NEPAL COLLEGE OF INFORMATION TECHNOLOGY

BALKUMARI LALITPUR

(Affiliated To Pokhara University)

SUBJECT : Database Management System

ASSIGNMENT # 4

Submitted By :

Name : Pradip Dhungana

Roll No : 201751

Semester : 4th

Submitted To :

Name : Amit K. Shrivastava

Department of Software

1. What are the roles of Assertions and Triggers in SQL? Consider following bank database :

Branch-schema = (branch-name, branch-city, assets)
Loan-schema = (loan-number, branch-name, amount)

Write an assertion for the bank database to ensure that the Assets value for the Perryridge branch is equal to the sum of all the amounts lent by the Perryridge branch.

Answer :

ASSRETIONS :

An assertion is a predicate expressing a condition that we wish the database always to satisfy. An assertion in SQL takes the form :

create assertion <assertion-name> **check** <predicate>

When an assertion is made, the system tests it for validity, and tests it again on every update that may violate the assertion. This testing may introduce a significant amount of overhead; hence assertions should be used with great care.

Asserting for all X, P(X) is achieved in a round-about fashion using not exists X such that not P(X).

TRIGGERS :

A trigger is a statement that is executed automatically by the system as a side effect of a modification to the database.

To design a trigger mechanism, we must:

- Specify the conditions under which the trigger is to be executed.
- Specify the actions to be taken when the trigger executes.

Triggers introduced to SQL standard in SQL:1999, but supported even earlier using non-standard syntax by most databases.

SOLUTION :

CREATE ASSERTION assets_sum_check CHECK ((SELECT assets FROM Branch WHERE branch-name = 'Perryridge') = (SELECT SUM(amount) FROM Loan WHERE branch-name = 'Perryridge'));

2. What is Normalization and why it is done? Give an example of a relation schema R and a set of dependencies such that R is not in 2NF and normalize it into 2NF.

Answer :

I. Normalization of data is a process that involves analyzing relation schemas based on their functional dependencies and primary keys.
II. The goal of normalization is to minimize redundancy and avoid insertion, deletion, and update anomalies in the database.
III. It acts as a filtering or purification process to improve the quality of the database design.
IV. Unsatisfactory relation schemas that do not meet certain conditions, known as normal form tests, are decomposed into smaller relation schemas that meet the tests and possess desirable properties.
V. Normalization provides a formal framework for analyzing relation schemas based on keys and functional dependencies, and it offers a series of normal form tests to guide the process of achieving a well-structured and normalized database.

Solution to Example :

Consider the relation schema R with attributes A, B, C, and D, and the following set of dependencies :

R(A, B, C, D) Functional dependencies:

I. A -> B
II. B -> C

R is not in 2NF because it violates the requirement that every non-prime attribute (attributes not part of any candidate key) must be fully functionally dependent on the primary key.

Normalization to 2NF:

I. Identify the candidate key(s) of R. In this case, let's assume that {A} is a candidate key.

II. Create a new relation schema R1(A, B) to hold the attributes A and B, and R2(B, C, D) to hold the attributes B, C, and D.

III. Remove the attributes B, C, and D from the original relation schema R.

The resulting normalized relations are:

R1(A, B)

R2(B, C, D)

Now, R1 and R2 are in 2NF as all non-prime attributes (B, C, and D) are fully functionally dependent on the candidate key(s) (A and B) in their respective relations.

3. What do you mean by Integrity Constraints? Explain its types.

Answer :

An integrity constraint is a mechanism used by DBMS to prevent invalid data entry into the table. It has enforcing the rules for the columns in a table. The types of the integrity constraints are:

a) Domain Integrity b) Entity Integrity c) Referential Integrity

a) Domain Integrity

This constraint sets a range and any violations that take place will prevent the user from performing the manipulation that caused the breach. It includes:

I) Not Null constraint:

- ✓ Enforces that a column in a table must have a value.
- ✓ Makes the column mandatory, disallowing null values.
- ✓ Null values are not equivalent to zero and are used when the value is unknown or not meaningful.
- ✓ Not null constraints cannot be added if the table already contains rows.

II) Check Constraint:

- ✓ Allows only specific values or a range of values to be stored in a column.
- ✓ Ensures data manipulation adheres to the specified condition.
- ✓ Cannot contain subqueries.
- ✓ Used to define rules and restrictions on column values.

E.g: Create table student (regno int, mark int, constraint b check (mark >=0 and mark <=100));

b) Entity Integrity

I. Entity integrity ensures uniqueness in a record.
II. Each row in a table represents an instance of an entity, and entity integrity maintains the uniqueness of each row.

It consists of two constraints:

I) Unique key constraint :

- ✓ Ensures values in a specified column or combination of columns are unique.
- ✓ Prevents duplication of values within the specified column(s).
- ✓ Allows null values in the column(s). Supports multiple columns, creating a composite unique key.
- ✓ Maximum of 16 columns can be included in a composite unique key.

II) Primary Key Constraint :

- ✓ Uniquely identifies each row in a table. Avoids duplication of rows.
- ✓ Does not allow null values.
- ✓ A table should have only one primary key.
- ✓ Can be assigned to one or more columns.
- ✓ Combination of columns forms a composite primary key.
- ✓ Composite primary key can contain up to 16 columns.

c) Referential Integrity

- ➢ Referential integrity enforces the relationship between tables in a database.
- ➢ It establishes a parent-child relationship between two tables that share a common column definition.
- ➢ To implement referential integrity, we define the column in the parent table as the primary key.
- ➢ The same column in the child table is then defined as a foreign key, referring to the corresponding entry in the parent table.
- ➢ The foreign key ensures that any value in the child table's column must exist as a primary key value in the parent table.
- ➢ This constraint helps maintain the consistency and integrity of data between related tables.

4. Give an example of a relation schema R and a set of dependencies which illustrates the importance of 4NF.

Answer :

A relation schema R is in 4NF with respect to a set of dependencies F (that includes functional dependencies and multivalued dependencies) if, for every nontrivial multivalued dependency X → → Y in F + , X is a superkey for R.

