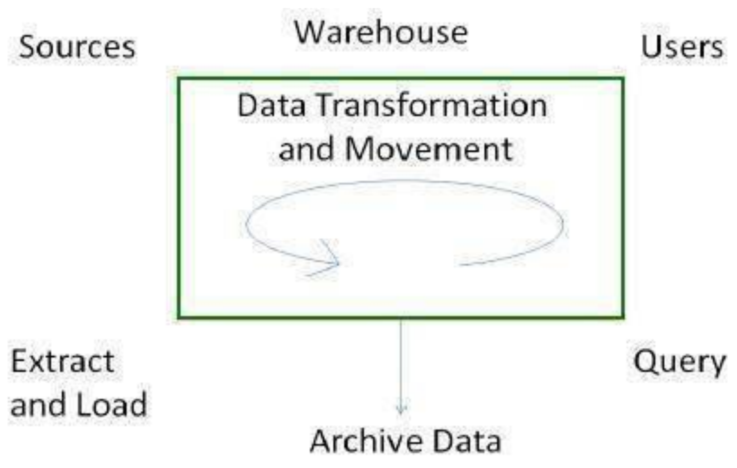
System Processes

Process Flow in Data Warehouse

There are four major processes that contribute to a data warehouse −

- Extract and load the data.
- Cleaning and transforming the data.
- Backup and archive the data.
- Managing queries and directing them to the appropriate data sources.



Extract and Load Process

Data extraction takes data from the source systems. Data load takes the extracted data and loads it into the data warehouse.

Controlling the Process

Controlling the process involves determining when to start data extraction and the consistency check on data. Controlling process ensures that the tools, the logic modules, and the programs are executed in correct sequence and at correct time.

When to Initiate Extract

Data needs to be in a consistent state when it is extracted, i.e., the data warehouse should represent a single, consistent version of the information to the user.

For example, in a customer profiling data warehouse in telecommunication sector, it is illogical to merge the list of customers at 8 pm on Wednesday from a customer database with the customer subscription events up to 8 pm on Tuesday. This would mean that we are finding the customers for whom there are no associated subscriptions.

Loading the Data

After extracting the data, it is loaded into a temporary data store where it is cleaned up and made consistent.

Clean and Transform Process

Once the data is extracted and loaded into the temporary data store, it is time to perform Cleaning and Transforming. Here is the list of steps involved in Cleaning and Transforming −

- Clean and transform the loaded data into a structure
- Partition the data
- Aggregation

Clean and Transform the Loaded Data into a Structure

Cleaning and transforming the loaded data helps speed up the queries. It can be done by making the data consistent −

- within itself.
- with other data within the same data source.
- with the data in other source systems.
- with the existing data present in the warehouse.

Transforming involves converting the source data into a structure. Structuring the data increases the query performance and decreases the operational cost. The data contained in a data warehouse must be transformed to support performance requirements and control the ongoing operational costs.

Partition the Data

It will optimize the hardware performance and simplify the management of data warehouse. Here we partition each fact table into multiple separate partitions.

Aggregation

Aggregation is required to speed up common queries. Aggregation relies on the fact that most common queries will analyze a subset or an aggregation of the detailed data.

Backup and Archive the Data

In order to recover the data in the event of data loss, software failure, or hardware failure, it is necessary to keep regular backups. Archiving involves removing the old data from the system in a format that allow it to be quickly restored whenever required.

Query Management Process

This process performs the following functions −

- manages the queries.
- helps speed up the execution time of queries.
- directs the queries to their most effective data sources.
- ensures that all the system sources are used in the most effective way.
- monitors actual query profiles.

The information generated in this process is used by the warehouse management process to determine which aggregations to generate. This process does not generally operate during the regular load of information into data warehouse.

