**DATA MODELLING IN BIG DATA**

**Definition:**

Data modelling in big data is the process of organizing and structuring large volumes of data in a way that makes it easier to store, process, and analyse. It involves creating a blueprint or plan for how the data will be organized, what types of information will be stored, and how different pieces of data relate to each other.

Think of it like organizing a messy room. When you have a lot of items scattered around, it can be challenging to find what you need. But if you organize things into categories, put labels on boxes, and arrange items systematically, it becomes much easier to locate specific things.

**Uses:**

In big data, data modelling helps to bring order to the vast amounts of information. It defines the structure of the data, such as what types of data will be collected, how it will be stored, and the relationships between different data elements. This structure allows data to be efficiently stored and retrieved, and it enables data analysts and data scientists to extract meaningful insights from the data.

By applying data modelling in big data, organizations can gain a better understanding of their data, make informed decisions, and uncover valuable patterns and trends.

**Big data can benefit from appropriate models and storage environments in the following ways:**

Performance: Good data models will help us quickly query the data we need and lower I/O throughput.

Cost: Good data models can help big data systems save money by reducing unnecessary data redundancy, reusing computing results, and lowering storage and computing costs.

Efficiency: Good data models can significantly enhance user experience and data utilization performance.

Quality: Good data models ensure that data statistics are accurate and that computing errors are minimized.

As a result, a big data system unquestionably necessitates high-quality data modelling methods for organizing and storing data, enabling us to achieve the best possible balance of performance, cost, reliability, and quality.

**Data models perspectives / types:**

**Conceptual Model:**

This stage specifies what must be included in the model's configuration to describe and coordinate market principles. It focuses primarily on business-related entries, characteristics, and relationships. Data Architects and Business Stakeholders are mainly responsible for its development.

The Conceptual Data Model is used to specify the scope of the method. It's a tool for organizing, scoping, and visualizing company ideas. The aim of developing a computational data model is to develop new entities, relationships, and attributes. Data architects and stakeholders typically create a computational data model.

The Conceptual Data Model is held by three key holders.

Entity: A real-life thing

Attribute: Properties of an entity

Relationship: Association between two entities

Let's take a look at an illustration of this data model.

A picture containing text, screenshot, font, line

Description automatically generated

Consider the following two entities: product and customer. The Product entity's attributes are the name and price of the product, while the Customer entity's attributes are the name and number of customers. Sales is the connection between these two entities.

The Conceptual Data Model was created with a corporate audience in mind.

It offers an overview of corporate principles for the whole organization.

It is created separately, with hardware requirements such as location and data storage space and software requirements such as technology and DBMS vendor.

**Logical Model**

The conceptual model lays out how the model can be put into use. It encompasses all types of data that must be captured, such as tables, columns, and so on. Business Analysts and Data Architects are the most prominent designers of this model.

The Logical Data Model is used to describe the arrangement of data structures as well as their relationships. It lays the groundwork for constructing a physical model. This model aids in the inclusion of extra data to the conceptual data model components. There is no primary or secondary key specified in this model. This model helps users to update and check the connector information for relationships that have been set previously.

The logical data model describes the data requirements for a single project, but it may be combined with other logical data models depending on the project's scope. Data attributes come with a variety of data types, many of which have exact lengths and precisions.

The logical data model is created and configured separately from the database management system. Data Types with accurate dimensions and precisions exist for data attributes.

It specifies the data needed for a project but, depending on the project's complexity, interacts with other logical data models.