To illustrate the importance of 4NF,

- EMP relation in Figure 16.4(a) has 16 tuples.
- Decomposing EMP into EMP_PROJECTS and EMP_DEPENDENTS (Figure 16.4(b)) reduces the total number of tuples to 11.
- This decomposition saves on storage and avoids update anomalies associated with multivalued dependencies (MVDs).
- MVDs can cause inconsistencies and require modifying multiple tuples when updating.
- 4NF normalization eliminates these update anomalies and allows efficient updates with fewer tuple modifications.

Figure 16.4
Decomposing a relation state of EMP that is not in 4NF. (a) EMP relation with additional tuples. (b) Two corresponding 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(a) EMP

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John
Brown	W	Jim
Brown	X	Jim
Brown	Y	Jim
Brown	Z	Jim
Brown	W	Joan
Brown	X	Joan
Brown	Y	Joan
Brown	Z	Joan
Brown	W	Bob
Brown	X	Bob
Brown	Y	Bob
Brown	Z	Bob

(b) EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y
Brown	W
Brown	X
Brown	Y
Brown	Z

EMP_DEPENDENTS

Ename	Dname
Smith	Anna
Smith	John
Brown	Jim
Brown	Joan
Brown	Bob

5. State and explain with example about functional dependency ,transitive dependency and multivalued dependency.

Answer :

a) Functional Dependency :

i. Functional dependency occurs when one or more attributes determine the values of other attributes in a relation.
ii. It is denoted by the symbol "→".
iii. If we know the value of one attribute, we can determine the value of another attribute(s).

Example :

Consider a table named "Employees" with the following attributes: Employee_ID, Employee_Name, Department_ID, and Department_Name.

Let's assume that each employee is assigned to only one department.

Employee_ID	Employee_Name	Department_ID	Department_Name
1	John	101	HR
2	Jane	102	Finance
3	Alice	101	HR
4	Bob	103	Sales

In this table, we can observe that the Department_Name is functionally dependent on the Department_ID. Given the Department_ID, we can determine the corresponding Department_Name.

b) Transitive Dependency

i. Transitive dependency occurs when an attribute depends on another attribute through a third attribute.
ii. It is denoted by the symbol "=>".
iii. If A depends on B and B depends on C, then A transitively depends on C.

Example :

Consider a table named "Students" with the following attributes: Student_ID, Course_ID, and Course_Teacher.

Let's assume that each student enrolls in multiple courses, and each course is taught by only one teacher.

Student_ID	Course_ID	Course_Teacher
1	101	John
1	102	Jane
2	101	John
2	103	Alice

In this table, the Course_Teacher is transitively dependent on the Student_ID through the Course_ID. By knowing the Student_ID, we can determine the Course_Teacher indirectly through the Course_ID.

c) Multivalued Dependency

i. Multivalued dependency occurs when two or more attributes depend on the same attribute(s) but independently of each other.
ii. It is denoted by the symbol "|-".
iii. The values of the dependent attributes can vary independently for the same set of determining attributes.

Example :

Consider a table named "Books" with the following attributes: Book_ID, Book_Name, Author, and Genre.

Let's assume that each book can have multiple authors and belong to multiple genres.

Book_ID	Book_Name	Author	Genre
1	Book 1	Author 1	Fiction, Drama
2	Book 2	Author 2	Mystery
3	Book 3	Author 1	Fiction, Romance

In this table, we can observe that the attributes Author and Genre have a multivalued dependency on the Book_ID. The values of Author and Genre can vary independently of each other for the same Book_ID.

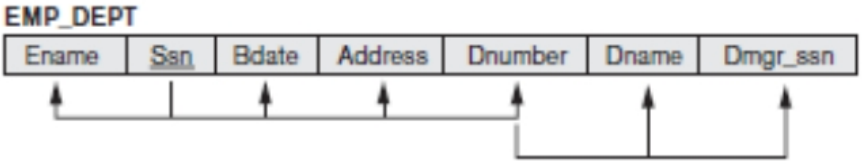
6. Give an example of a relation schema R and a set of dependencies such that R is not in 3NF and normalize it into 3NF.

Answer :

According to Codd's original definition, a relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key.

Example :

The relation schema EMP_DEPT in Figure below is in 2NF, since no partial dependencies on a key exist. However, EMP_DEPT is not in 3NF because of the transitive dependency of Dmgr_ssn (and also Dname) on Ssn via Dnumber.



We can normalize EMP_DEPT by decomposing it into the two 3NF relation schemas ED1 and ED2 shown in Figure below. Intuitively, we see that ED1 and ED2 represent independent entity facts about employees and departments. A NATURAL JOIN operation on ED1 and ED2 will recover the original relation EMP_DEPT without generating spurious tuples.

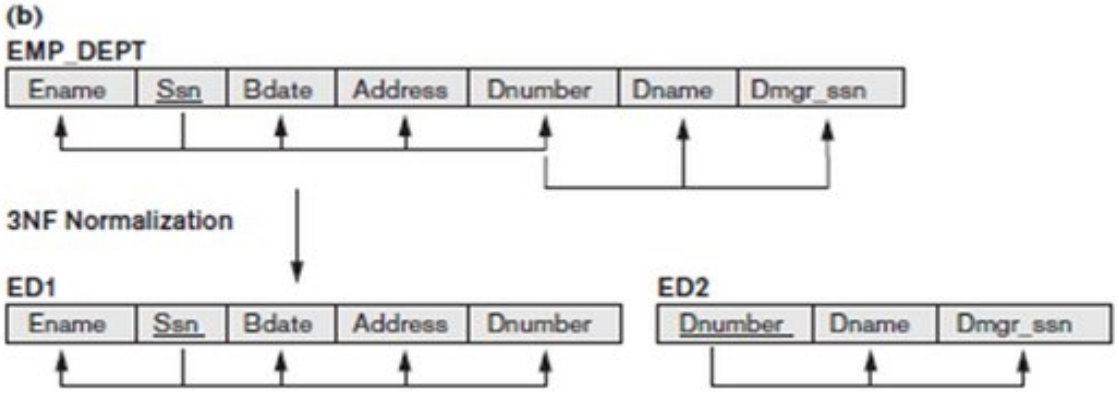


Figure 14.11
Normalizing into 3NF (b) Normalizing EMP_DEPT into 3NF relations.

