**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**

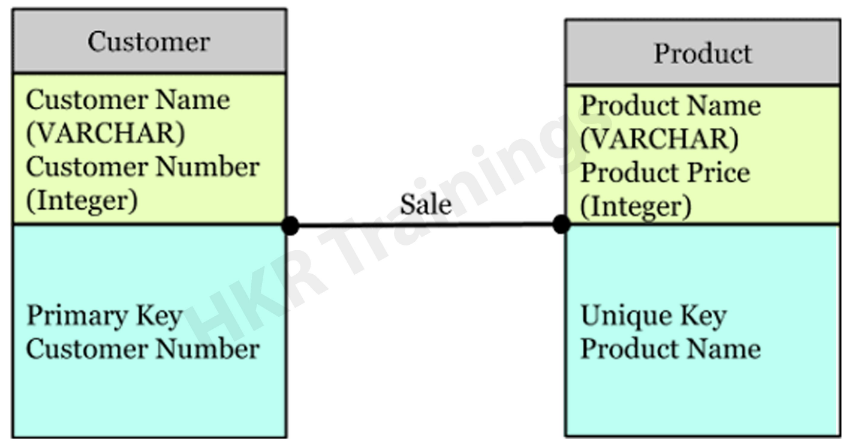
The physical model explains how to use a database management system to execute a data model. It lays out the process in terms of tables, CRUD operations, indexes, partitioning, etc. Database Administrators and Developers build it.

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**

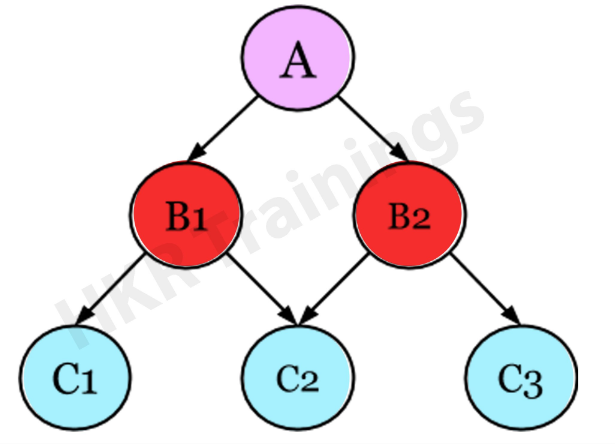
The relational data model is the most widely used data model today. It organizes data into tables with rows and columns, where relationships are established through keys. It provides a powerful and flexible way to represent and manipulate data, and it forms the foundation of relational database management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

**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.

**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.