A picture containing text, screenshot, font, line

Description automatically generated

**Physical Model**

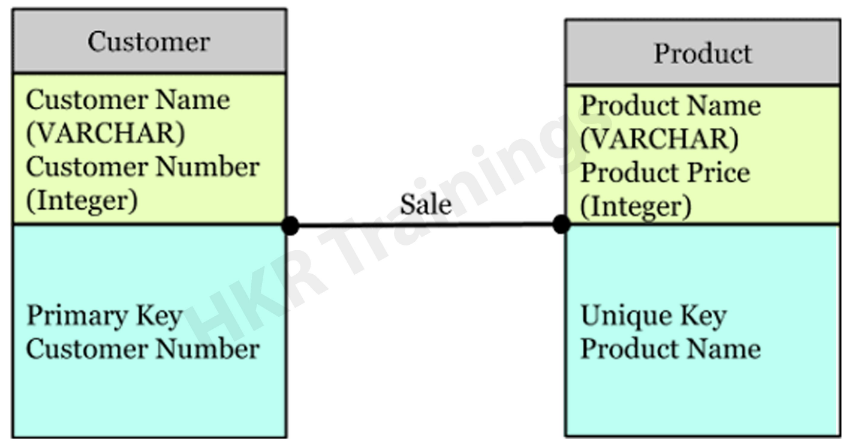
The physical model explains how to use a database management system to execute a data model. It lays out the process in terms of tables, CRUD operations, indexes, partitioning, etc. Database Administrators and Developers build it.

The Physical Data Model specifies how a data model is implemented in a database. It attracts databases and aids in developing schemas by duplicating database constraints, triggers, column keys, other RDBMS functions, and indexes. This data model aids in visualizing the database layout. Views, access profiles, authorizations, primary and foreign keys, and so on are all specified in this model.

The majority and minority relationships are defined in the Data Model by the relationship between tables. It is created for a specific version of a database management system, data storage, and project site.

The Physical Data Model was created for a database management system (DBMS), data storage, and a project site. It contains table relationships that address the nullability and cardinality of the relationships.

Views, access profiles, authorizations, primary and foreign keys, and so on are all specified here.



**Hierarchical Model**

The hierarchical model is used to assemble data into a tree-like structure with a single root that connects all of the data. A single root like this evolves like a branch, connecting nodes to the parent nodes, with each child node having just one parent node. The data is structured in a relational system with a one-to-many relationship between two different data types in this model. For example, in a college, a department consists of a set of courses, professors, and students.

**A picture containing text, font, screenshot, diagram

Description automatically generated**

**Relational Model**

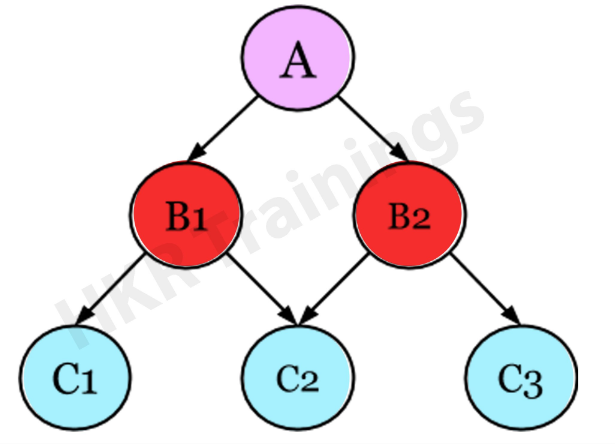
The relational data model is the most widely used data model today. It organizes data into tables with rows and columns, where relationships are established through keys. It provides a powerful and flexible way to represent and manipulate data, and it forms the foundation of relational database management systems (RDBMS).

**A picture containing text, screenshot, diagram, font

Description automatically generated**

**Network Model**

The network data model is similar to the hierarchical model but allows for more complex relationships. It represents data using records and sets, and it allows entities to have multiple relationships. This model was prevalent in the early days of database systems.

****

**Entity–Relationship Model**

The Entity-Relationship Model (ERM) is a diagram that depicts entities and their relationships. The E-R model generates an entity set, attributes, relationship set, and constraints when constructing a real-world scenario database model. The E-R diagram is a graphical representation of this kind.

An entity may be an object, a concept, or a piece of data stored in relation to the data. It has properties called attributes, and a set of values called domain defines each attribute. A relationship is a logical connection between two or more entities. These connections are mapped to entities in several ways.

Consider a College Database, where a Student is an entity, and the Attributes are Student details such as Name, ID, Age, Address, and so on. As a result, there will be a relation between them.

