By continuously optimizing queries, organization can adapt to changing requirements and leverage new technologies to further improve performance.

Q11.

	Relational Algebra	Relational calculus.
Formalism	Procedural query language specifying operations	Non-procedural query language specifying predicates
Focus.	operations on relations (table)	selection of tuples from relations based on predicates.
operations	selection, projection, union, intersection, etc	Use mathematical predicates and logical formulas.
Expressiveness	Limited compared to calculus	More expressive and declarative.
Quantifiers	Does not include quantifiers	Include existential ($\exists$) and universal ($\forall$) quantifiers.
Use	Basis for query optimization and execution	Theoretical foundation for understanding query languages.

the describe the basics features of ODBMS and what are the advantages?.

Object-oriented db management system are dbms that store data in the form of objects similar to how they are represented in OOPL.

Features of ODBMS :-

1. support for complex data

ODBMS supports complex data types such as objects arrays and user defined datatypes. This allows for more flexible data modeling compared to traditional relational db.

2. Encapsulation

↳ object encapsulate both data and behaviour allowing for more natural representation and and manipulation of real world entities. It also helps maintain data integrity and provides abstraction.

3. Inheritance

ODBMS supports inheritance, allowing objects to inherit attributes and methods from parent classes. This promotes code reuse and simplifies data modeling by enabling the creation of hierarchical structures.

4. Polymorphism

↳ It allows object of different types to be treated uniformly. It supports polymorphic behaviour, enabling more flexible and extensible application.

5. Complex Relationships

↳ It supports complex relationships b/w objects, including one-to-one, one-to-many, and many to many relationships. This enables more natural representation of relationship b/w entities.

6. Query language.

↳ It typically provides a query language tailored to work with objects, such as Object Query language (OQL). This language allows users to query and manipulate objects directly.

7. Concurrency Control and Transaction Management

↳ It supports concurrency control mechanisms to ensure data consistency in multi-user environments. It also provides transaction management features to maintain data integrity during transaction operations.

Advantages of ODBMS.

1. Better Representation of Real world entities.
   - It provides a more natural representation of real world entities by allowing data and behavior to be encapsulated within objects. This simplifies data modeling and promotes code reusability.

2. Increased Productivity
   - The use of object oriented concepts such as inheritance and polymorphism can lead to more modular and maintainable code. This can result in increased developer productivity and shorter development cycles.

3. Improved Performance
   - ODBMS can offer better performance for certain types of applications, especially those with complex data structures and relationships. By directly mapping objects to db entities, ODBMS can reduce the need for complex joins and improve query performance.

4. Flexible data Modeling
   - ODBMS allows for flexible data modeling, making it easier to evolve and adopt the db schema as the application

requirements change. This flexibility can result in more agile and responsive systems.

5. Reduced Mapping overhead

↳ Unlike RDBMS which require an object-relational model (ORM) layer to map objects to tables, ODBMS eliminates this mapping overhead by directly storing objects in the DB. This can simply application development and reduce complexity.

6. Support for complex data.

↳ It support complex data types such as arrays and user defined types, which can be more difficult to represent in RDB. This enables richer data modeling and more expressive data structures.

Q.13 Explain various locking techniques for concurrency control.

It is a crucial aspect of DB management system to ensure data consistency and integrity in multi user environments.

Locking techniques are commonly used to manage concurrent access to data by multiple transaction

1. Binary locks (Binary locking)

Description

↳ Also known as 1- Lock or binary locking

This technique uses a single lock to control access to a resource. A transaction acquires the lock before accessing the resource and releases it afterward.

Advantages – simple to implement and understand.

Disadvantages
↳ can lead to contention and reduced concurrency as only one transaction can access the resource at a time.

2. Shared / Exclusive locks (Read / Write locks)

Description
↳ This technique distinguishes between shared lock (read locks) and exclusive locks (write locks). Multiple transaction can hold shared locks simultaneously, allowing for concurrent reads. However, exclusive locks are exclusive, meaning only one transaction can hold a write lock at a time

Advantages → Allows for concurrent read access while ensuring exclusive write access.

Disadvantages → can still lead to contention for write access if multiple transaction require exclusive locks.

5. Two Phase locking (2PL)

Description → Two-Phase locking is a concurrency control method that requires transactions to acquire locks in two Phase : (i) the growing phase and the (ii) shrinking Phase.

In growing Phase, transactions acquire locks on data items they need, but they cannot release any locks.

In shrinking phase, transaction release locks but cannot acquire any new locks.