Process Architecture:-

1. **Load Manager**

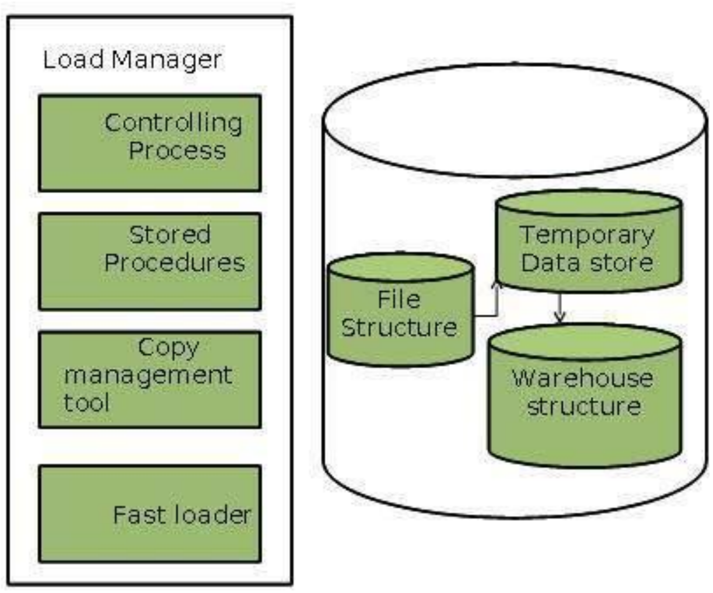
This component performs the operations required to extract and load process.

The size and complexity of the load manager varies between specific solutions from one data warehouse to other.

Load Manager Architecture

The load manager performs the following functions:

- Extract the data from source system.
- Fast Load the extracted data into temporary data store.
- Perform simple transformations into structure similar to the one in the data warehouse.



Extract Data from Source

The data is extracted from the operational databases or the external information providers. Gateways is the application programs that are used to extract data. It is supported by underlying DBMS and allows client program to generate SQL to be executed at a server. Open Database Connection (ODBC), Java Database Connection (JDBC), are examples of gateway.

Fast Load

- In order to minimize the total load window the data need to be loaded into the warehouse in the fastest possible time.
- The transformations affects the speed of data processing.
- It is more effective to load the data into relational database prior to applying transformations and checks.
- Gateway technology proves to be not suitable, since they tend not be performant when large data volumes are involved.

Simple Transformations

While loading it may be required to perform simple transformations. After this has been completed we are in position to do the complex checks. Suppose we are loading the EPOS sales transaction we need to perform the following checks:

- Strip out all the columns that are not required within the warehouse.
- Convert all the values to required data types.

2. **Warehouse Manager**

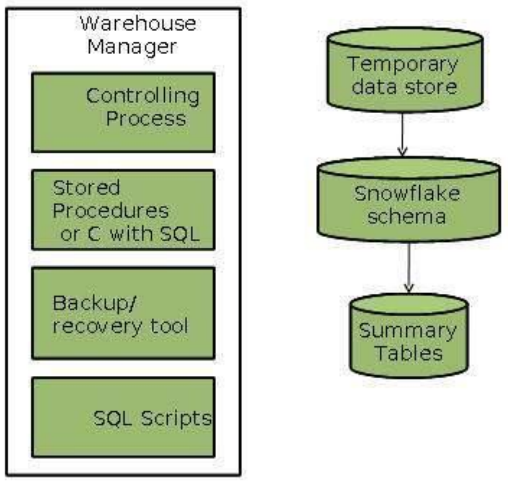
A warehouse manager is responsible for the warehouse management process. It consists of third-party system software, C programs, and shell scripts.

The size and complexity of warehouse managers varies between specific solutions.

Warehouse Manager Architecture

A warehouse manager includes the following:

- The controlling process
- Stored procedures or C with SQL
- Backup/Recovery tool
- SQL Scripts



Operations Performed by Warehouse Manager

- A warehouse manager analyzes the data to perform consistency and referential integrity checks.
- Creates indexes, business views, partition views against the base data.
- Generates new aggregations and updates existing aggregations. Generates normalizations.
- Transforms and merges the source data into the published data warehouse.
- Backup the data in the data warehouse.
- Archives the data that has reached the end of its captured life.