**A diagram of a student

Description automatically generated with medium confidence**

**Facts:** Facts represent the measurable and numerical data points or metrics that provide insights into a specific aspect of a business process or activity. They are typically numeric values, such as sales revenue, quantity sold, or customer satisfaction score. Facts are the core information that businesses analyse to gain insights and make data-driven decisions. Facts are associated with a specific event, transaction, or measurement and are usually stored in the fact table in a data warehouse.

**Fact tables and its different types:**

**Measure types**

A fact table can store different types of measures such as additive, non-additive, semi-additive.

**Additive** – As its name implied, additive measures are measures that can be added to all dimensions.

**Non-additive** – different from additive measures, non-additive measures are measures that cannot be added to all dimensions.

**Semi-additive** – semi-additive measures are the measure that can be added to only some dimensions.

**Types of fact tables.**

All fact tables are categorized by the three most basic measurement events:

**Transactional** – Transactional fact table is the most basic one that each grain associated with it indicated as “one row per line in a transaction”, e.g., every line item appears on an invoice. Transaction fact table stores data of the most detailed level, therefore, it has a high number of dimensions associated with it.

**Periodic snapshots** – Periodic snapshots fact table stores the data that is a snapshot in a period of time. The source data of the periodic snapshots fact table is data from a transaction fact table where you choose a period to get the output.

**Accumulating snapshots** – The accumulating snapshots fact table describes the activity of a business process that has a clear beginning and end. This type of fact table, therefore, has multiple date columns to represent milestones in the process. A good example of accumulating snapshots fact table is the processing of a material. As steps towards handling the material are finished, the corresponding record in the accumulating snapshots fact table gets updated.

**Dimensions:** Dimensions provide the context and descriptive attributes related to the facts. They describe the characteristics or qualities associated with the facts and provide additional details for analysis and filtering. Dimensions can include various attributes such as time, location, product, customer, or any other relevant information that helps to understand the facts from different perspectives. Dimensions are typically stored in dimension tables and are linked to the fact table through relationships.

**Example:**

To illustrate with an example: In a sales analysis scenario, the sales revenue (fact) can be associated with dimensions like date, product, and customer. The date dimension can include attributes such as year, month, and day. The product dimension may have attributes like product name, category, and price. The customer dimension can include attributes such as customer ID, name, and geographic location.

By organizing data into facts and dimensions, a data model provides a structure that enables powerful analytical capabilities. Analysing facts in the context of different dimensions allows businesses to slice and dice data, perform aggregations, and gain insights into performance, trends, and patterns.

**Here are some common dimension types:**

**Conformed Dimensions:** Conformed dimensions are dimensions that have the same meaning and structure across multiple data sources or business processes. They provide consistency and allow for integration and comparison of data from different systems. Conformed dimensions are essential for building enterprise-wide data warehouses and ensuring data accuracy and consistency.

**Slowly Changing Dimensions (SCDs):** Slowly changing dimensions are dimensions that capture historical changes in attribute values over time. They are particularly useful when tracking changes in descriptive attributes, such as customer addresses or product descriptions. SCDs are categorized into different types based on how they handle changes, including Type 1 (overwrite existing values), Type 2 (maintain history with new records), and Type 3 (maintain a limited history within the same record).

**Junk Dimensions:** Junk dimensions are used to handle low-cardinality attributes that do not warrant a separate dimension table. They are created by combining several low-cardinality attributes into a single dimension table, reducing the overall number of dimension tables in the model. Junk dimensions are especially useful for simplifying complex or diverse attributes, such as flags, indicators, or categorical attributes.

**Role-Playing Dimensions:** Role-playing dimensions are dimensions that play multiple roles within the same fact table. This typically occurs when the fact table captures different perspectives or interpretations of a dimension. For example, a date dimension could be used to represent both the order date and the ship date in a sales fact table. Role-playing dimensions allow for more flexibility and efficiency in modeling without duplicating dimension tables.