7. How security can be granted using view explain. What are two advantages of encrypting data stored in the database?

Answer :

Security using views:

- ✓ Views can be used to grant security by controlling access to specific data.
- ✓ Access privileges can be granted to users or roles for the view, limiting their access to only the defined data.
- ✓ Views provide data restriction by allowing selective exposure of data. Instead of granting direct access to tables, views can be created with filters or restrictions, ensuring users only see authorized data.

Example :

Consider a table named "Employees" with the following attributes: Employee_ID, Employee_Name, Salary, and SSN (Social Security Number).

Employee_ID	Employee_Name	Salary	SSN
1	John	5000	123-45-6789
2	Jane	6000	987-65-4321
3	Alice	4000	246-80-1357

- ➢ Query to create a view for restricted access :

```
CREATE VIEW Restricted_Employees AS(
    SELECT Employee_ID, Employee_Name FROM Employees);
```

- ➢ Query to grant access and retrieve data from the view :

```
GRANT SELECT ON Restricted_Employees TO User1;
SELECT * FROM Restricted_Employees;
```

Advantages of encrypting data stored in the database:

a) **Data Confidentiality:**

- ➢ Encryption ensures sensitive data remains confidential.
- ➢ Encrypted data is converted into an unreadable format, making it challenging for unauthorized individuals to access or decipher.
- ➢ Example: Encrypting the "SSN" column in the "Employees" table using an encryption algorithm.

- Query to encrypt the SSN column :

```
UPDATE Employees SET SSN = ENCRYPT(SSN, 'encryption_key');
```

b) **Data Integrity:**

- Encryption helps maintain data integrity by protecting it from unauthorized modifications.
- Any unauthorized changes to encrypted data would render it invalid or undecipherable.
- Example: Attempting to modify the encrypted "Salary" value.
- Query to modify the encrypted Salary value :

```
UPDATE Employees SET Salary = 8000 WHERE Employee_ID = 1;
```

8.What is Log-Based Recovery? Compare the deferred and immediate modification versions of the log-based recovery scheme in terms of ease of implementation and overhead cost.

Answer :

i. Log-based recovery in DBMS ensures data consistency and provides the ability to recover data in the event of system failure.

ii. A log file is created to record every transaction operation performed on the database.

iii. Before a transaction starts, it registers itself by writing a "<Ti start>" log record.

iv. When a transaction executes a write operation, a log record "<Ti, X, V1, V2>" is written, where X is the data item, V1 is the value before the write, and V2 is the value to be written.

v. When a transaction finishes its last statement, a "<Ti commit>" log record is written.

vi. Log records are written directly to stable storage to ensure durability and prevent loss in case of system failure.

vii. Log-based recovery is used in systems like Apache HBase, where a Write-Ahead Log (WAL) is maintained and replayed to reconstruct data and ensure consistency during system restart or recovery.

Deferred Modification	Immediate Modification
1) Relatively complex	1) Relatively simpler
2) Requires additional bookkeeping and mechanisms to handle deferred updates.	2) Easier to implement as updates are immediately applied to the database.
3) Lower overhead cost	3) Higher overhead cost
4) Fewer disk writes required as updates are accumulated in the log and applied during the redo phase.	4) More disk writes required as updates are applied directly to the database.
5) Lower resource utilization as updates can be grouped and applied in batches.	5) Higher resource utilization as updates are applied immediately.
6) Better performance for workloads with a high number of read operations and few write operations.	6) Better performance for workloads with a high number of write operations.

9. What is Log-Based Recovery? Explain Deferred Database modification and Immediate Database modification with an illustration.

Answer :

Log-based recovery in DBMS is a mechanism that ensures data consistency and provides the ability to recover data by keeping a record of every transaction on stable storage, allowing for easy access and reconstruction of the database in case of system failures.

There are two approaches to modify the database:

(a) Deferred database modification

(b) Immediate database modification

1. Deferred database modification:

❍The deferred modification technique occurs if the transaction does not modify the database until it has committed.

❍In this method, all the logs are created and stored in the stable storage, and the database is updated when a transaction commits.

❍Illustration:

Let's consider a scenario where we have two transactions, T1 and T2, modifying the same account balance.

- Transaction T1 deducts $200 from Account A, and Transaction T2 adds $300 to Account A.
- These modifications are recorded in the transaction log but not immediately applied to the database.
- During the commit phase, the accumulated changes are applied, resulting in the final balance of Account A.

2. Immediate database modification:

❍The Immediate modification technique occurs if database modification occurs while the transaction is still active.

❍In this technique, the database is modified immediately after every operation. It follows an actual database modification.

❍Illustration:

Consider a scenario where we have a single transaction, T3, inserting a new customer record into a "Customers" table.

- Transaction T3 inserts a new customer record for "John Doe" into the "Customers" table.
- The insert operation is immediately applied to the database, making the new record visible for other transactions.

