>> Data Engineering notes by Suraj <<

Table of contents

00. Relational database management system(RDBMS)

A relational database management system (RDBMS) is a software system that allows us to create, update, and manage relational databases. A relational database is a database that organizes data into one or more tables, where each table consists of rows and columns.

E.g -

Let's say we have a business that sells products to customers. We want to keep track of our products, customers and orders. To do this, create three tables: Products, Customers and Orders.

A product table can have columns for product ID, product name, price, and in-stock quantity. A customer table can have columns for customer ID, first name, last name, email address, and phone number. The Orders table can have columns OrderID, CustomerID, ProductID, OrderDate, and OrderAmount. We can create relationships between these tables using foreign keys. For example, the Customer ID column in the Orders table would be a foreign key that references the Customer ID column in the Customers table. This allows us to associate each order with the customer who placed the order.

With an RDBMS, We can perform various operations on a database, such as inserting new data, updating existing data, and performing data queries to retrieve information. For example, We can use SQL (Structured Query Language) to write a query to retrieve all orders placed by a particular customer or all products priced above a certain amount.

RDBMS also provides means to ensure data integrity and consistency. For example, you can define constraints on a table to enforce rules, such as not allowing duplicate data or requiring certain fields to be filled. You can also use transactions to ensure that a set of operations is performed as a unit, so

that if part of the transaction fails, the entire transaction is rolled back. Overall, RDBMSs provide a powerful and flexible way to manage and work with relational databases.

01 Database and its types

A database is a collection of data organized for efficient storage, retrieval, and management. There are many different types of databases, each with their own unique strengths and weaknesses.

common database and its types

1. Relational database - This is the most common type of database where data is stored in tables with rows and columns. Each row represents a record, and each column represents a specific attribute of that record. Relational databases are suitable for structured data with clear relationships between tables.

Examples of relational databases include MySQL, Oracle, and Microsoft SQL Server. Example: A library system with tables for books, authors, and borrowers. The books table will have columns for the book title, author ID, and ISBN, while the authors table will have columns for the author name and ID. The borrower's desk is the borrower IID, the name and the contact information. Table links by author -id and borrower -id.

2. NoSQL databases are non-relational databases that store data in a way that does not require a fixed schema or predefined relationships between tables. NoSQL databases are suitable for unstructured data with complex relationships. Examples of NoSQL databases are MongoDB and Cassandra. Example: A social media platform that stores user data in a NoSQL database. Each user profile will be saved as a document with embedded fields for user preferences, contacts and messages.

3. Object-oriented databases - These are databases that store objects instead of tables, making them ideal candidates for object-oriented programming languages. Object-oriented databases are ideal for complex data with many relationships between objects. Examples of object-oriented databases are ObjectDB and Versant. Example: video games with players, objects and enemies. Each player has healthy, horizontal and storage properties. Each element object has attributes such as name, type, and value. These objects will be related through relationships such as ownership and interaction.

4. Graph Databases - These databases store data as nodes and edges, allowing for efficient use of complex relationships between data. Graph databases are suitable for data containing many interconnected entities, such as social networks or recommendation engines. Examples of graph databases are Neo4j and OrientDB.

Example. The recommendation engine for the music streaming service uses a graph database to maintain relationships between artists, albums and songs. Each node represents an artist, album, or song, and each edge represents a relationship, such as "artist" or "similar."

02 Data modelling

Data modeling is the process of creating a visual representation of the structure of a database. It helps organize data in a logical way and ensures that it is accurate, consistent and easily accessible. A common approach to data modeling is to use Entity Relationship (ER) diagrams. An ER diagram is a graphical representation of entities (things) in a system and their interrelationships. Let's take an example to illustrate how data modeling works:

Let's say you're building a database for a university. You can start by

identifying the entities involved, such as "student", "course", and "professor". You will then identify attributes (features) for each entity, such as "Student ID", "Course Name", and "Professor Name".

Then define the relationships between the entities.