**Degenerate Dimensions:** Degenerate dimensions are attributes that are embedded directly within the fact table instead of having a separate dimension table. These attributes have no hierarchies or descriptive details and are typically used as identifiers or transactional codes. Examples include invoice numbers, order numbers, or transaction IDs. By incorporating degenerate dimensions into the fact table, it eliminates the need for an additional dimension table.

**Multivalued Dimensions:** Multivalued dimensions are dimensions that can have multiple values associated with a single fact record. They are used to represent attributes with multiple occurrences or values, such as product categories, tags, or customer preferences. Multivalued dimensions are implemented using bridge tables that establish the many-to-many relationship between the dimension and fact tables.

**Outrigger Dimensions:** Outrigger dimensions are additional dimension tables that are connected to the primary dimension table to provide supplementary attributes. They are used when certain attributes pertain to only a subset of dimension records. For example, in a product dimension, an outrigger dimension may be created to store warranty information for a specific group of products.

**Dimensional Modelling-Related Keys:**

**Primary Key (PK):** A primary key is a unique identifier within a table that uniquely identifies each record. It ensures that each row in the table is unique and serves as a reference for establishing relationships with other tables.

**Foreign Key (FK):** A foreign key is a field or set of fields in one table that refers to the primary key in another table. It establishes a relationship between the tables, allowing for data integrity and the ability to join tables for analysis.

**Surrogate Key:** A surrogate key is a unique identifier assigned to a dimension table to provide a stable reference for relationships. It is typically an artificially generated key, such as an auto-incrementing integer or a globally unique identifier (GUID). Surrogate keys are used to avoid complexities that may arise from using natural keys, which can change over time.

**Business Key:** A business key is a natural key that uniquely identifies a dimension entity based on its inherent characteristics. It represents a meaningful attribute of the dimension, such as a customer ID, product code, or employee ID. Business keys are used for lookups and in data integration processes.

**Composite Key:** A composite key is a key that consists of multiple attributes or fields that, together, uniquely identify a record in a table. It is used when a single attribute is not sufficient to establish uniqueness. Composite keys are commonly used in bridge tables or junction tables in many-to-many relationships.

**Degenerate Dimension Key:** A degenerate dimension key is a key that is derived from a transactional fact record and represents a dimension attribute that does not have a separate dimension table. Examples include invoice numbers, order numbers, or transaction IDs that exist only within the fact table.

**Natural Key:** In contrast to surrogate keys, which are artificially generated and have no inherent meaning, natural keys are derived from the data domain and have a direct relationship to the real-world entity being represented.

**Benefits of Data Modelling:**

**Clarity and Structure:** Data modelling provides a clear structure for organizing data, making it easier to understand, manage, and communicate within an organization. It establishes relationships, defines attributes, and establishes rules for data integrity.

**Data Consistency:** By defining data models, organizations can ensure consistency and standardization in data representation and storage. This consistency improves data quality and reduces data anomalies or discrepancies.

**Efficient Data Integration:** Data modelling facilitates data integration by defining relationships between different data entities. It allows for the consolidation and aggregation of data from various sources, making it easier to access and analyse**.**

**Enhanced Data Analysis and Reporting:** Well-designed data models enable effective data analysis and reporting. They provide a foundation for creating queries, generating insights, and producing meaningful reports that support decision-making processes.

**Improved Data Governance and Security:** Data models help in implementing data governance practices by establishing rules and standards for data management, access controls, and security. They enable organizations to define roles, responsibilities, and permissions for data handling.

**Drawbacks of Data Modelling:**

**Time and Effort:** Developing a comprehensive data model can be a time-consuming and resource-intensive process. It requires gathering requirements, analyzing data relationships, and collaborating with stakeholders to create an accurate representation of the data.

**Flexibility Challenges:** Data models can be rigid and less adaptable to changes in business requirements or evolving data needs. Modifying existing data models can be complex, especially if there are dependencies and impact on downstream systems.

**Data Complexity:** In complex systems with numerous data entities and relationships, creating and managing data models can become challenging. Maintaining large and intricate data models may require dedicated resources and expertise.