Advantages. ↳ ensures serializability and prevents issues like lost update and inconsistent retrievals.

Disadvantages. → May lead to deadlock situation if not managed properly and may limit concurrency due to strict lock acquisition rules.

4. Timestamp ordering.

Description → Each transaction is assigned a unique timestamp and transaction and ordered based on their timestamps. Conflicts are resolved by comparing the timestamps of conflicting transaction with the older transaction usually given Priority.

Advantages:
↳ Provides a strict and efficient mechanism for enforcing serializability without explicit locks.

Disadvantages.
↳ Requires a reliable timestamp generation mechanism and may result in high abort rate if transactions frequently conflict.

5. Optimistic concurrency control

Description → In optimistic concurrency control, transaction proceed without acquiring locks initially. Instead they perform their operation and validate them at the end of the transaction. If conflict are detected during validation, transactions may need to be aborted and retried.

Advantages
↳ Reduces contention and lock overhead, potentially leading to higher concurrency.

Disadvantages.
↳ May result in increased abort rate if conflicts are frequent, and may not be suitable for highly concurrent environments with high contention.

6 Multi - Version Concurrency Control

Description:

↳ MVCC allows for multiple versions of data items to exist concurrently. Read operations access the appropriate version based on the transaction's timestamp, while write operations create new versions of data items.

Advantages:

↳ Allows for consistent read operations without blocking write operations, leading to improved concurrency.

Disadvantages

↳ Requires additional storage space to maintain multiple versions of data items and may result in increased complexity.

Q14. Describe optimistic concurrency control technique.

Ans. Optimistic concurrency control (OCC) is a technique used in DBMS to manage concurrent access to data without acquiring locks preemptively.

Instead of locking data items during transaction optimistic concurrency control allows transactions to proceed independently and validates them at the end to ensure data consistency.

OCC works.

1. Transaction Execution

↳ In optimistic concurrency control, transactions are allowed to execute without acquiring locks on data items up front. This means that multiple transaction can access the same data items concurrently without being blocked by locks.

2. Validation Phase

↳ After a transaction completes its operation (reads and writes), it enters the validation phase. During this phase the DBMS checks whether the transaction's operations have caused any conflicts with concurrent transactions.

3. Conflict Detection

↳ The DBMS examines the changes made by transaction and compares them with changes made by other concurrent transactions. Conflicts may arise if multiple transactions attempt to modify the same data item concurrently or if a transaction reads data that has been modified by another transaction.

4. Conflict Resolution

↳ If conflicts are detected during the validation phase, the DBMS take appropriate actions to resolve them. This may involve aborting one or more transactions and rolling back their changes to maintain data consistency.

5. Commit or Abort

↳ If no conflicts are detected during the validation phase, the transaction is considered valid and can proceed to commit its changes to the db. However if conflicts are detected, the transaction may need to be aborted and its changes rolled back.

DIS. Advantages and Disadvantages of OCC.

1. Reduced locking overhead
   ↳ It reduces the overhead associated with acquiring and releasing locks as transactions are allowed to proceed without blocking.

2. Increased concurrency
   ↳ By allowing transaction to execute concurrently without acquiring locks preemptively, optimistic concurrency control can improve overall system concurrency and performance.

3. lower contention
   ↳ Since transaction do not hold locks during execution, there is less contention for resources, leading to reduced wait time and improved through put.

4. Improved Scalability
   ↳ optimistic concurrency control can lead to better scalability in systems with high concurrency levels, as it minimizes the impact of lock contention on system performance.

5. Suitability for Read Mostly workloads.

↳ It is particularly well-suited for workloads with a high Proportion of read operations as it allow read operation transaction to proceed without being blocked by write transactions.

Dis advantages of OCC.

1. Potential for Increased Abort Rates

↳ It may result in higher transaction abort rates if conflicts are frequent, leading to increased overhead due to transaction restarts.

2. Delayed Conflict Detection

↳ Conflict detection occurs after transactions have completed ther operations, which means that conflicts may not be detected until late in the transaction lifecycle, potentially wasting resources on transaction that ultimately need to be aborted.

3. Limited Applicability

It may not be suitable for all types of workloads, particularly those with high contention for resources or frequent conflicts b/w transactions.

Q.16: Define Meta data with examples.

Meta Data → Metadata is Data that provides information about other data. It describes the content structure, format, relationships and other properties of data elements.

Examples of MetaData

1. Table Metadata

↳ In RDBMS, Metadata about tables includes information such as table name, Column names, datatypes, constraints (eg: PK, FK) indexes and table size.

