ADVANCED DBMS ASSIGNMENT

Submitted By,

Ronika Paul

- What are databases?

A database is a collection of information stored within a computer. Professionals use databases to store files on your computer to maintain records like customer data. Databases allow computers to store essential information in an organised and easily searchable way.

- Two Primary Database Types

  - **Single-file**: single-file, or flat-file databases, use simple structures and individual files to represent one piece of data or information.
  - **Multi-file relational**: relational databases are more complex databases that use tables to show the relationship between data.

- Different types of Databases

1)Relational Databases

A relational database is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. In a relational database, each row in the table is a

record with a unique ID called the key. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points. Relational databases have been around since the 1970s. The name comes from the way that data is stored in multiple, related tables. Within the tables, data is stored in rows and columns. The relational database management system (RDBMS) is the program that allows you to create, update, and administer a relational database. Structured Query Language (SQL) is the most common language for reading, creating, updating and deleting data. Relational databases are very reliable. They are compliant with ACID (Atomicity, Consistency, Isolation, Durability), which is a standard set of properties for reliable database transactions. Relational databases work well with structured data. Organizations that have a lot of unstructured or semi-structured data should not be considering a relational database.

Examples: Microsoft SQL Server, Oracle Database, MySQL, PostgreSQL and IBM Db2

2) Object-oriented databases

An object-oriented database is based on object-oriented programming, so data and all of its

attributes, are tied together as an object. Object-oriented databases are managed by object-oriented database management systems (OODBMS). These databases work well with object-oriented programming languages, such as C++ and Java. Like relational databases, object-oriented databases conform to ACID standards. An object-oriented database management system works in concert with an object-oriented programming language to facilitate the storage and retrieval of object-oriented data. With an OOD, data objects are stored with all of their properties in the database. When your program terminates, the objects continue to persist, stored in the OOD. When your program starts up again, it can retrieve an object with the properties from the database. The process of storing and retrieving a complex data object with an OOD is transparent to the user of the database.

Examples: Wakanda, ObjectStore, Caché and ZODB

3)Personal database

A personal database is one that is designed for a single person. You typically find these on a personal computer and has a very simple design, comprising only a few tables. Personal databases are not typically suitable for complex operations, large

amounts of data or business operations. A personal database is used to store frequently used, unique, or customized information. This provides faster access to targeted information for use across all studies and projects. Personal databases are a quick and convenient method of accessing frequently used material information in any project or study.

Examples: Microsoft Access and Filemaker

4) NoSQL databases

NoSQL is a broad category that includes any database that doesn’t use SQL as its primary data access language. These types of databases are also sometimes referred to as non-relational databases. Unlike in relational databases, data in a NoSQL database doesn’t have to conform to a pre-defined schema, so these types of databases are great for organizations seeking to store unstructured or semi-structured data. One advantage of NoSQL databases is that developers can make changes to the database on the fly, without affecting applications that are using the database.

Examples: Apache Cassandra, MongoDB, CouchDB, and CouchBase

5)Hierarchical databases

Hierarchical databases use a parent-child model to store data. Hierarchical database model is most appropriate for use cases in which the main focus of information gathering is based on a concrete hierarchy, such as several individual employees reporting to a single department at a company. If you were to draw a picture of a hierarchical database, it would look like a family tree, with one object on top branching down to multiple objects beneath it. The one-to-many format is rigid, so child records can’t have more than one parent record. Originally developed by IBM in the early 1960s, hierarchical databases are commonly used to support high-performance and high availability applications. The key advantage of a hierarchical database is its ease of use. The one-to-many organization of data makes traversing the database simple and fast, which is ideal for use cases such as website drop-down menus or computer folders in systems like Microsoft Windows OS. Due to the separation of the tables from physical storage structures, information can easily be added or deleted without affecting the entirety of the database.

Examples: IBM Information Management System (IMS), Windows Registry

6)Cloud databases

A cloud database is a database service built and accessed through a cloud platform. It serves many of the same functions as a traditional database with the added flexibility of cloud computing. Users install software on a cloud infrastructure to implement the database. A cloud database refers to any database that's designed to run in the cloud. Like other cloud-based applications, cloud databases offer flexibility and scalability, along with high availability. Cloud databases are also often low-maintenance, since many are offered via a SaaS model.

Examples: Microsoft Azure SQL Database, Amazon Relational Database Service, Oracle Autonomous Database.

7) Columnar databases

A columnar database is a database management system (DBMS) that stores data in columns instead of rows. The purpose of a columnar

database is to efficiently write and read data to and from hard disk storage in order to speed up the time it takes to return a query. Columnar databases store data in a way that greatly improves disk I/O performance. They are particularly helpful for data analytics and data warehousing.

The main benefit of a columnar database is faster performance compared to a row-oriented one. That's because it accesses less memory to output data. Because a columnar database stores data by columns instead of rows, it can store more data in a smaller amount of memory. And because the initial data retrieval is done on a column-by-column basis, only the columns that need to be used are retrieved. This makes it possible for a columnar database to scale efficiently and handle large amounts of data.

Examples: Google BigQuery, Cassandra, HBase, MariaDB, Azure SQL Data Warehouse

8) Document databases

A document database is a type of nonrelational database that is designed to store and query data as JSON-like documents. Document databases make it easier for developers to store and query data in a database by using the same document-model format they use in their application

code. The flexible, semi structured, and hierarchical nature of documents and document databases allows them to evolve with applications' needs. The document model works well with use cases such as catalogs, user profiles, and content management systems where each document is unique and evolves over time. Document databases enable flexible indexing, powerful ad hoc queries, and analytics over collections of documents.

Examples: MongoDB, Amazon DocumentDB, Apache CouchDB

9) Graph databases

Graph databases are purpose-built to store and navigate relationships. Relationships are first-class citizens in graph databases, and most of the value of graph databases is derived from these relationships. Graph databases use nodes to store data entities, and edges to store relationships between entities. An edge always has a start node, end node, type, and direction, and an edge can describe parent-child relationships, actions, ownership, and the like. There is no limit to the number and kind of relationships a node can have.

A graph in a graph database can be traversed along specific edge types or across the entire graph. In

graph databases, traversing the joins or relationships is very fast because the relationships between nodes are not calculated at query times but are persisted in the database. Graph databases have advantages for use cases such as social networking, recommendation engines, and fraud detection, when you need to create relationships between data and quickly query these relationships.

Examples: Datastax Enterprise Graph, Neo4J

10) Time series databases

As the name implies, a time series database (TSDB) makes it possible to efficiently and continuously add, process, and track massive quantities of real-time data with lightning speed and precision. While other database models have been used for these kinds of workloads in the past, TSDBs utilize specific algorithms and architecture to deal with their unique needs.

A time series database stores data as pairs of time(s) and value(s). By storing data in this way, it makes it easy to analyze time series, or a sequence of points recorded in order over time. A TSDB can handle concurrent series, measuring many different variables or metrics in parallel.

Early time series databases were primarily used for processing volatile financial data and streamlining securities trading. However, the world's changed a lot since they were first introduced and many new use cases have emerged as technology has continued to evolve.

Examples: Druid, eXtremeDB, InfluxDB

****