**STAR SCHEMA:**

A star schema is a type of data modelling technique used in data warehousing to represent data in a structured and intuitive way. In a star schema, data is organized into a central fact table that contains the measures of interest, surrounded by dimension tables that describe the attributes of the measures.

In a star schema, each dimension table is joined to the fact table through a foreign key relationship. This allows users to query the data in the fact table using attributes from the dimension tables.

The star schema is a popular data modelling technique in data warehousing because it is easy to understand and query. The simple structure of the star schema allows for fast query response times and efficient use of database resources. Additionally, the star schema can be easily extended by adding new dimension tables or measures to the fact table, making it a scalable and flexible solution for data warehousing.

It is said to be star as its physical model resembles to the star shape having a fact table at its centre and the dimension tables at its peripheral representing the star’s points. Below is an example to demonstrate the Star Schema:

A picture containing text, diagram, screenshot, plan

Description automatically generated

In the above demonstration, SALES is a fact table having attributes i.e. (Product ID, Order ID, Customer ID, Employer ID, Total, Quantity, Discount) which references to the dimension tables. Employee dimension table contains the attributes: Emp ID, Emp Name, Title, Department and Region. Product dimension table contains the attributes: Product ID, Product Name, Product Category, Unit Price. Customer dimension table contains the attributes: Customer ID, Customer Name, Address, City, Zip. Time dimension table contains the attributes: Order ID, Order Date, Year, Quarter, Month.

**Model of Star Schema:**

In Star Schema, Business process data, that holds the quantitative data about a business is distributed in fact tables, and dimensions which are descriptive characteristics related to fact data. Sales price, sale quantity, distant, speed, weight, and weight measurements are few examples of fact data in star schema.

**Features:**

**Central fact table**: The star schema revolves around a central fact table that contains the numerical data being analysed. This table contains foreign keys to link to dimension tables.

**Dimension tables**: Dimension tables are tables that contain descriptive attributes about the data being analysed. These attributes provide context to the numerical data in the fact table. Each dimension table is linked to the fact table through a foreign key.

**Denormalized structure**: A star schema is denormalized, which means that redundancy is allowed in the schema design to improve query performance. This is because it is easier and faster to join a small number of tables than a large number of tables.

**Simple queries**: Star schema is designed to make queries simple and fast. Queries can be written in a straightforward manner by joining the fact table with the appropriate dimension tables.

**Aggregated data**: The numerical data in the fact table is usually aggregated at different levels of granularity, such as daily, weekly, or monthly. This allows for analysis at different levels of detail.

**Fast performance**: Star schema is designed for fast query performance. This is because the schema is denormalized and data is pre-aggregated, making queries faster and more efficient.

**Easy to understand**: The star schema is easy to understand and interpret, even for non-technical users. This is because the schema is designed to provide context to the numerical data through the use of dimension tables.

**SNOWFLAKE SCHEMA:**

A snowflake schema is a type of data modelling technique used in data warehousing to represent data in a structured way that is optimized for querying large amounts of data efficiently. In a snowflake schema, the dimension tables are normalized into multiple related tables, creating a hierarchical or “snowflake” structure.

In a snowflake schema, the fact table is still located at the centre of the schema, surrounded by the dimension tables. However, each dimension table is further broken down into multiple related tables, creating a hierarchical structure that resembles a snowflake.

The advantage of a snowflake schema is that it can help to reduce data redundancy and improve data integrity. By normalizing the dimension tables into multiple related tables, redundant data can be eliminated, and data integrity can be improved through the use of foreign key relationships.

However, the snowflake schema can also be more complex to query than a star schema because it requires more table joins. This can result in slower query response times and higher resource usage in the database. Additionally, the snowflake schema can be more difficult to understand and maintain because of the increased complexity of the schema design.

**What is snowflaking?**

The snowflake design is the result of further expansion and normalized of the dimension table. In other words, a dimension table is said to be snowflaked if the low-cardinality attribute (columns with less possibilities values, like Gender = Male and Female only ) of the dimensions has been divided into separate normalized tables.