10. Explain the basic steps in query processing.

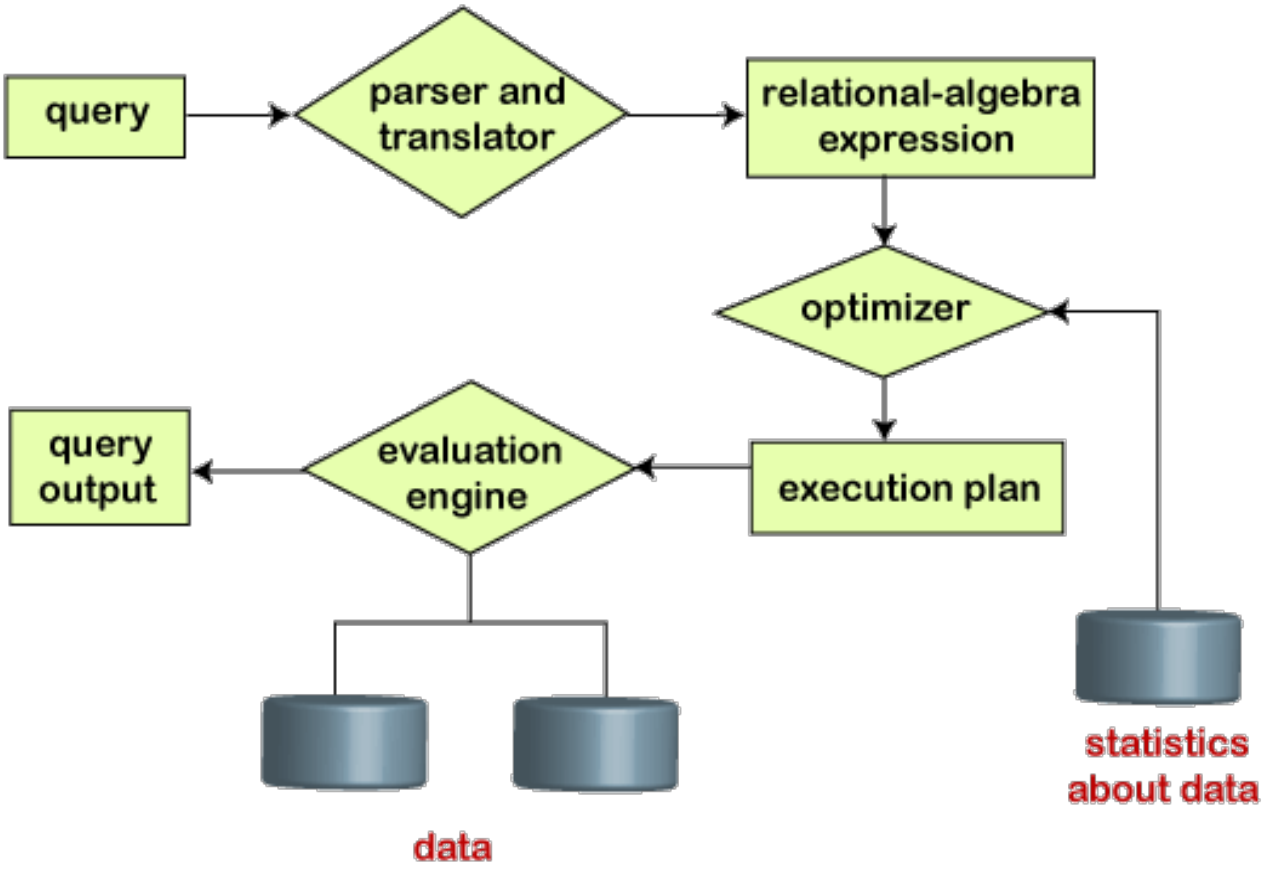
Answer :

Query Processing is the activity performed in extracting data from the database. In query processing, it takes various steps for fetching the data from the database. The steps involved are:

1.Parsing and translation

2.Optimization

3.Evaluation



Steps in query processing

(a) Parsing and Translation:

- Translate the given query into its internal form.
- Check the syntax of the query and verify the correctness of relation names.
- Construct a parse-tree representation of the query.
- Translate the parse tree into a relational-algebra expression.

(b) Evaluation:

- Relational-algebra operations, annotated with evaluation instructions, are used.
- The sequence of primitive operations forms a query-execution plan or query-evaluation plan.
- The plan outlines how to evaluate the query step-by-step.

(c) Optimization:

- Consider multiple equivalent evaluation plans and choose the one with the lowest cost.
- Cost estimation is based on statistical information from the database catalog, such as the number of tuples in each relation and the size of tuples.
- Different evaluation plans for the same query may have different costs.
- The goal is to find an optimal plan that minimizes the overall cost of query execution.

As an illustration, consider the query :

select emp_name from Employee where salary>10000;

Thus, to make the system understand the user query, it needs to be translated in the form of relational algebra. We can bring this query in the relational algebra form as:

σsalary>10000(πsalary(Employee))

πsalary(σsalary>10000(Employee))

After translating the given query, we can execute each relational algebra operation by using different algorithms. So, in this way, a query processing begins its working.

11. Explain the distinction between the terms serial schedule and serializable schedule. Consider the following two transactions:

```
T31: read(A);
     read(B);
     if A = 0 then B := B + 1;
     write(B).
T32: read(B);
     read(A);
     if B = 0 then A := A + 1;
     write(A).
```

Add lock and unlock instructions to transactions T31 and T32, so that they observe the two- phase locking protocol.

Answer :

Lock and unlock instructions:

```
T31:  lock-S(A)
      read(A)
      lock-X(B)
      read(B)
      if A = 0 then B := B + 1
      write(B)
      unlock(A)
      unlock(B)
T32:  lock-S(B)
      read(B)
      lock-X(A)
      read(A)
      if B = 0 then A := A + 1
      write(A)
      unlock(B)
      unlock(A)
```

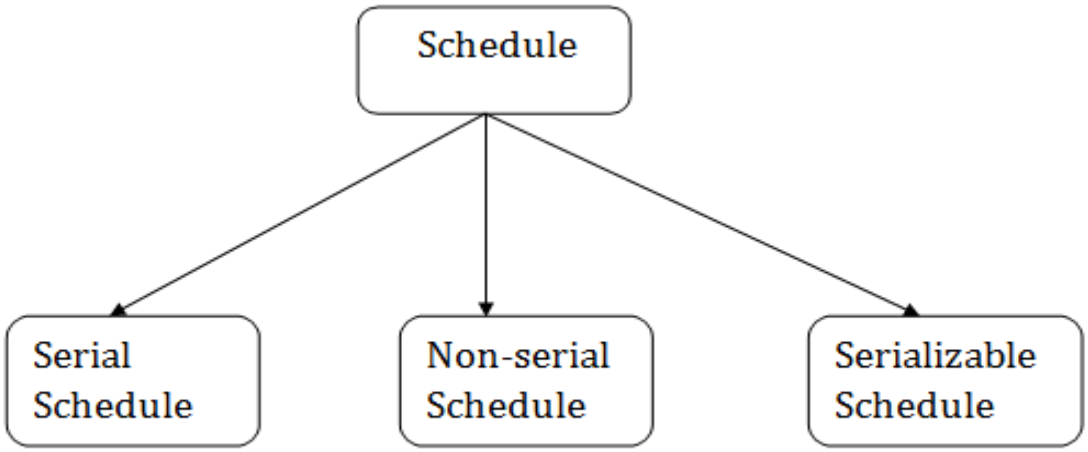
Serial Schedule	Serializable Schedule
• No concurrency; transactions are executed one after another.	• Transactions may execute concurrently.
• Provides strict isolation between transactions.	• Provides isolation between transactions, ensuring that their combined effect is equivalent to a serial execution.
• Lower concurrency, potentially slower performance.	• Higher concurrency, potentially improved performance.
• No conflicts as transactions do not overlap.	• Conflicts need to be resolved to maintain serializability.
• Not required since transactions are executed serially.	• Requires concurrency control mechanisms, such as locking, to ensure serializability.

