B.E V Sem CSE (Code: 2639/CBCS)
Database Management Systems
PART-A

1. *List the Database Administrator functions.*
   - Schema definition
   - Storage structure and access method definition
   - Schema and physical organization modification
   - Granting of authorization
   - Routine maintenance

2. *Discuss ER Design issues in brief.*
   - Use of entity sets vs attributes
   - Use of entiry sets vs relationship sets
   - Binary Vs n-ary relationship sets
   - Placement of relationship attributes
3. *Discuss SQL sub languages*
   a. DDL
   b. DML
4. *Discuss about group by clause and having clause*

   The GROUP BY Statement in SQL is used to arrange identical data into groups with the help of some functions. i.e if a particular column has same values in different rows then it will arrange these rows in a group.

   HAVING Clause: We can use HAVING clause to place conditions to decide which group will be the part of final result-set. Also we cannot use the aggregate functions like SUM(), COUNT() etc. with WHERE clause. So we have to use HAVING clause if we want to use any of these functions in the conditions.
5. *What is Embedded SQL,*

   Embedded SQL provides a means by which a program can interact with a database server. Under this SQL statements are identified at compile time using preprocessor. The preprocessor submits the SQL statements to the database system for pre-compilation and optimization.

   Embedded SQL statements syntax:

   ```
   EXEC SQL <embedded SQL statement>;
   ```
6. *Define Trigger, write the syntax.*

   Trigger is a statement that is executed automatically when a DML operation(event) is performed on a database table. To design a trigger there are two requirements

   a. Specify when trigger is to be executed (Event)

   b. Specify the actions to be taken when trigger executes.

   Once we create a trigger the database systems responsibility to execute it whenever the specified event occurs.

   ```
   Syntax:          create trigger [trigger_name] [before | after]
                    {insert | update | delete}  on [table_name]
                     [for each row]  [trigger_body]
   ```

7. *Define ACID properties*

   ACID stands for **A**tomicity, **C**onsistency, **I**solation, and **D**urability.

- **Atomicity:** A transaction is a single unit of operation. You either execute it entirely or do not execute it at all. There cannot be partial execution.
- **Consistency:** Once the transaction is executed, it should move from one consistent state to another.
- **Isolation:** Transaction should be executed in isolation from other transactions (no Locks). During concurrent transaction execution, intermediate transaction results from simultaneously executed transactions should not be made available to each other. (Level 0,1,2,3)
- **Durability:** · After successful completion of a transaction, the changes in the database should persist. Even in the case of system failures.

8. *Difference between Primary and Secondary Index.*

- **Primary Index** − Primary index is defined on an ordered data file. The data file is ordered on a **key field**. The key field is generally the primary key of the relation.
- **Secondary Index** − Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

9. *What is stable storage?*

Stable storage is a classification of computer data **storage** technology that guarantees atomicity for any given write operation and allows software to be written that is robust against some hardware and power failures. A form of storage that survives all failures.

10. *Define Recoverability.*

Transaction may not execute completely due to hardware failure, system crash or software issues. In that case, we have to roll back the failed transaction. But some other transaction may also have used values produced by the failed transaction. So we have to roll back those transactions as well.

Recoverable Schedules: Schedules in which transactions commit only after all transactions whose changes they read commit are called recoverable schedules. In other words, if some transaction T_j is reading value updated or written by some other transaction T_i, then the commit of T_j must occur after the commit of T_i.