For example, a student may be enrolled in multiple courses, and a course may have multiple students. This is a many-to-many relationship and requires a "connection table" to connect the two entities. Also, one professor can teach several courses, and one course can be taught by several professors. This is another many-to-many relationship that requires a join table. Once the entities and relationships are identified, an ER diagram can be created to visually represent the structure of the database. ER diagrams show entities as boxes with attributes specified in each box. These relationships are shown as lines connecting boxes, with symbols indicating the type of relationship (for example, crow's feet for many-to-many relationships).

03 Data warehousing

Data warehousing is the process of collecting, storing and managing data from various sources in a centralized repository for data analysis and decision making. A repository is called a data warehouse, and it typically includes historical data as well as data from various departments and systems within an organization.

Let's say we have an online store and we have a database that stores information about customer orders, products, shipping and payment information. We also have data from our social media campaigns, customer surveys and website analytics. To make sense of all this data, we can create a data warehouse that collects all the information in one place.

A data warehouse will use a process called extract, transform, load (ETL) to

extract data from various sources, transform the data into an easy-to-analyze format, and then load it into the data warehouse. Once the data is in the data warehouse, we can use business intelligence tools and analytics software to gain insights and make data-driven decisions.

For example, we can analyze sales data to determine which products sell the most, determine which marketing campaigns drive the most traffic to our website, or identify customer behavior patterns that can help improve customer satisfaction and retention. Overall, a data warehouse is a powerful tool for organizations that need to make sense of large amounts of data from multiple sources. By creating a centralized repository of data that is easy to access and analyze, organizations can gain valuable insights and make smarter decisions.

04 ETL processes

ETL stands for Extract, Transform and Load. It is the process used in a data warehouse to obtain data from various sources, transform the data to meet the specific requirements of the target data model, and load it into the data warehouse.

To visualize the ETL process, let's take the example of a retail company that wants to analyze its sales data. Sales data can be scattered across different databases, spreadsheets, and other data sources. Businesses must extract, transform, and load this data into a data warehouse for meaningful analysis.

The ETL process for this retail business is as follows:

1. Extraction: Data is extracted from various sources such as point-of-sale systems, customer databases and spreadsheets.

2. Transformation: Data is transformed to make it suitable for analysis. For example, data may need to be cleaned, filtered and converted into a consistent format. Businesses may also need to perform calculations, combine data from multiple sources, and create derived fields.

3. Load: The transformed data is loaded into the data warehouse. This step involves mapping the transformed data into the appropriate tables and columns in the data warehouse. The retail store's sales data can then be analyzed using a data warehouse. The ETL process is a key part of the data warehousing process because it ensures that the data in the warehouse is accurate, consistent, and up-to-date.

05 Data Integration

Data integration is the process of combining data from different sources into one unified view. It involves gathering data from different formats, systems and locations into one place where it can be analyzed and used to make informed decisions. Think of it as a puzzle. Each fragment represents different data from a different source. Taken separately, these fragments mean little, but put together they form a complete picture. Similarly, data integration combines data from different sources to create a complete, unified view that can be used to gain insights and make informed decisions.

For example, a business may have customer data in multiple databases, spreadsheets, and other formats. By integrating this data, they can gain a comprehensive understanding of customer behavior, preferences and purchase history. This information can then be used to improve marketing efforts, customer service and other business processes.

06. Data mining

Data mining is the process of discovering patterns, trends and insights from

large data sets. This involves using a variety of statistical and machine learning techniques to identify meaningful relationships and patterns in the data.

For example, a retail company can use data mining to analyze customer buying patterns and determine which products are most popular at any given time of year. This information can help companies optimize their inventory and marketing strategies, thereby increasing sales and profitability.

07.. Data Analysis methods

Data analysis is the process of examining and interpreting data to gain insights and make informed decisions. Depending on the type of data and the problem to be solved, different data analysis methods can be used.