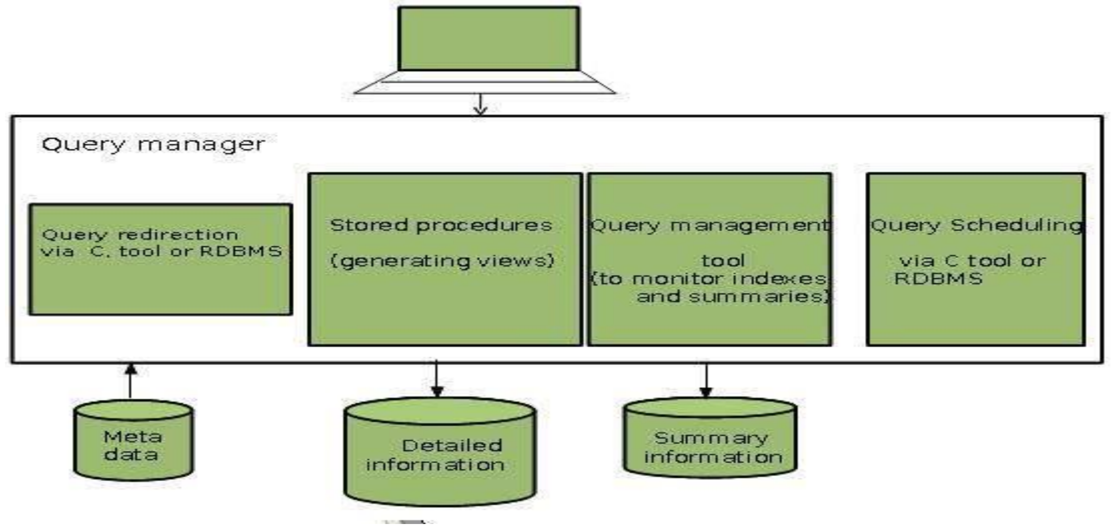
3. **Query Manager**

- Query manager is responsible for directing the queries to the suitable tables.
- By directing the queries to appropriate tables, the speed of querying and response generation can be increased.
- Query manager is responsible for scheduling the execution of the queries posed by the user.

Query Manager Architecture

The following screenshot shows the architecture of a query manager. It includes the following:

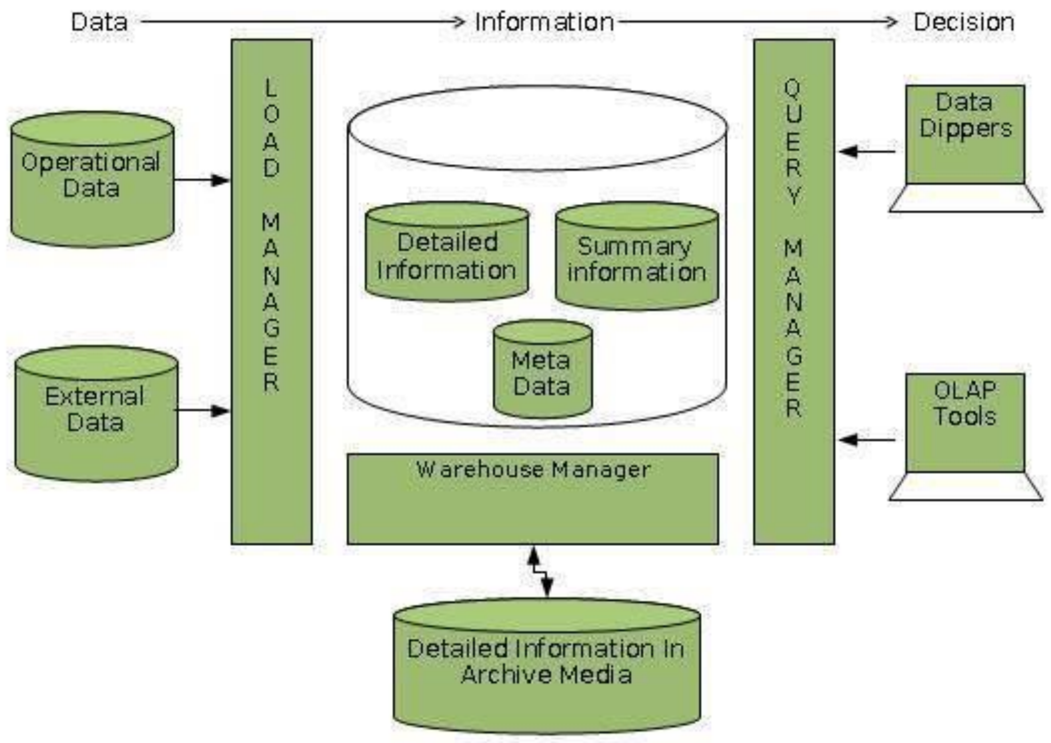
- Query redirection via C tool or RDBMS
- Stored procedures
- Query management tool
- Query scheduling via C tool or RDBMS
- Query scheduling via third-party software


Query manager
Query redirection via C tool or RDBMS
Stored procedures (generating views)
Query management tool (to monitor indexes and summaries)
Query Scheduling via C tool or RDBMS
Meta data
Detailed information
Summary information

Detailed Information

Detailed information is not kept online, rather it is aggregated to the next level of detail and then archived to tape. The detailed information part of data warehouse keeps the detailed information in the star flake schema. Detailed information is loaded into the data warehouse to supplement the aggregated data.