2. File Metadata

↳ Metadata associated with files includes attributes such as file name, file size, file type, creation data, modification data, and file Permissions

3. DB Schema MetaData

↳ Metadata describing the db schema. includes information about tables, views, indexes, constraints and other db objects

4. API Documentation

↳ Metadata for APIs includes description of endpoints, parameters, request/response formats authentication methods and error codes.

5. Data Dictionary

↳ A data dictionary is a centralized repository of metadata that provides definations and description of data elements, including their meanings, relationships and usage within an organisation.

Q17 What is nested query and how it is different from a correlated query?.

Nested Query → A nested query or subquery is a query embedded within another SQL query. It is enclosed with Paranthesis and can be used in various parts of an SQL statement, such as the SELECT, WHERE, FROM and Having clauses.

Correlated Queries → A correlated query is subquery that references columns from the outer query The subquery is executed once for each row processed by the outer query, making it dependent on the outer query.

Aspect	Nested Query	Correlate Query
Defination	Query within another Query.	sub Query that reference Columns from the outer Query.
Execution order	Inner Query executed first then outer Query.	Outer Query executed first, subquery executed for each row.
Dependency	Independent of the outer Query	Dependent on the outer Query.
Performance	Generally more efficient if properly indexed	Can be less efficient due to multiple executions of subquery.
use case	suitable for static comparisons	suitable for row by row comparisons.
Ex:-	WHERE department_id = (Select ...)	where salary > (Select Avg (Salary) From where (...))
Complexity	used for filtering results based on a single Query results. more complex	used for condition involving row-specific data comparisons. less complex
Result scope	Can return a single value or a set of values.	Typically returns a single value for each row of outer Query.

Q18. What is unsafe query give an example. Why it is important to disallow such queries.

Ans. An unsafe query, also known as an unsafe operation or unsafe SQL query.

It is a query that can potentially produce unintended or harmful results.

It typically involve the use of dynamic or user controlled input directly in SQL statement without proper validation or sanitization.

These SQL statement can lead to security vulnerabilities such as SQL injection attacks, data breaches, or unauthorized access to sensitive information.

Example of an unsafe query.

Consider a web application that allows users to search for products by entering keywords. The App. constructs an SQL query dynamically based on the user input, without properly validating or sanitizing it.

[ ]

In this example, if a malicious user enters the following input in the search field.

[ ]

The resulting SQL query becomes:

[ ]

This query will return all rows from the table because the condition '1=1' always evaluates to true, effectively bypassing any filtering based on the product name.

Importance of disallowing unsafe queries

1. SQL injection Attacks.

↳ unsafe queries are vulnerable to SQL injection attacks, where attackers can manipulate the query's structure by injecting malicious SQL code. This can lead to unauthorized access to data, data manipulation or even complete database compromise.

2. Data Breaches

↳ unsafe queries can result in data breaches by exposing sensitive information stored in the db to unauthorized users.

Attackers can exploit vulnerabilities in the application to retrieve confidential data such as user credentials financial records, or personal information.

3. Data Integrity

↳ Unsafe queries can compromise data integrity by allowing unauthorized modifications or deletions of data.

Attackers may inject malicious SQL code to alter or records, insert fraudulent data or delete critical information, leading to data loss or corruption.

4. Legal and compliance Risks

↳ Data breaches resulting from unsafe queries can have legal and compliance implications for organisation, including regulatory fines, lawsuits damage to reputation, and loss of customer trust.

Disallowing unsafe queries helps mitigate these risks by ensuring the security and privacy of sensitive data.

5. Application Reliability

↳ Unsafe queries can impact the reliability and availability of applications by introducing vulnerabilities that could by exploited by attackers to disrupt services, manipulate data or compromise system resources.

Disallowing unsafe queries helps maintain the integrity and reliability of the Application.

Q.19. What is update anomalies explain with examples.

Ans. Update anomalies refer to inconsistencies or unintended changes that can occur in a DB when updating data.

These anomalies arise due to the redundancy or denormalization of data, where the same information is stored in multiple places within the DB.

There are three main types of update anomalies

(i) Insertion
(ii) Deletion
(iii) Modification anomalies.

1. Insertion Anomalies.

↳ It occurs when it is not possible to add data to the DB without also adding unrelated data.

This typically happens when adding a new record requires the inclusion of values that are not applicable to the new record.

Example. Consider a denormalized table that combines information about students and the courses they are enrolled in.

StudentID	StudentName	CourseID	CourseName	Instructor
1	Alice	C101	Math.	Smith
2	Bob	C102	Science	Jones.