Common data analysis methods are :

1. Descriptive Analysis: Descriptive analysis is used to summarize and describe the main features of a dataset, such as its mean, median, mode, range, and standard deviation. This method is useful for getting an overview of the dataset and understanding its general characteristics.

2. Inferential analysis - It typically involves hypothesis testing and estimation using statistical techniques.

3. Regression Analysis - It is used to analyze the relationship between two or more variables.

4. Time series analysis - It is used to analyze data that is collected over time. It involves identifying patterns and trends in the data.

5. Cluster analysis : It is used to group similar objects together based on

their characteristics. It is used to identify patterns in data that are not immediately apparent. Use cases include customer segmentation, fraud detection and others.

6. Factor analysis - It is used to identify underlying factors or dimensions that explain the variation in a dataset. It is often used to reduce the number of variables in a dataset and identify the most imp. factors driving the variation.

08 Big Data

Big data refers to extremely large and complex data sets that cannot be processed or analyzed using traditional methods or software. These datasets often come from a variety of sources, such as social media, e-commerce transactions, scientific research, and sensor data.

Big data is often characterized by its volume, speed and variety. The amount of data is so large that it requires specialized tools and techniques to store and manage it.

Velocity refers to the rate at which data is generated and processed, fast enough to require real-time or near-real-time analysis. Data diversity refers to the many different sources and formats of data, which can include structured, semi-structured and unstructured data.

Given the volume and complexity of big data, the biggest challenge of big data is extracting valuable insights and knowledge from it.

09 Distributed computing

Distributed computing refers to the use of multiple computers working together to solve computational problems or perform tasks. Instead of relying

on a single computer to perform all tasks, distributed computing allows you to spread the workload across multiple computers. It allows large data sets and faster calculations and improves failure tolerance and scalability.

The basic idea of distributed computing is that the computer involved as a team collaboration, each computer promotes its treatment functions on the current task. This can be achieved using different technologies such as clusters, grid computing or cloud computing.

10 cloud computing

Cloud computing is a technology that allows you to access computing resources such as servers, storage, and software applications via the Internet instead of relying on our local hardware.

Traditionally, if we needed to run programs or store data, we had to buy and maintain your own servers and storage. But with cloud computing, we can rent a virtual computer or storage space, so we don't have to worry about physical hardware or maintenance.

11 Stream processing

Stream processing is a method of processing continuous streams of data in real time or near real time as they are generated. This involves processing and analyzing data while it is still in motion, rather than waiting for the data to be stored in a database or data warehouse. Data streams can come from a variety of sources, such as sensors, social media, or IoT devices, and contain large amounts of data that need to be processed quickly to gain insights, detect anomalies, or trigger alerts.

Stream processing technology makes it possible to process these high-speed data streams in real-time by breaking the data into smaller chunks and

processing each chunk as it is received. It enables businesses to make decisions and act in real-time based on insights gained from data streams.

For example, stream processing can be used to monitor stock market data in real time, identify trading patterns, and send alerts to traders when certain conditions are met. It can also be used to analyze social media feeds to identify trends and sentiment around a particular product or event, or process IoT sensor data to identify equipment failures before they occur.

12 Batch processing

Batch processing refers to batch processing of large amounts of data simultaneously. It involves collecting data from various sources, organizing it, processing it and generating output. Batch processing is typically used for tasks that do not need to be processed immediately or in real time. It is usually used for tasks that are performed regularly, such as processing large amounts of data every night or every week.

13 Parallel Processing

Parallel processing refers to the execution of multiple computer tasks at the same time instead of one after the other. The goal of parallel processing is to reduce the time it takes to complete a task by breaking it into smaller chunks and assigning each chunk to a different processor, core, or thread to process at the same time.

The idea of parallel processing is similar to teamwork, where a group of people work together on a project to complete it faster than if each person tackled it individually. Parallel processing also allows multiple processors to work together to complete tasks faster than a single processor.