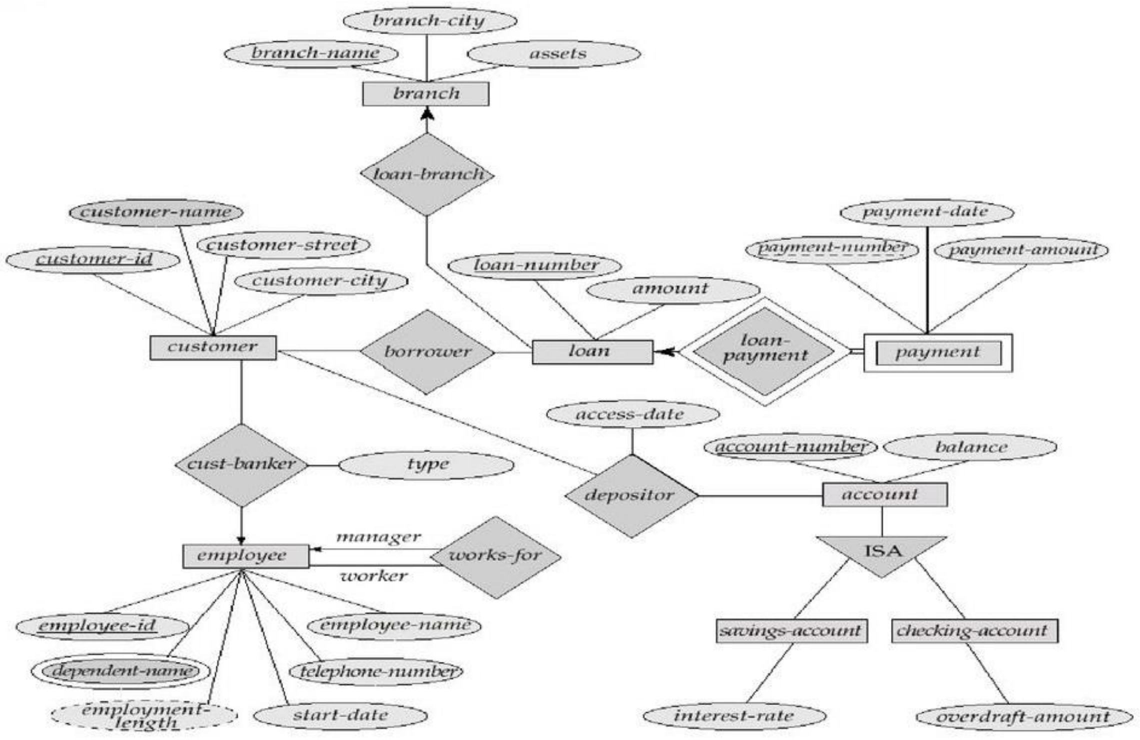
PART-B

11 a) Explain the difference between file processing system and DBMS.

Answer should cover the following points

- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation
- Integrity Problems
- Atomicity problems
- Concurrent access anomalies
- Security problems

11 b) Database design of Banking Enterprise using E-R.



12 a) String operations and aggregate functions in SQL.

String functions
are used to perform an operation on input string and return an output string.
Following are the string functions defined in SQL:

Students can write any four from the following

CONCAT(): This function is used to add two words or strings.
Syntax: SELECT 'Osmania' || ' ' || 'University' FROM dual;
Output: 'Osmania University'
FIND_IN_SET(): This function is used to find a symbol from a set of symbols.
Syntax: SELECT FIND_IN_SET('b', 'a, b, c, d, e, f');
Output: 2
INSTR(): This function is used to find the occurrence of an alphabet.
Syntax: INSTR('Osmania University', 'a');
Output: 4 (the first occurrence of 'e')
Syntax: INSTR('geeks for geeks', 'e', 1, 2);
Output: 3 (the second occurrence of 'e')
LCASE(): This function is used to convert the given string into lower case.
Syntax: LCASE ("OSMANIA University To Learn");
Output: osmania university to learn
LEFT(): This function is used to SELECT a sub string from the left of given size or characters.
Syntax: SELECT LEFT('osmania.ac.in', 5);
Output: osman

