**ARINDA HILLARY**

**DBMS**

**ASSIGNMENT 1**

**2/26/2024**

1. Make a list of all functionalities that a DBMS should offer. Think of all the possibilities you might ever want from a DBMS.
2. Data Definition Language (DDL) for defining the database schema.
3. Data Manipulation Language (DML) for querying and modifying data.
4. Data Storage, Retrieval, and Indexing for efficient data access.
5. Data Security and Access Control to ensure only authorized users can access and modify data.
6. Transaction Management to maintain data consistency and integrity.
7. Concurrency Control to handle multiple users accessing and modifying data simultaneously.
8. Backup and Recovery to protect against data loss and restore data in case of failures.
9. Data Integrity Enforcement to maintain the accuracy and consistency of data.
10. Query Optimization for improving the performance of database queries.
11. Replication and Sharding for distributing data across multiple servers for scalability and availability.
12. Data Migration to transfer data between different databases or platforms.
13. Logging and Auditing to track changes to the database for compliance and troubleshooting purposes.
14. Data Warehousing and Business Intelligence for analyzing and reporting on large volumes of data.
15. Data Compression and Encryption for efficient storage and secure transmission of data.
16. Database Monitoring and Administration for managing and monitoring database performance and health.
17. What are the 5 major components of DBMS environment? Which one of them do you think we'll be mostly concentrating on in this course?
18. **Software**: The software system that manages the database and provides various functionalities. Includes network software if the DBMS is being used over a network.
19. **Hardware**: The physical hardware on which the DBMS software runs, including servers, storage devices, etc.
20. **Data**: The data acts as a bridge between the machine components and the human components. The database contains both the operational data the meta-data.
21. **People**: This includes the database designers, database administrators (DBAs), application programmers and the end-users.
22. **Procedures**: These are instructions and rules that govern the design and use of the database. This includes instructions on how to log on to the DBMS, make up backup copies of the database, and how to handle hardware or software.

In this course, we'll likely be mostly concentrating on the DBMS software component, understanding its features, functionalities, and how to interact with it effectively.

1. An application needs data from the database. To get data from the database we typically use SQL. We can define the SQL statements in the application programs or in the database itself. Explain the advantages and disadvantages of both these approaches.

**#Defining SQL Statements in Application Programs:**

**Advantages:**

1. **Greater Control:** Developers have full control over the SQL statements, allowing them to tailor queries to specific requirements and optimize them based on application needs.
2. **Customization**: SQL statements can be customized dynamically based on user inputs, business logic, or other external factors.
3. **Portability**: Application-specific SQL statements are portable across different database systems, enabling easier migration or integration with other systems.
4. **Debugging**: Easier to debug and troubleshoot SQL statements directly within the application code, as developers can trace the execution flow and identify any issues more effectively.

**Disadvantages**:

1. **Increased Complexity**: Embedding SQL statements within application code can lead to increased complexity, especially in large-scale applications with numerous SQL queries scattered throughout the codebase.
2. **SQL Injection Vulnerabilities:** Directly concatenating user inputs into SQL queries can expose the application to SQL injection attacks if proper input validation and parameterization techniques are not implemented.
3. **Maintenance Challenges:** Maintenance becomes more challenging as SQL logic is spread across multiple application components, making it harder to track and update SQL statements when needed.
4. **Performance Impact:** Executing SQL queries within application code may introduce performance overhead, especially if the same query is executed repeatedly without proper optimization.

**#Defining SQL Statements in the Database Itself:**

**Advantages:**

1. **Centralized Logic:** SQL statements are encapsulated within the database, promoting a centralized approach to managing database operations and improving maintainability.
2. **Security**: Reduces the risk of SQL injection attacks, as SQL statements are predefined and encapsulated within stored procedures or views, minimizing direct user input into queries.
3. **Performance Optimization:** Database engines can optimize execution plans for stored procedures and views, leading to potential performance benefits compared to ad-hoc SQL queries executed from application code.
4. **Encapsulation**: Encapsulation of SQL logic within stored procedures or views abstracts the underlying data model, making it easier to enforce data access policies and maintain data integrity.

**Disadvantages**:

1. **Limited Flexibility:** Predefined SQL statements within the database may limit flexibility, as developers have less control over query customization and dynamic adjustments based on application requirements.
2. **Vendor Lock-in:** Dependency on database-specific stored procedures or views may introduce vendor lock-in, making it harder to migrate to a different database platform in the future.
3. **Debugging Challenges:** Debugging stored procedures or complex SQL views can be more challenging compared to debugging SQL statements within application code, as developers may have limited visibility into the execution context.
4. **Performance Impact**: Poorly optimized stored procedures or views can degrade database performance, especially if they involve complex logic or inefficient query execution plans.

1. Applications that access the database typically do the following actions:

1. Insert new data

2. Update existing data

3. Delete existing data

4. Search (query) data

Describe the different characteristics of *insert*, *update* and *delete* compared to *search*.  
In other words, how does *insert, update* and *delete* differ from *search* in a typical database application?