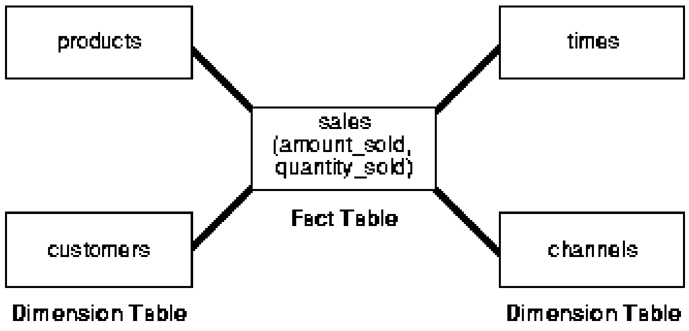
The following diagram shows a pictorial impression of where detailed information is stored and how it is used.



Database Schema

A schema is a collection of database objects, including tables, views, indexes, and synonyms. Schema is a logical description of the entire database. It includes the name and description of records of all record types including all associated data-items and aggregates. Much like a database, a data warehouse also requires to maintain a schema. A database uses relational model, while a data warehouse uses Star, Snowflake, Galaxy Schema, and Fact Constellation schema.

Star Schema

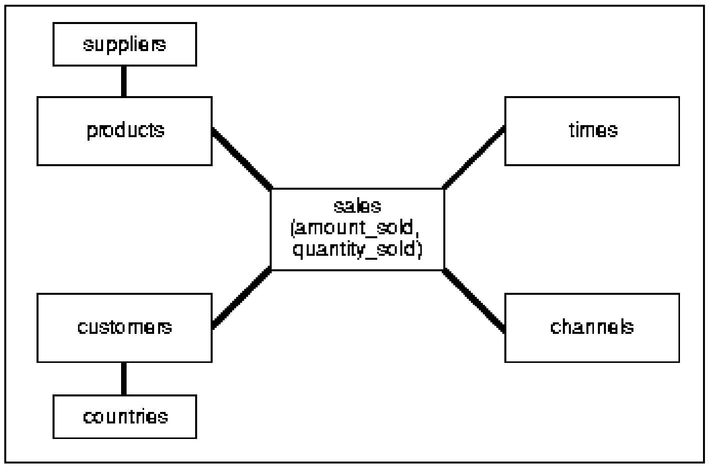


Star Schema is a relational database schema for representing multidimensional data. It is the simplest form of data warehouse schema that contains one or more dimensions and fact tables. It is called a star schema because the entity-relationship diagram between dimensions and fact tables resembles a star where one fact table is connected to multiple dimensions. The center of the star schema consists of a large fact table, and it points towards the dimension tables. The advantage of star schema are slicing down, performance increase and easy understanding of data.

Steps in designing Star Schema

- Identify a business process for analysis (like sales).
- Identify measures or facts (sales dollar).
- Identify dimensions for facts (product dimension, location dimension, time dimension, organization dimension).
- List the columns that describe each dimension. (region name, branch name, region name).
- Determine the lowest level of summary in a fact table (sales dollar).

Snowflake Schema



Snowflake schema is an extension of star schema means it is more complex than star schema. It is called as a snowflake schema the diagram resembles a snowflake. In star schema each dimension is represented by a single dimension table whereas in snowflake schema each dimension is grouped into multiple lookup table to eliminate the redundancy.

Steps in designing Snowflake Schema

- Identify a business process for analysis(like sales).
- Identify measures or facts (sales dollar).
- Identify dimensions for facts(product dimension, location dimension, time dimension, organization dimension).
- List the columns that describe each dimension.(region name, branch name, region name).
- Determine the lowest level of summary in a fact table(sales dollar)

Identifying Fact and Dimension Tables

When designing a data warehouse, you will need to identify which of your tables and views represent facts (containing numerical values such as sales, revenue, or budget figures), and which dimensions (providing ways of aggregating these figures, such as by region, date, customer, or product).

Fact Table

The centralized table in a star schema is called as the FACT table. Fact table also called as measure table. It is located center of a snowflake or star schema and surrounded by dimensions. Fact table mostly have foreign keys to dimension table. It consist of two types of columns one of those facts and another one of foreign keys. For example sales fact table may contain sales quantity, sales cost etc.

Dimension tables

Dimension tables is usually descriptive information that qualifies a fact. Dimensions are also part of star schema or snowflake. They are surrounded to fact table to make star like design structure and their table rows are uniquely identified by a single key field. Dimensions are nothing but reference which speaks about fact table. For example, each record in a product dimension represents a specific product.