LENGTH(): This function is used to find the length of a word.
Syntax: LENGTH('GeeksForGeeks');
Output: 13
LOCATE(): This function is used to find the nth position of the given word in a string.
Syntax: SELECT LOCATE('man', 'osmania', 1);
Output: 3
LOWER(): This function is used to convert the upper case string into lower case.
Syntax: SELECT LOWER('OSMANIA');
Output: osmania
LPAD(): This function is used to make the given string of the given size by adding the given symbol.
Syntax: LPAD('osman', 8, '0');
Output:
000osman
LTRIM(): This function is used to cut the given sub string from the original string.
Syntax: LTRIM('123123osman', '123');
Output: osman
MID(): This function is to find a word from the given position and of the given size.
Syntax: Mid ("osmanforosman", 6, 2);
Output: for
POSITION(): This function is used to find position of the first occurrence of the given alphabet.
Syntax: SELECT POSITION('a' IN 'osmania');
Output: 4
REPEAT(): This function is used to write the given string again and again till the number of times mentioned.
Syntax: SELECT REPEAT('geeks', 2);
Output: geeksgeeks
REPLACE(): This function is used to cut the given string by removing the given sub string.
Syntax: REPLACE('123osmania123', '123');
Output: osmania
REVERSE(): This function is used to reverse a string.
Syntax: SELECT REVERSE('osmania');
Output: ainamso
RIGHT(): This function is used to SELECT a sub string from the right end of the given size.
Syntax: SELECT RIGHT('osmania.org', 4);
Output: '.org'
RPAD(): This function is used to make the given string as long as the given size by adding the given symbol on the right.
Syntax: RPAD('osman', 8, '0');
Output: 'osman000'
STRCMP(): This function is used to compare 2 strings.
If string1 and string2 are the same, the STRCMP function will return 0.
If string1 is smaller than string2, the STRCMP function will return -1.
If string1 is larger than string2, the STRCMP function will return 1.
SUBSTR(): This function is used to find a sub string from the a string from the given position.
Syntax:SUBSTR('osmaniauniversity', 1, 5);
SUBSTRING_INDEX(): This function is used to find a sub string before the given symbol.
Syntax: SELECT SUBSTRING_INDEX('www.osmania.org', '.', 1);
Output: 'www'
TRIM(): This function is used to cut the given symbol from the string.
Syntax: TRIM(LEADING '0' FROM '000123');
Output: 123

Aggregate Functions: sum(), avg(), max(),min(), count()

12 b) Explain Fundamental and Additional relational algebra operations using suitable examples.

Fundamental: SELECT (σ)
PROJECT (π)
UNION (U)
SET DIFFERENCE (-)
CARTESIAN PRODUCT (X)
RENAME (ρ)
ADDITIONAL: SET INTERSECTION
NATURAL JOIN
OUTER JOIN
ASSIGNMENT OPERATION

13 a) Write Dynamic SQL program to retrieve data from database.

```
        import java.sql.*;
                class MysqlCon{
                public static void main(String args[]){
                try{
                Class.forName("com.mysql.jdbc.Driver");
                Connection con=DriverManager.getConnection( "jdbc:mysql://localhost:3306/db","root","root");
                                OR
                 Class.forName ("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection( "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

                Statement stmt=con.createStatement();
                ResultSet rs=stmt.executeQuery("select * from emp");
                while(rs.next())
                        System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
                        con.close();
                }catch(Exception e){ System.out.println(e);}
                }
                }
```

13 b) Features of good database design.

Relational database design requires that we find a “good”collection of relation schemas.
A bad design may lead to

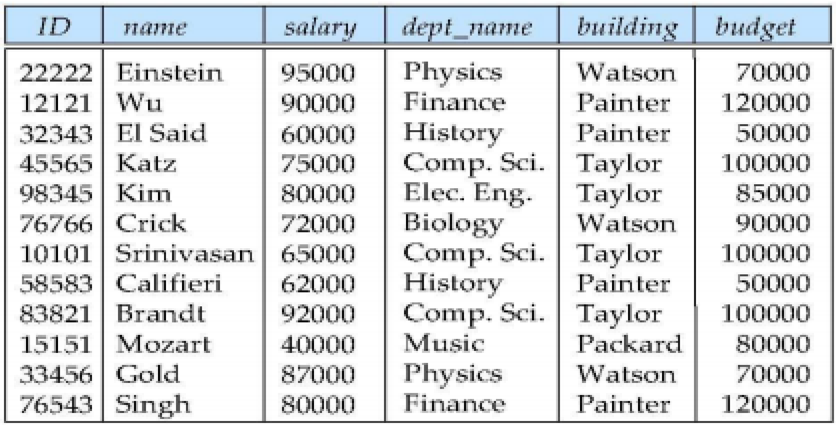
- Repetition of Information.
- Inability to represent certain information.