14 Map reduce

MapReduce is a programming paradigm for processing large amounts of data in a distributed and parallel manner. It consists of two main steps: Map and Reduce.

The map step takes input data and converts it into a set of key-value pairs. It applies a function to each element of the input data and outputs a set of intermediate key-value pairs. A key is usually a subset of the input data, and a value is the piece of data associated with that key.

The mapping step can be performed in parallel on different parts of the input data. Step Reducer takes the output from the map step and sums it by key. It applies a function to all values associated with each key and produces a single output value for that key. Reduction operations can also be performed in parallel on different subsets of intermediate data.

In short, MapReduce is like a large-scale version of the divide-and-conquer strategy, where the input data is divided into smaller parts, processed independently and in parallel, and then combined to produce a final result. This makes it ideal for processing large data sets in a scalable and efficient manner.

15 Hadoop is an open source framework for distributed storage and processing of large data sets on clusters of commodity hardware. It provides a cost-effective and scalable solution for big data processing. Hadoop consists of two main components: the Hadoop Distributed File System (HDFS) and MapReduce.

HDFS is a distributed file system that can store data across multiple nodes in a cluster. It breaks large data sets into smaller chunks and distributes them across cluster nodes for storage. This approach makes data processing faster and more efficient, as data can be accessed in parallel on multiple nodes. MapReduce is a programming model for parallel processing of large data sets on Hadoop clusters. It consists of two phases: Map and Reduce. The map

phase takes input data, processes it, and generates intermediate key-value pairs. The reducer stage takes the output from the map stage and combines the intermediate key-value pairs into the final output. Here is an example of how Hadoop MapReduce works:

Let's say we have a large data set that contains sales transactions in a retail store. We want to calculate the total revenue that the store generates from each product category. For this we can use Hadoop MapReduce.

In the map phase, we divide the data set into smaller chunks and distribute them to different nodes of the Hadoop cluster. Each node processes its own blocks of data and generates intermediate key-value pairs. For example, a node could process all transactions for the electronics category and output a key-value pair where the key is "electronics" and the value is the revenue generated by the electronics category. In the reducer phase, we merge the intermediate key-value pairs generated in the map phase to get the final result. For example, we could have a node that takes all intermediate key-value pairs for each product category, sums the revenue for each category, and produces a final output of the total revenue for each product category. In summary, Hadoop is a powerful framework for distributed storage and processing of large data sets. It consists of two main components, HDFS and MapReduce, which work together to provide a cost-effective and scalable solution for processing big data. Hadoop MapReduce breaks large data sets into smaller chunks and processes them in parallel across a cluster of nodes, making data processing faster and more efficient.

16 Spark

Apache Spark is an open source big data processing engine designed to enable fast and distributed processing of large data sets across clusters of computers. It provides an interface for cluster programming with implicit data

parallelism and fault tolerance. Here are some key features and capabilities of Spark:

1. In-Memory Processing: It processes data in-memory, making it much faster than other big data processing engines.

2. Distributed computing: It processes data in a distributed manner, which allows it to scale horizontally as more nodes are added to the cluster. Fault tolerance: Spark provides fault tolerance, which means it can handle the failure of a cluster node without losing data.

3. Compute APIs: It provides APIs for several programming languages such as Python, Java, Scala, and R. This makes it easier for developers to work with data in their preferred language.

4. SQL and DataFrames: It provides an SQL interface for working with structured data, as well as DataFrames, which are a higher-level abstraction for distributed datasets. Machine Learning: Spark provides machine learning libraries such as MLlib that allow developers to build and train machine learning models at scale.

17- Data Lakes

A data lake is a large centralized repository that allows you to store all structured and unstructured data at any scale. The concept of a data lake is similar to that of a real lake. Water enters the lake from various sources and users can draw water as needed. Similarly, in a data lake, data comes from various sources such as IoT devices, websites, applications, social media and many more. This data is stored in a raw format, and you can use different types of analytics to gain insights.