Defining Fact and Dimension Tables

Designing a dimensional data mart for any organization requires an understanding of business rules and the content of data that is collected. Ultimately, you need to match the needs of the users with the realities of the accessible data.

The following points summarize the process of designing a schema for a decision-support database:

1. Choose a business process to model in order to identify the fact tables.

This is a major operational process in the organization that generates raw data. Users store and retrieve data in an existing system, such as a legacy system or company-wide data warehouse. In the diagram in this chapter, the business process is sales. Examples of other business processes are orders, invoices, shipments, inventory, and general ledger.

2. Choose the granularity of each fact table.

This is the most detailed level of data to include in the fact table for the business process. The finer the granularity of each dimension, the more precisely a query can get through the database. The granularity of a fact table is usually the individual transaction, the individual line item, a daily snapshot, or a monthly snapshot.

3. Choose the dimensions for each fact table and their respective granularity.

Examples of typical dimensions are time, product, customer, promotion, transaction type, and status. For each dimension, identify all the distinct attributes that describe the dimension. The most useful attributes are textual and discrete. A dimension can include numeric attributes, such as package size, if the attribute acts more like a description than a measurement.

4. Choose the measured facts.

These are the measurements that make up each fact table record. Typical measured facts are numeric additive quantities, such as sales in dollars or sales in units.

Multidimensional Schemas

Multidimensional schema is especially designed to model data warehouse systems. The schemas are designed to address the unique needs of very large databases designed for the analytical purpose (OLAP).

Multidimensional schema is defined using Data Mining Query Language (DMQL). The two primitives, cube definition and dimension definition, can be used for defining the data warehouses and data marts.

Hardware Partitioning

Partitioning is done to enhance performance and facilitate easy management of data. Partitioning also helps in balancing the various requirements of the system. It optimizes the hardware performance and simplifies the management of data warehouse by partitioning each fact table into multiple separate partitions.

Why is it Necessary to Partition?

Partitioning is important for the following reasons −

- For easy management,
- To assist backup/recovery,
- To enhance performance.

For Easy Management

The fact table in a data warehouse can grow up to hundreds of gigabytes in size. This huge size of fact table is very hard to manage as a single entity. Therefore it needs partitioning.

To Assist Backup/Recovery

If we do not partition the fact table, then we have to load the complete fact table with all the data. Partitioning allows us to load only as much data as is required on a regular basis. It reduces the time to load and also enhances the performance of the system.

To Enhance Performance

By partitioning the fact table into sets of data, the query procedures can be enhanced. Query performance is enhanced because now the query scans only those partitions that are relevant. It does not have to scan the whole data.

Horizontal Partitioning

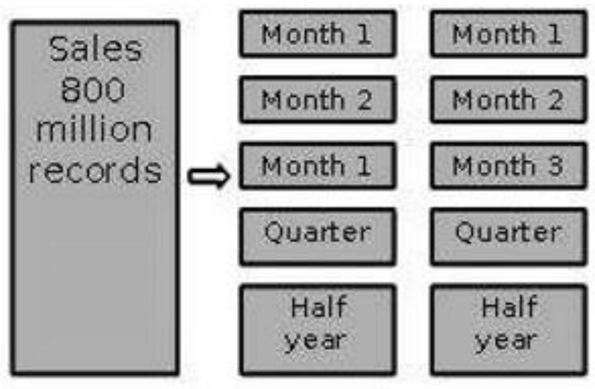
There are various ways in which a fact table can be partitioned. In horizontal partitioning, we have to keep in mind the requirements for manageability of the data warehouse.

❖ **Partitioning by Time into Equal Segments**

In this partitioning strategy, the fact table is partitioned on the basis of time period. Here each time period represents a significant retention period within the business. For example, if the user queries for **month to date data** then it is appropriate to partition the data into monthly segments. We can reuse the partitioned tables by removing the data in them.

❖ **Partition by Time into Different-sized Segments**

This kind of partition is done where the aged data is accessed infrequently. It is implemented as a set of small partitions for relatively current data, larger partition for inactive data.



❖ **Partition on a Different Dimension**

The fact table can also be partitioned on the basis of dimensions other than time such as product group, region, supplier, or any other dimension. Let's have an example.

Suppose a market function has been structured into distinct regional departments like on a **state by state** basis. If each region wants to query on information captured within its region, it would prove to be more effective to partition the fact table into regional partitions. This will cause the queries to speed up because it does not require to scan information that is not relevant.