**Features of the snowflake schema include:**

**Normalization**: The snowflake schema is a normalized design, which means that data is organized into multiple related tables. This reduces data redundancy and improves data consistency.

**Hierarchical Structure**: The snowflake schema has a hierarchical structure that is organized around a central fact table. The fact table contains the measures or metrics of interest, and the dimension tables contain the attributes that provide context to the measures.

**Multiple Levels**: The snowflake schema can have multiple levels of dimension tables, each related to the central fact table. This allows for more granular analysis of data and enables users to drill down into specific subsets of data.

**Joins**: The snowflake schema typically requires more complex SQL queries that involve multiple tables joins. This can impact performance, especially when dealing with large data sets.

**Scalability**: The snowflake schema is scalable and can handle large volumes of data. However, the complexity of the schema can make it difficult to manage and maintain.

**GALAXY SCHEMA:**

A Galaxy Schema contains two fact table that share dimension tables between them. It is also called Fact Constellation Schema. The schema is viewed as a collection of stars hence the name Galaxy Schema.

A picture containing text, screenshot, font, number

Description automatically generated

As you can see in above example, there are two facts table.

Revenue and Product.

In Galaxy schema shares dimensions are called Conformed Dimensions.

**Characteristics of Galaxy Schema**

The dimensions in this schema are separated into separate dimensions based on the various levels of hierarchy.

For example, if geography has four levels of hierarchy like region, country, state, and city then Galaxy schema should have four dimensions.

Moreover, it is possible to build this type of schema by splitting the one-star schema into more Star schemes.

The dimensions are large in this schema which is needed to build based on the levels of hierarchy.

This schema is helpful for aggregating fact tables for better understanding.

**DIMENSIONAL DATA MODELLING**

Dimensional data modelling is a technique used in data warehousing to organize and structure data in a way that makes it easy to analyse and understand. In a dimensional data model, data is organized into dimensions and facts.

In a dimensional data model, the fact table is the central table that contains the measures or metrics of interest, surrounded by the dimension tables that describe the attributes of the measures. The dimension tables are related to the fact table through foreign key relationships.

The advantage of dimensional data modelling is that it provides a simple, intuitive, and flexible way to organize data for analysis. The dimensional model is easy to understand and query, allowing users to quickly access and analyse the data they need. Additionally, the dimensional model can be easily extended to add new dimensions or facts as the business needs change.

Main goal of this modelling is to improve the data retrieval, so it is optimized for SELECT OPERATION. The advantage of using this model is that we can store data in such a way that it is easier to store and retrieve the data once stored in a data warehouse.

**Steps for DDM:**

A picture containing text, coil spring

Description automatically generated

**1) Identify the Business Process**

A Business Process is a very important aspect when dealing with Dimensional Data Modelling. The business process helps to identify what sort of Dimension and Facts are needed and maintain the quality of data.

**2) Identify Grain**

Identification of Grain is the process of identifying how much normalisation (lowest level of information) can be achieved within the data. It is the stage to decide the incoming frequency of data (i.e., Daily, weekly, monthly, yearly), how much data we want to store in the database (one day, one month, one year, ten years), and how much the storage will cost.

**3) Identify the Dimensions**

Dimensions are the key components in the Dimensional Data Modelling process. It contains detailed information about the objects like date, store, name, address, contacts, etc. For example, in an E-Commerce use case, a Dimension can be:

Product, Order Details, Order Items, Departments, Customers (etc).

**4) Identify the Facts**

Once the Dimensions are created, the measures/transactions are supposed to be linked with the associated Dimensions. The Fact Tables hold measures and are linked to Dimensions via foreign keys. Usually, Facts contain fewer columns and huge rows. For example, in an E-Commerce use case, one of the Fact Tables can be of orders, which holds the products’ daily ordered quantity. Facts may contain more than one foreign key to build relationships with different Dimensions.