Design Goals:

- Avoid redundant data
- Ensure that relationships among attributes are represented
- Facilitate the checking of updates for violation of database integrity constraints.

Combine Schemas?

Suppose we combine instructor and department into inst_dept (No connection to relationship set inst_dept)
Result is possible repetition of information

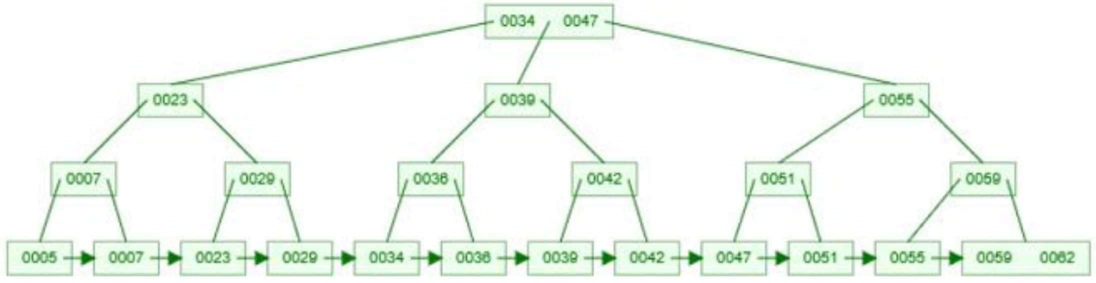


ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

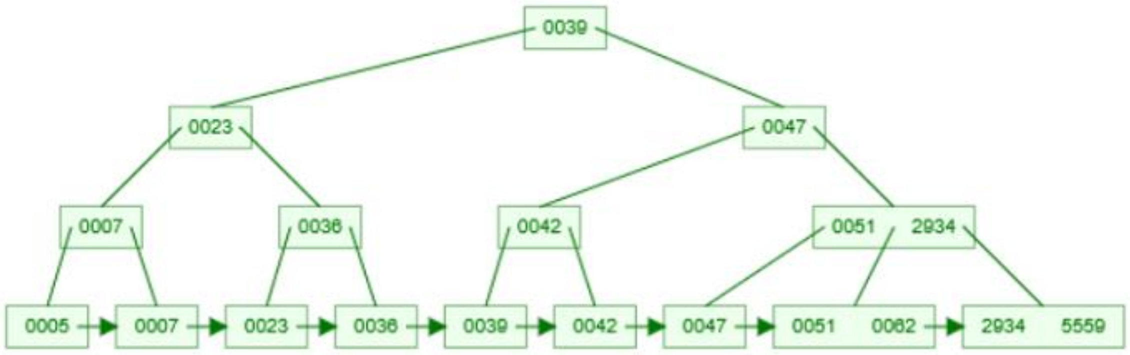
A Combined Schema Without Repetition
Consider combining relations sec_class(sec_id, building, room_number) and section(course_id, sec_id, semester, year) into one relation section(course_id, sec_id, semester, year,building, room_number) No repetition of data