❖ **Partition by Size of Table**

When there are no clear basis for partitioning the fact table on any dimension, then we should **partition the fact table on the basis of their size.** We can set the predetermined size as a critical point. When the table exceeds the predetermined size, a new table partition is created.

❖ **Partitioning Dimensions**

If a dimension contains large number of entries, then it is required to partition the dimensions. Here we have to check the size of a dimension.

Consider a large design that changes over time. If we need to store all the variations in order to apply comparisons, that dimension may be very large. This would definitely affect the response time.

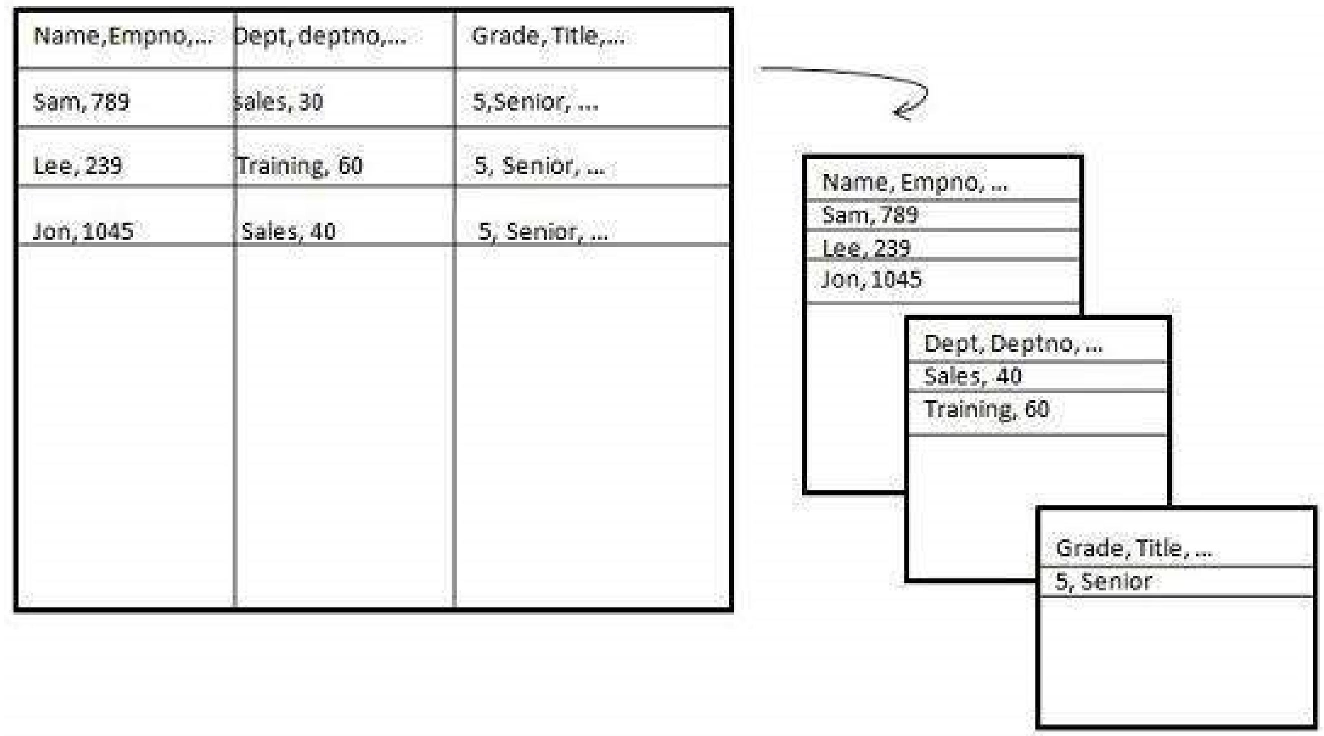
❖ **Round Robin Partitions**

In the round robin technique, when a new partition is needed, the old one is archived. It uses metadata to allow user access tool to refer to the correct table partition.

This technique makes it easy to automate table management facilities within the data warehouse.

Vertical Partition

Vertical partitioning, splits the data vertically. The following images depicts how vertical partitioning is done.



Vertical partitioning can be performed in the following two ways −

- Normalization
- Row Splitting
-

Normalization

Normalization is the standard relational method of database organization. In this method, the rows are collapsed into a single row, hence it reduce space. Take a look at the following tables that show how normalization is performed.

Row Splitting

Row splitting tends to leave a one-to-one map between partitions. The motive of row splitting is to speed up the access to large table by reducing its size.