**5) Build the Schema**

The next step is to tie Dimensions and Facts into the Schema. Schemas are the table structure, and they align the tables within the database. There are 2 types of Schemas:

Star Schema: The Star Schema is the Schema with the simplest structure. In a Star Schema, the Fact Table surrounds a series of Dimensions Tables. Each Dimension represents one Dimension Table. These Dimension Tables are not fully normalised. In this Schema, the Dimension Tables will contain a set of attributes that describes the Dimension. They also contain foreign keys that are joined with the Fact Table to obtain results.

Snowflake Schema: A Snowflake Schema is the extension of a Star Schema and includes more Dimensions. Unlike a Star Schema, the Dimensions are fully normalised and are split down into further tables. This Schema uses less disk space because they are already normalised. It is easy to add Dimensions to this Schema and the data redundancy is also less because of the intricate Schema design.

**Difference between OLTP and OLAP:**

OLTP (Online Transaction Processing) systems manage transaction-oriented applications, typically for data entry and retrieval transactions. OLTP databases are highly normalized and store data in multiple related tables. Credit card payment systems, ATM cards, etc., are some examples of OLTP systems.

On the other hand, OLAP (Online Analytical Processing) is used for data analysis and business intelligence. OLAP databases are optimized for read-only operations and store data in multidimensional schemas for fast access and aggregations. Data warehouses are the most common examples of OLAP systems.

**DATA MART**

**What is Data Mart?**

A Data Mart is focused on a single functional area of an organization and contains a subset of data stored in a Data Warehouse. A Data Mart is a condensed version of Data Warehouse and is designed for use by a specific department, unit or set of users in an organization. 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 Datawarehouse.

**Why do we need Data Mart?**

Data Mart helps to enhance user’s response time due to reduction in volume of data.

It provides easy access to frequently requested data.

Data mart are simpler to implement when compared to corporate Datawarehouse. At the same time, the cost of implementing Data Mart is certainly lower compared with implementing a full data warehouse.

Compared to Data Warehouse, a datamart is agile. In case of change in model, datamart can be built quicker due to a smaller size.

**Types of Data Mart**

There are three main types of data mart:

**Dependent:** Dependent data marts are created by drawing data directly from operational, external or both sources.

**Independent:** Independent data mart is created without the use of a central data warehouse.

**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 is one of the data mart examples which 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 Mart in data warehouse 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.

A diagram of a data warehouse

Description automatically generated with low confidence

**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.

A picture containing cylinder, design, illustration

Description automatically generated

**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 the best data mart example 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.

A picture containing drawing, illustration, clipart, design

Description automatically generated

**Steps in Implementing a Datamart**

Implementing a Data Mart is a rewarding but complex procedure. Here are the detailed steps to implement a Data Mart:

**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 Data Mart design.

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.

**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-users 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.

**Advantages and Disadvantages of a Data Mart**

**Advantages**

Data marts contain a subset of organization-wide data. This Data is valuable to a specific group of people in an organization.

It is cost-effective alternatives to a data warehouse, which can take high costs to build.

Data Mart allows faster access of Data.

Data Mart is easy to use as it is specifically designed for the needs of its users. Thus, a data mart can accelerate business processes.

Data Marts needs less implementation time compared to Data Warehouse systems. It is faster to implement Data Mart as you only need to concentrate the only subset of the data.

It contains historical data which enables the analyst to determine data trends.

**Disadvantages**

Many a times enterprises create too many disparate and unrelated data marts without much benefit. It can become a big hurdle to maintain.

Data Mart cannot provide company-wide data analysis as their data set is limited.

Financial reporting as example of data model:

**Fact Table: FinancialTransaction**

AccountID (Foreign Key referencing the Account dimension)

TimeID (Foreign Key referencing the Time dimension)

OrganizationID (Foreign Key referencing the Organization dimension)

Amount

**Dimension Tables:**

**Account**

AccountID (Primary Key)

AccountName

AccountType

AccountCategory

Currency

**Time**

TimeID (Primary Key)

Date

Year

Quarter

Month

Day

FiscalYear

**Organization**

OrganizationID (Primary Key)

OrganizationName

Region

Department