12. What do you mean schedule and serialzability? What are view serialization schedules?

Answer :

Schedule :

- A schedule represents the order in which transactions are executed in a database system.
- It is a sequence of operations performed by transactions, including reads and writes on data items.
- The schedule depicts the interleaved execution of multiple transactions.



Serializability :

- Ensures equivalent results for concurrent transactions as if executed serially.
- Guarantees data consistency and correctness.
- Maintained through concurrency control mechanisms.
- Prevents conflicts and maintains data integrity.
- Enables correct execution of concurrent transactions.

View Serializability

- A schedule will view serializable if it is view equivalent to a serial schedule.
- If a schedule is conflict serializable, then it will be view serializable.

❍The view serializable which does not conflict serializable contains blind writes.

View Equivalent

Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions:

1. Initial Read

An initial read of both schedules must be the same. Suppose two schedule S1 and S2. In schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A.

T1	T2
Read(A)	Write(A)

Schedule S1

T1	T2
Read(A)	Write(A)

Schedule S2

Above two schedules are view equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1.

2. Updated Read

In schedule S1, if Ti is reading A which is updated by Tj then in S2 also, Ti should read A which is updated by Tj.

T1	T2	T3
Write(A)	Write(A)	Read(A)

Schedule S1

T1	T2	T3
Write(A)	Write(A)	Read(A)

Schedule S2

Above two schedules are not view equal because, in S1, T3 is reading A updated by T2 and in S2, T3 is reading A updated by T1.

3. Final Write

A final write must be the same between both the schedules. In schedule S1, if a transaction T1 updates A at last then in S2, final writes operations should also be done by T1.

T1	T2	T3
Write(A)	Read(A)	Write(A)

Schedule S1

T1	T2	T3
Write(A)	Read(A)	Write(A)

Schedule S2

Above two schedules is view equal because Final write operation in S1 is done by T3 and in S2, the final write operation is also done by T3.

Example:

T1	T2	T3
Read(A) Write(A)	Write(A)	Write(A)

Schedule S

With 3 transactions, the total number of possible schedule

= 3! = 6

S1=<T1 T2 T3>

S2=<T1 T3 T2>

S3=<T2 T3 T1>

S4=<T2 T1 T3>

S5=<T3 T1 T2>

S6=<T3 T2 T1>

Taking first schedule S1 :

T1	T2	T3
Read(A) Write(A)	Write(A)	Write(A)

Schedule S1

Step 1 : Final updation on data items

In both schedules S and S1, there is no read except the initial read that's why we don't need to check that condition.

Step 2 : Initial Read

The initial read operation in S is done by T1 and in S1, it is also done by T1.

Step 3 : Final Write

The final write operation in S is done by T3 and in S1, it is also done by T3. So, S and S1 are view Equivalent.

The first schedule S1 satisfies all three conditions, so we don't need to check another schedule.

Hence, view equivalent serial schedule is :

$$T1 \rightarrow T2 \rightarrow T3$$

13. What do you mean by a schedule? What are conflict serialization schedules?

Answer :

Conflict Serializable Schedule :

- A schedule is called conflict serializability if after swapping of non-conflicting operations, it can transform into a serial schedule.
- The schedule will be a conflict serializable if it is conflict equivalent to a serial schedule.

Conflicting Operations :

The two operations become conflicting if all conditions satisfy:

1. Both belong to separate transactions.
2. They have the same data item.
3. They contain at least one write operation.

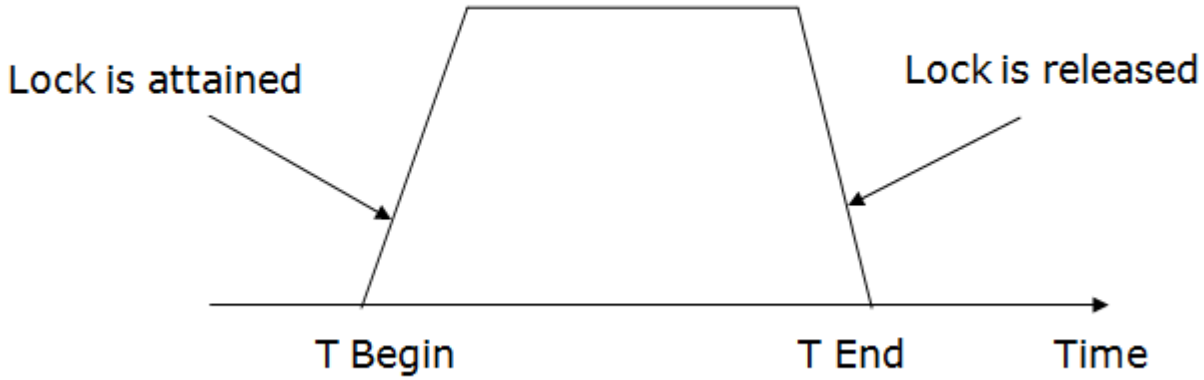
14. What do you mean by concurrency control? Describe Two phase locking Protocol.

Answer :

Concurrency Control is the management procedure that is required for controlling concurrent execution of the operations that take place on a database.