14 a) Insert the keys into B+ tree for the order n=3

5,7,23,36,2934,39,42,47,51,5559,62



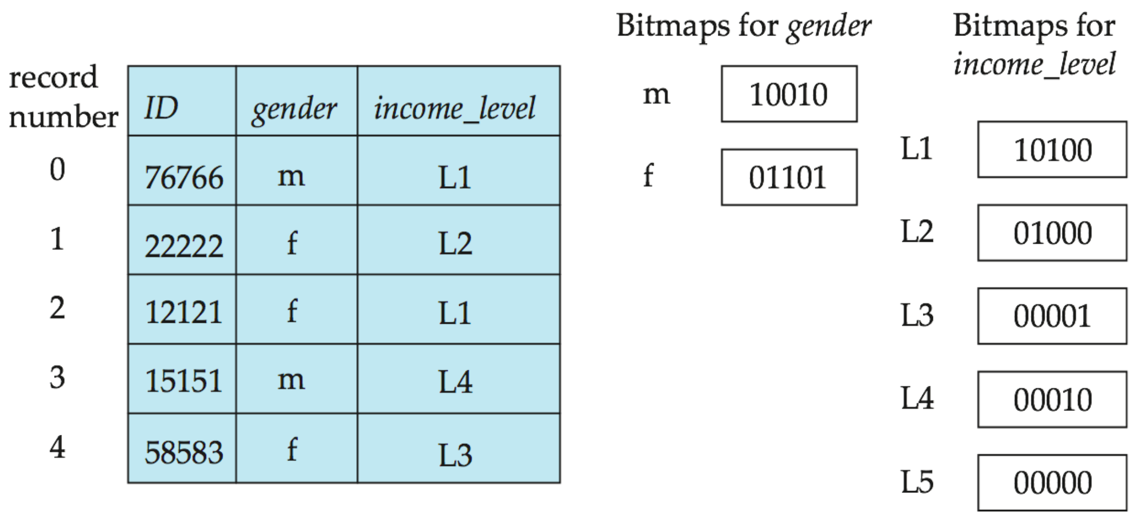
OR



14b) Explain Bit map indices? How they are useful in multi key access.

Bitmap index on an attribute has a bitmap for each value of the attribute

- Bitmap has as many bits as records
- In a bitmap for value v, the bit for a record is 1 if the record has the value v for the attribute, and is 0 otherwise



record number	ID	gender	income_level
0	76766	m	L1
1	22222	f	L2
2	12121	f	L1
3	15151	m	L4
4	58583	f	L3

Bitmaps for *gender*

m	10010
f	01101

Bitmaps for *income_level*

L1	10100
L2	01000
L3	00001
L4	00010
L5	00000

Bitmap indices are useful for queries on multiple attributes not particularly useful for single attribute queries

Queries are answered using bitmap operations use

Intersection (and)
Union (or)
Complementation (not)

Each operation takes two bitmaps of the same size and applies the operation on corresponding bits to get the result bitmap

E.g. 100110 AND 110011 = 100010
100110 OR 110011 = 110111
NOT 100110 = 011001
Males with income level L1: 10010 AND 10100 = 10000
Can then retrieve required tuples.

Counting number of matching tuples is even faster

15 a) What is currency control? Explain Lock based protocols.

Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each another. Concurrent access is quite easy if all users are just reading data. There is no way they can interfere with one another. Though for any practical database, would have a mix of reading and WRITE operations and hence the concurrency is a challenge.

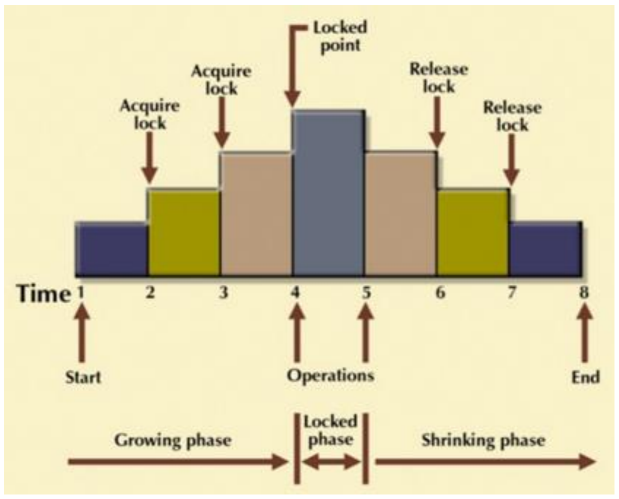
Concurrency control is used to address such conflicts which mostly occur with a multi-user system. It helps you to make sure that database transactions are performed concurrently without violating the data integrity of respective databases.

