, it is a faceted system.

It supports 'locating' files the way you think about them.

The searches are very fast, which gives direct feedback when browsing your files. And instead of directories the DBFS uses keywords. Keywords work somewhat like directories, but they can do much more. The keywords, as the DBFS uses them, are like a superset of directories.