Think in terms of performance, multi-user access, data integrity, etc.

The operations Insert, Update, Delete, and Search (Query) in a database application have distinct characteristics, especially when considering performance, multi-user access, and data integrity:

1. **Insert Operation**

- **Performance**: Insert operations add new records to a database. Performance is generally quick but can be impacted by factors such as the need to update indexes, triggers that may execute on insertion, and the complexity of integrity constraints that must be checked.

- **Multi-User Access**: Concurrency control is critical during insert operations to prevent conflicts like two users trying to insert into the same spot. Transactional locks or multi-version concurrency control (MVCC) are used to handle this.

- **Data Integrity**: Insert must adhere to all database constraints like foreign keys, unique constraints, and check constraints to maintain data integrity.

1. **Update Operation:**

- **Performance**: Update modifies existing records and can be resource-intensive if it affects many rows or triggers complex integrity checks or index updates.

- **Multi-User Access**: Similar to insert operations, updates require careful concurrency control to prevent conflicts, such as lost updates or dirty reads, by using locks or MVCC.

- **Data Integrity:** Updates must maintain data integrity by adhering to constraints and can involve cascading actions if foreign keys are set to cascade on update.

1. **Delete Operation:**

- **Performance**: Delete operations can be quick for a few records but may become complex if there are cascading deletes or a large number of index updates.

- **Multi-User Access:** Deletes require concurrency mechanisms to ensure that no user reads or writes to the data being deleted. This often involves exclusive locks on the affected rows.

- **Data Integrity:** Deletes must also maintain referential integrity, ensuring that no orphaned records are left behind unless explicitly allowed by the database design.

1. **Search (Query) Operation:**

- **Performance:** Search operations are generally read-only and can range from very fast for simple indexed searches to slow for complex queries involving full table scans, joins, and sorting.

- **Multi-User Access:** Since searches do not modify data, they are less disruptive in a multi-user environment. However, they still need to provide consistent views of the data, which is managed by MVCC or shared locks.

- **Data Integrity:** Search operations are not directly responsible for maintaining data integrity, but they must reflect the current state of the database, considering all integrity constraints.

In summary, Insert, Update, and Delete operations change the state of the database and thus involve significant considerations for maintaining data integrity and handling concurrent access to ensure consistency and prevent data anomalies. In contrast, Search operations are primarily concerned with data retrieval and are optimized for performance through indexing and query optimization techniques.

1. Research the following databases:

1. Oracle

2. MySQL

3. MongoDB

4. ElasticSearch

For each of these databases describe the following:

* + Most important characteristics
  + In what situation would you use this database.

1. **Oracle Database**:

**Most Important Characteristics**:

* Oracle is a multi-model database management system that supports a wide range of data models, including document, graph, relational, and key-value.
* It is known for its robustness, scalability, and a multitude of features like Real Application Clusters (RAC), Data Guard for disaster recovery, and Automatic Storage Management (ASM).
* Oracle databases have strong consistency, comprehensive backup and recovery capabilities, and advanced security features.
* They typically use PL/SQL as the procedural language extension to SQL.
  + **Usage Situations**:
  + Oracle is suitable for large enterprises requiring high performance, reliability, and security.
  + It is often used in banking, financial services, and any other area where complex transactions or large-scale data warehousing is needed.

1. **MySQL**:

**Most Important Characteristics**:

* + - MySQL is an open-source relational database management system.
    - It is known for its ease of use, reliability, and compatibility with numerous platforms and programming languages.
    - MySQL uses a simple data storage mechanism, making it a good choice for web applications.
    - It is commonly used with the LAMP stack (Linux, Apache, MySQL, PHP/Perl/Python).
  + **Usage Situations**:
* MySQL is ideal for web applications and small to medium-sized databases where cost is a significant factor.
* It is often used in web development environments, small to medium enterprise applications, and as an embedded database.

1. **MongoDB**:

**Most Important Characteristics**:

* + - MongoDB is a document-oriented NoSQL database designed for high volume data storage.
    - It stores data in flexible, JSON-like documents where fields can vary from document to document.
    - It offers features like secondary indexes, aggregations, and a powerful query language.
    - It is designed for scalability and developer agility with a schema-less design that allows for the quick iteration of applications.
  + **Usage Situations**:
* MongoDB is well-suited for applications that require a flexible schema and the ability to scale horizontally, like big data applications, content management, and mobile and social infrastructure.

1. **Elasticsearch**:

**Most Important Characteristics**:

* Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases.
* It is known for its powerful full-text search capabilities, real-time indexing, and scalability.
* Elasticsearch is part of the ELK stack (Elasticsearch, Logstash, Kibana) and is often used for log and event data analysis.
* It supports complex search queries and is schema-free, using JSON documents to store data.
  + **Usage Situations**:
* Elasticsearch is ideal for search applications, real-time analytics, and situations where you need to analyze and visualize large volumes of data in near real-time.
* It is commonly used in logging and log analysis, search engines, and business analytics.