Concurrency control is a most important element for the proper functioning of a system where two or multiple database transactions that require access to the same data, are executed simultaneously.

Lock based protocols:

Two Phase Locking (2PL) Protocol

Two-Phase locking protocol which is also known as a 2PL protocol. It is also called P2L. In this type of locking protocol, the transaction should acquire a lock after it releases one of its locks.



This locking protocol divides the execution phase of a transaction into three different parts.

- In the first phase, when the transaction begins to execute, it requires permission for the locks it needs.
- The second part is where the transaction obtains all the locks. When a transaction releases its first lock, the third phase starts.
- In this third phase, the transaction cannot demand any new locks. Instead, it only releases the acquired locks.

The Two-Phase Locking protocol allows each transaction to make a lock or unlock request in two steps:

- Growing Phase: In this phase transaction may obtain locks but may not release any locks.
- Shrinking Phase: In this phase, a transaction may release locks but not obtain any new lock

It is true that the 2PL protocol offers serializability. However, it does not ensure that deadlocks do not happen.

In the above-given diagram, you can see that local and global deadlock detectors are searching for deadlocks and solve them with resuming transactions to their initial states.

Strict Two-Phase Locking Method
Strict-Two phase locking system is almost similar to 2PL. The only difference is that Strict-2PL never releases a lock after using it. It holds all the locks until the commit point and releases all the locks at one go when the process is over.

Conservative Two-Phase Locking Method
15 b) Explain ARIES algorithm.

Algorithm for Recovery and Isolation Exploiting Semantics (ARIES) is based on the Write Ahead Log (WAL) protocol. Every update operation writes a log record which is one of the following :

1. Undo-only log record:
2. Redo-only log record:
3. Undo-redo log record:
   In it, every log record is assigned a unique and monotonically increasing log sequence number (LSN). Every data page has a page LSN field that is set to the LSN of the log record corresponding to the last update on the page. WAL requires that the log record corresponding to an update make it to stable storage before the data page corresponding to that update is written to disk. For performance reasons, each log write is not immediately forced to disk. A log tail is maintained in main memory to buffer log writes. The log tail is flushed to disk when it gets full. A transaction cannot be declared committed until the commit log record makes it to disk.

Once in a while the recovery subsystem writes a checkpoint record to the log. The checkpoint record contains the transaction table and the dirty page table. A master log record is maintained separately, in stable storage, to store the LSN of the latest checkpoint record that made it to disk. On restart, the recovery subsystem reads the master log record to find the checkpoint's LSN, reads the checkpoint record, and starts recovery from there on.

The recovery process actually consists of 3 phases:

1. Analysis:
   The recovery subsystem determines the earliest log record from which the next pass must start. It also scans the log forward from the checkpoint record to construct a snapshot of what the system looked like at the instant of the crash.
2. Redo:
   Starting at the earliest LSN, the log is read forward and each update redone.
3. Undo:
   The log is scanned backward and updates corresponding to loser transactions are undone.

16 a) Explain Log based recovery?

Log based Recovery in DBMS

To achieve our goal of atomicity, user must first output to stable storage information describing the modifications, without modifying the database itself. This information can help us ensure that all modifications performed by committed transactions are reflected in the database. This information can also help us ensure that no modifications made by an aborted transaction persist in the database.

Log and log records –
The log is a sequence of log records, recording all the update activities in the database. In a stable storage, logs for each transaction are maintained. Any operation which is performed on the database is recorded is on the log.
An update log record represented as: <Ti, Xj, V1, V2> has these fields:

1. Transaction identifier: Unique Identifier of the transaction that performed the write operation.
2. Data item: Unique identifier of the data item written.
3. Old value: Value of data item prior to write.
4. New value: Value of data item after write operation.