Data lakes offer several advantages such as flexibility, scalability and cost-

Since the data is stored in its original format, you don't need to spend time and resources to convert the data to a specific format. In addition, data lakes can handle large amounts of data, and you can easily scale them up or down as needed.

Using a data lake, you can also perform different types of analysis, such as batch processing, real-time processing, machine learning, and artificial intelligence. For example, you can use batch processing to process large amounts of historical data, real-time processing to analyze generated data, machine learning to build predictive models, and AI to perform advanced analytics.

18 Data pipelines

A data pipeline is a series of operations that move data from one place to another, often involving transformation and processing. The goal of a data pipeline is to take raw data and turn it into valuable insights for business decision making.

1. Batch pipeline: It involves processing large amount of data simultaneously. It involves taking large data sets, processing them all at once and outputting the results. Batch processing is often used for data that does not need to be processed in real time, such as financial statements or historical analysis.

2. Streaming pipelines: They are used to process incoming data in real-time. Data streams can be processed instantly and results read in real time. Streaming pipelines are often used in applications such as fraud detection, real-time analytics, and social media monitoring. 3. ETL (Extract, Transform, Load) pipeline: The ETL pipeline involves extracting data from one or more sources, transforming it into an appropriate format or structure, and loading it into the target system. This type of pipeline is often used to move data between systems, clean data, and integrate data from multiple sources.

4. Machine Learning Pipeline: It involves taking raw data, preparing it for machine learning, training a model on the data, and then deploying the model to make predictions. Machine learning pipelines are commonly used in applications such as recommender systems, image recognition, and natural language processing.

5. DevOps Pipeline: It involves automating the process of building, testing and deploying software. This type of pipeline is often used in software development projects that aim to quickly move code changes from a development environment to a production environment.

19 Function as a service

Functions as a Service (FaaS) is a cloud computing model that allows developers to build, run, and manage applications without building and maintaining the underlying infrastructure. In a FaaS cloud provider, the infrastructure is managed, and developers focus only on writing the application logic in the form of functions. Service providers then perform these functions in response to events or requests.

20 Event driven architecture

Event-driven architecture (EDA) is an architectural pattern that facilitates the generation, detection, consumption, and response to events. An event can be defined as any event that occurs at a given time and can be represented by a piece of data.

In EDA, various software components are designed to respond to events that occur in real time. These events can come from a variety of sources, such as users, sensors, or other systems, and can trigger actions that cause changes in system state. EDA events are the backbone of the architecture. They are used to trigger actions, update data, and notify other system components. Events can be processed in real time or queued for later processing, and can be used to coordinate the flow of data between different system components.

For example, consider an e-commerce website that uses EDA. When a customer places an order on the website, an event is generated that contains information about the order, such as the customer's name, address, and payment information. This event can trigger other actions, such as updating the inventory system to reflect the purchased item, sending a confirmation email to the customer, and updating website analytics to track purchases.

21. CEP

Complex Event Processing (CEP) is a method of processing and analyzing data from multiple sources in real-time to identify patterns and relationships between events and then trigger actions based on those patterns. It involves processing and analyzing data streams and detecting complex patterns in those data streams. Simply put, CEP is like a brain that processes large amounts of data from multiple sources, identifies relevant patterns, and triggers actions based on those patterns.

For example, in a stock trading scenario, CEP can analyze stock prices and other relevant data from multiple sources in real time, detect patterns that indicate a potential stock market crash, and automatically trigger actions such as selling shares to prevent losses. CEP works by processing data in real time as it flows through the system.

Data is first collected from various sources, then filtered and processed to identify patterns and finally results are produced. This can be done by combining event processing and rule-based systems

22 Data Mesh

A data mesh is an architectural approach to the design and management of data systems that emphasizes the decentralization of data ownership and management.

In it, data is considered a product, and individual teams are responsible for the quality, management, and delivery of the data generated. A data grid approach allows teams to create and manage their own data products, rather than a central data team controlling all aspects of data management.