Two-phase locking (2PL)

- The two-phase locking protocol divides the execution phase of the transaction into three parts.
- In the first part, when the execution of the transaction starts, it seeks permission for the lock it requires.
- In the second part, the transaction acquires all the locks. The third phase is started as soon as the transaction releases its first lock.
- In the third phase, the transaction cannot demand any new locks. It only releases the acquired locks.



There are two phases of 2PL:

Growing phase : In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released.

Shrinking phase : In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired.

In the below example, if lock conversion is allowed then the following phase can happen:

1.Upgrading of lock (from S(a) to X (a)) is allowed in growing phase.

2.Downgrading of lock (from X(a) to S(a)) must be done in shrinking phase.

Example:

	T1	T2
0	LOCK-S(A)	
1		LOCK-S(A)
2	LOCK-X(B)	
3	——	——
4	UNLOCK(A)	
5		LOCK-X(C)
6	UNLOCK(B)	
7		UNLOCK(A)
8		UNLOCK(C)
9	——	——

The following way shows how unlocking and locking work with 2-PL.

Transaction T1:

Growing phase : from step 1-3

Shrinking phase : from step 5-7

Lock point : at 3

Transaction T2:

Growing phase : from step 2-6

Shrinking phase : from step 8-9

Lock point : at 6

15. Construct a B+ tree for the following set of key values:

(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the tree is initially empty and values are added in ascending order. Construct B+- trees for the case where the number of pointers that will fit in one node is Six. Also show the form of the tree after insertion of 9.

Answer :

B+-tree with maximum of six pointers per node:

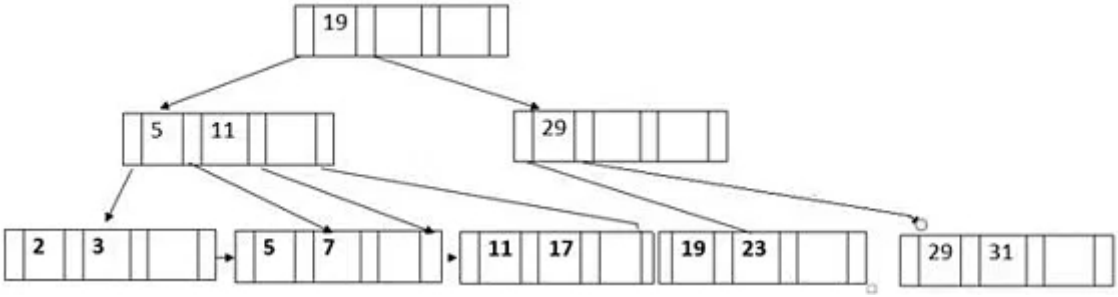
16. Construct a B+ tree for the following set of key values:

(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the tree is initially empty and values are added in ascending order. Construct B+- trees for the case where the number of pointers that will fit in one node is Four. Also show the form of the tree after deletion of 23.

Answer :

B+-tree with maximum of four pointers per node :



17. What are the reasons for building distributed database systems. Also discuss some of the disadvantages of distributed database.

Answer :

Reasons for building distributed database systems :

- ✔ Place unrelated
  - Data stored and accessed from multiple geographic locations.
- ✔ Spread-out query processing
  - Parallel processing of queries across multiple nodes improves performance.
- ✔ The administration of distributed transactions
  - Management of transactions spanning multiple nodes.
- ✔ Independent of hardware
  - Flexibility in choosing hardware platforms for different nodes or servers.
- ✔ Network independent of operating systems
  - Compatibility with various operating systems across nodes or servers.
- ✔ Transparency of transactions
  - Users and applications access data without concern for distribution.
- ✔ DBMS unrelated
  - Flexibility to integrate different DBMS for different nodes or servers.

Disadvantages :

- **Costly Software :** Ensuring data transparency and coordination across multiple sites often requires using expensive software in a distributed database system.

- **Large Overhead :** Many operations on multiple sites requires numerous calculations and constant synchronization when database replication is used, causing a lot of processing overhead.

- **Data Integrity :** A possible issue when using database replication is data integrity, which is compromised by updating data at multiple sites.

- **Improper Data Distribution :** Responsiveness to user requests largely depends on proper data distribution. That means responsiveness can be reduced if data is not correctly distributed across multiple sites.

18. Explain about sequential file access and hash index with examples.

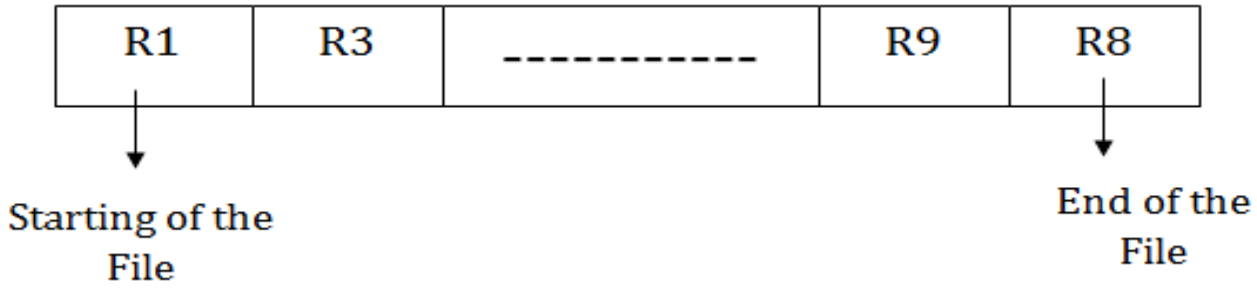
Answer :

Sequential File Organization :

This method is the easiest method for file organization. In this method, files are stored sequentially. This method can be implemented in two ways:

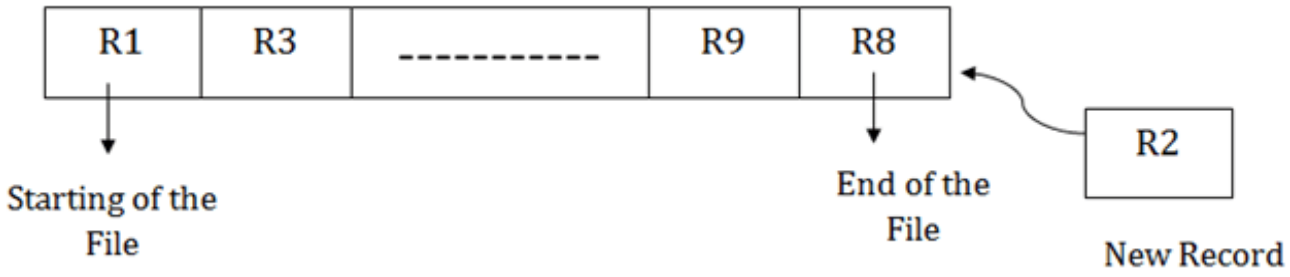
1. Pile File Method:

- ❍ It is a quite simple method. In this method, we store the record in a sequence, i.e., one after another. Here, the record will be inserted in the order in which they are inserted into tables.
- ❍ In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.



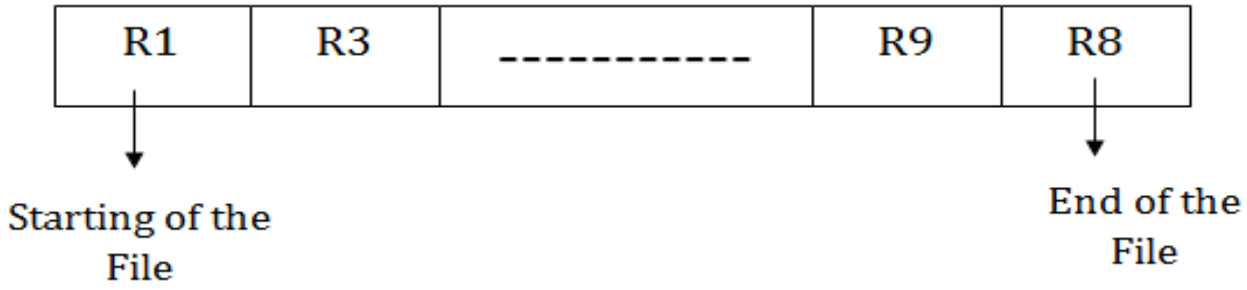
Insertion of the new record:

Suppose we have four records R1, R3 and so on upto R9 and R8 in a sequence. Hence, records are nothing but a row in the table. Suppose we want to insert a new record R2 in the sequence, then it will be placed at the end of the file. Here, records are nothing but a row in any table.



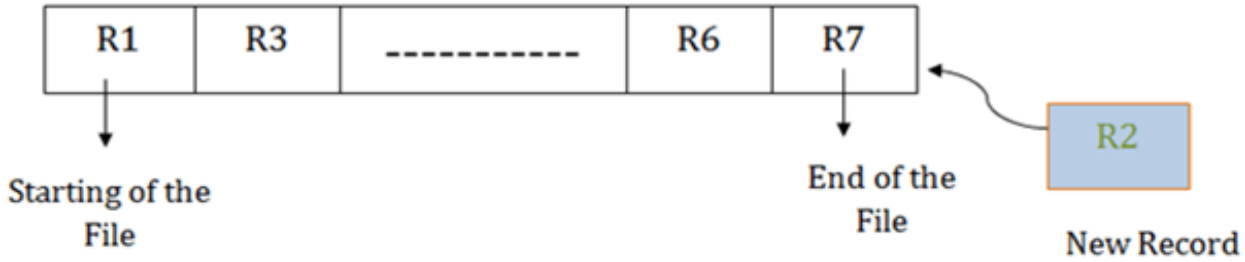
2. Sorted File Method :

- ❍ In this method, the new record is always inserted at the file's end, and then it will sort the sequence in ascending or descending order. Sorting of records is based on any primary key or any other key.
- ❍ In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.



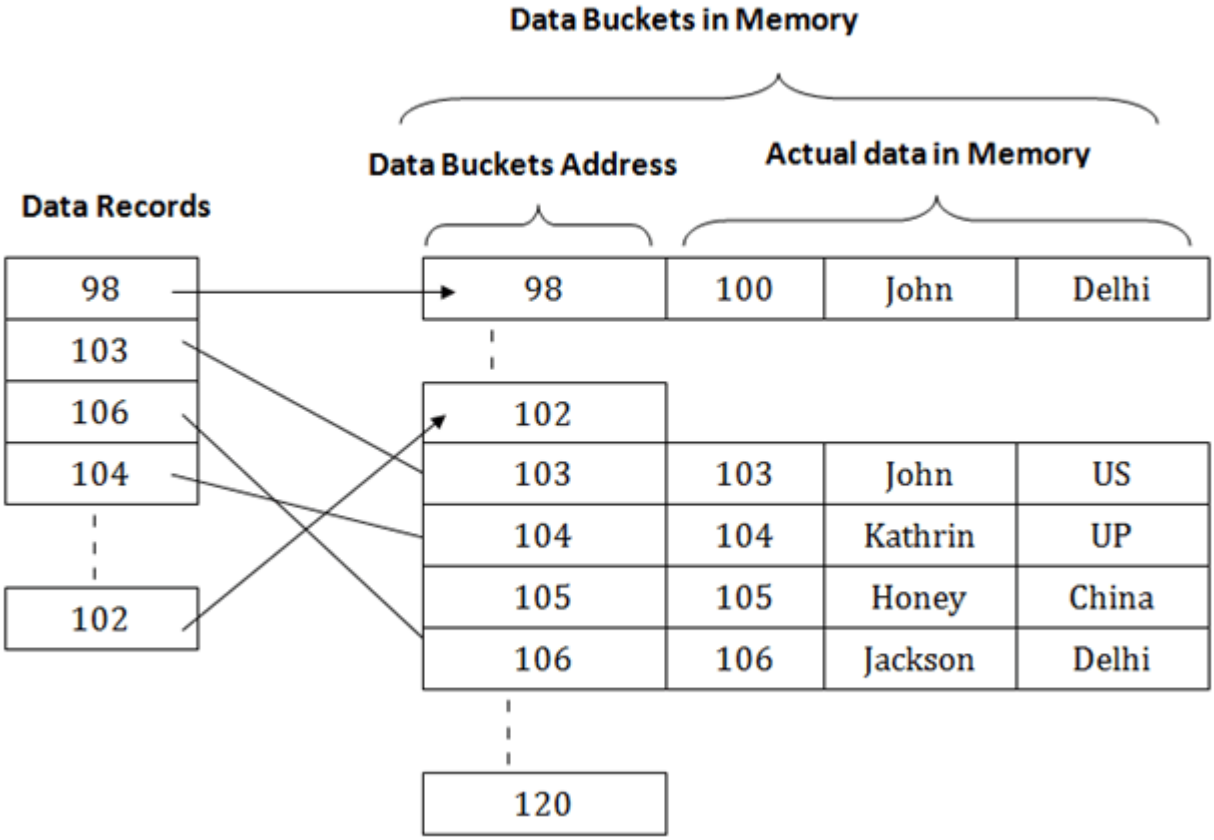
Insertion of the new record:

Suppose there is a preexisting sorted sequence of four records R1, R3 and so on upto R6 and R7. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file, and then it will sort the sequence.

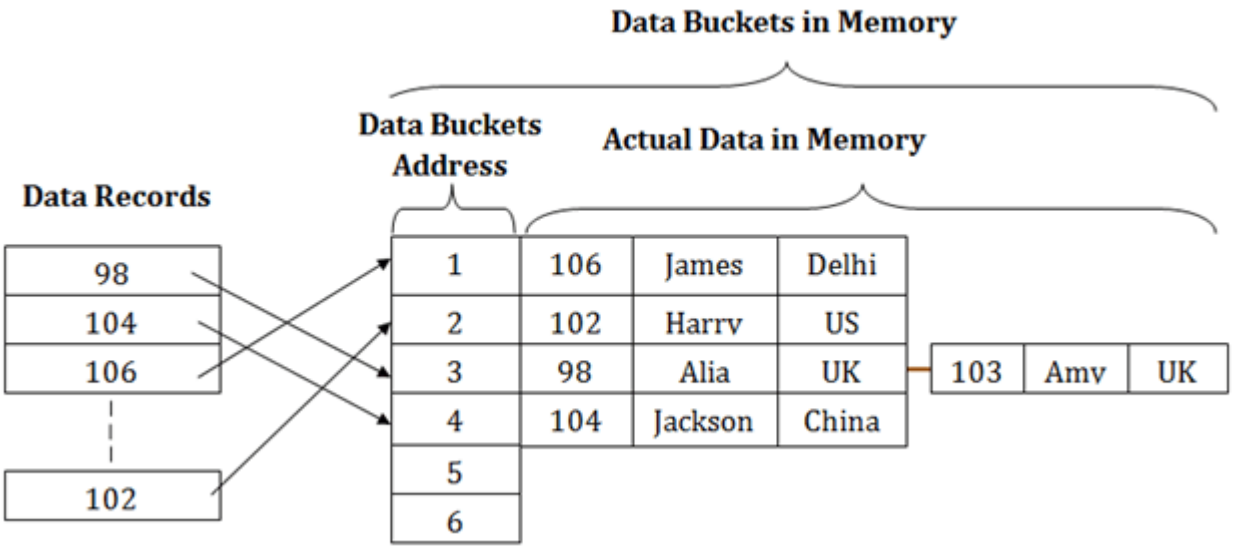


Hashing in DBMS

- Hashing is used in databases to directly locate data records on disk without relying on index structures.
- It uses a hash function to generate the address of data blocks where records are stored.
- Hash functions can be based on any column value, often the primary key.
- Data blocks are also called data buckets and store records.
- Hashing allows for efficient retrieval of data without searching through all index values.
- It provides quick access to data in large databases.



- The hash function used in hashing can be a simple mathematical function like mod, exponential, cos, sin, etc.
- In the given example with mod (5) hash function, the primary keys are processed through the function, resulting in addresses 3, 3, 1, 4, and 2 for the data blocks.
- The records are stored in the corresponding data block addresses based on the hash function output.


Data Buckets in Memory
Data Buckets Address
Actual Data in Memory
Data Records
98
104
106
102
1
2
3
4
5
6
106
James
Delhi
102
Harrv
US
98
Alia
UK
104
Jackson
China
103
Amv
UK

19. Define Referential-integrity constraints. Consider a database that includes the following relations:

salaried-worker (name, office, phone, salary)
hourly-worker (name, hourly-wage)
address (name, street, city)

Suppose that we wish to require that every name that appears in address appear in either salaried- worker or hourly-worker, but not necessarily in both.
a. Propose a syntax for expressing such constraints.
b. Discuss the actions that the system must take to enforce a constraint of this form

Answer :

Referential Integrity

- Referential integrity enforces the relationship between tables in a database.
- It establishes a parent-child relationship between two tables that share a common column definition.
- To implement referential integrity, we define the column in the parent table as the primary key.
- The same column in the child table is then defined as a foreign key, referring to the corresponding entry in the parent table.
- The foreign key ensures that any value in the child table's column must exist as a primary key value in the parent table.
- This constraint helps maintain the consistency and integrity of data between related tables.

SOLUTION :

a. For simplicity, we present a variant of the SQL syntax. As part of the create table expression for address we include

foreign key (name) **references** salaried-worker or hourly-worker

b. To enforce this constraint, whenever a tuple is inserted into the address relation, a lookup on the name value must be made on the salaried-worker relation and (if that lookup failed) on the hourly-worker relation (or vice-versa).

20. What is importance of trigger? Write an SQL trigger to carry out the following action: On delete of an account, for each owner of the account, check if the owner has any remaining accounts, and if she does not, delete her from the depositor relation.

Answer :

create trigger check-delete-trigger **after delete on** account

referencing old row as orow

for each row

delete from depositor

where depositor.customer-name **not in**

(**select** customer-name **from** depositor

where account-number <> orow.account-number)

end