These teams are responsible for ensuring that their data meets certain

quality and governance standards and that it is easily used by other teams in the organization. Datagrid architectures often use APIs to facilitate data sharing between teams. These APIs can be used to publish data to a central directory or give other teams direct access to the data.

Datagrids also emphasizes the use of domain-driven design, which means that teams are organized around specific business areas rather than technical functions. This enables teams to better understand business needs and develop data products that better meet those needs.

23 Data Ops

DataOps is an approach to managing the entire data lifecycle, from data ingestion and processing to analysis and implementation, using Agile and DevOps principles. It involves collaboration between data engineers, data scientists, and business stakeholders to quickly and efficiently develop, test, and deploy data-driven applications.

At its core, DataOps focuses on automating and simplifying the processes involved in building and deploying data-driven applications. This includes automated testing and deployment, continuous integration and delivery (CI/CD), version control and data quality monitoring. By implementing DataOps, organizations can increase the speed and accuracy of data operations, improve data quality, and reduce the time and cost of data-driven projects.

For example, data teams can use DataOps principles to create pipelines for receiving and processing customer data from a variety of sources, such as social media and online purchases. This pipeline can include automated data quality checks, data enrichment and feature development, as well as automated testing and deployment to production.

24 MLOps

MLOps, short for "machine learning operations," is the application of DevOps principles to the machine learning lifecycle. It is a set of best practices, processes, and tools that enable data scientists and machine learning engineers to build, deploy, and monitor machine learning models in a systematic, automated, and scalable manner.

It includes several steps in the machine learning lifecycle, including data preparation, model training, model evaluation, deployment, and monitoring. This involves collaboration between various teams such as data scientists, data engineers, software engineers, DevOps engineers and IT operations to ensure that machine learning models are developed and deployed with high quality, security, reliability and scalability.

Simply put, It helps organizations manage and automate the entire machine learning pipeline, from data entry to model deployment and monitoring, ensuring machine learning models are consistently delivered with high quality and reliability. This enables organizations to reduce time to market for machine learning solutions, improve model performance and accuracy, and ultimately derive more value from data.

25 Workflow scheduling

Workflow scheduling is the process of organizing and scheduling tasks that are part of a larger workflow or process. This involves breaking down the overall process into smaller tasks or steps and determining when and in what order those tasks should be executed.

26 Orchestration frameworks

It is a tool or framework that provides a way to manage the coordination, scheduling, and execution of various processes and services in a distributed system or workflow. These platforms aim to simplify the management of

complex systems by providing a centralized interface for managing various components and services.

Simply put, It helps automate and simplify the management of complex workflows involving multiple services, applications, and data sources. They allow organizations to easily create and manage workflows, automate processes, and monitor the performance of those workflows.

For example, in a data processing workflow, It can be used to schedule the execution of various tasks involved in the workflow, such as data extraction, transformation, and loading into a data warehouse. The platform can also manage dependencies between different tasks so that a task is executed only after its predecessors have completed. In addition, the platform can monitor the status of each task and alert users when a task fails or exceeds a certain time limit.

Some popular orchestration platforms include Apache Airflow, Kubernetes, Apache NiFi, and Jenkins.

27- Workflow automation

It refers to the use of software tools to automate and simplify the process of data processing and analysis. This includes automating repetitive and time-consuming tasks such as data extraction, transformation and loading (ETL), data cleansing, data modeling and analysis. E.g

1. ETL pipelines: These pipelines automate the process of extracting data from various sources, transforming it into a suitable format for the required and loading it into the target database or data warehouse.

2. Data Quality Checks: These checks automate the data quality validation process by identifying errors, inconsistencies and missing values.

3. Data Modeling: It can also be used to automate the process of building predictive models by selecting features, training the model and evaluating its performance.

4. Reports: Automation can also be used to create automated reports based on data analysis. Apache Airflow, Apache Nifi or Luigi are often used to create workflows for automating data processing tasks.

< The end