Other type of log records are:

1. <Ti start>: It contains information about when a transaction Ti starts.
2. <Ti commit>: It contains information about when a transaction Ti commits.
3. <Ti abort>: It contains information about when a transaction Ti aborts.

Undo and Redo Operations –
Because all database modifications must be preceded by creation of log record, the system has available both the old

value prior to modification of data item and new value that is to be written for data item. This allows system to perform redo and undo operations as appropriate:

1. **Undo:** using a log record sets the data item specified in log record to old value.
2. **Redo:** using a log record sets the data item specified in log record to new value.

The database can be modified using two approaches –

1. **Deferred Modification Technique:** If the transaction does not modify the database until it has partially committed, it is said to use deferred modification technique.
2. **Immediate Modification Technique:** If database modification occur while transaction is still active, it is said to use immediate modification technique.

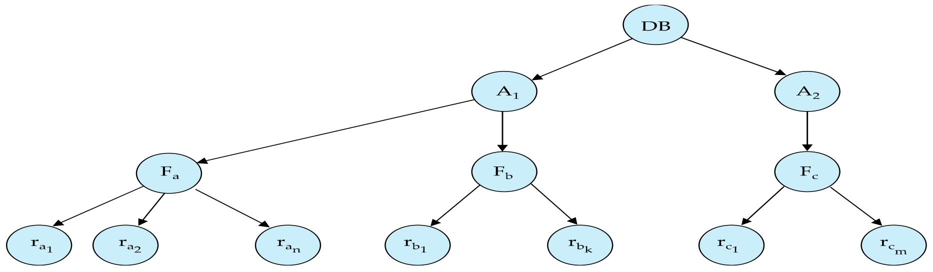
Recovery using Log records –

After a system crash has occurred, the system consults the log to determine which transactions need to be redone and which need to be undone.

1. Transaction Ti needs to be undone if the log contains the record <Ti start> but does not contain either the record <Ti commit> or the record <Ti abort>.
2. Transaction Ti needs to be redone if log contains record <Ti start> and either the record <Ti commit> or the record <Ti abort>.

16 b) Explain Multiple granularity protocol

- Allow data items to be of various sizes and define a hierarchy of data granularities, where the small granularities are nested within larger ones
- Can be represented graphically as a tree.
- When a transaction locks a node in the tree explicitly, it implicitly locks all the node's descendents in the same mode.
- Granularity of locking (level in tree where locking is done):
    - fine granularity (lower in tree): high concurrency, high locking overhead
    - coarse granularity (higher in tree): low locking overhead, low concurrency
- In addition to S and X lock modes, there are three additional lock modes with multiple granularity:
    - intention-shared (IS): indicates explicit locking at a lower level of the tree but only with shared locks.
    - intention-exclusive (IX): indicates explicit locking at a lower level with exclusive or shared locks
    - shared and intention-exclusive (SIX): the subtree rooted by that node is locked explicitly in shared mode and explicit locking is being done at a lower level with exclusive-mode locks.
- intention locks allow a higher level node to be locked in S or X mode without having to check all descendent nodes.



The levels, starting from the coarsest (top) level are

- database
- area
- file
- record

- **Transaction *T_i* can lock a node *Q*, using the following rules:**
    - **The lock compatibility matrix must be observed.**

    - The root of the tree must be locked first, and may be locked in any mode.
    - A node *Q* can be locked by T_i in S or IS mode only if the parent of *Q* is currently locked by T_i in either IX or IS mode.
    - A node *Q* can be locked by T_i in X, SIX, or IX mode only if the parent of *Q* is currently locked by T_i in either IX or SIX mode.
    - T_i can lock a node only if it has not previously unlocked any node (that is, T_i is two-phase).
    - T_i can unlock a node *Q* only if none of the children of *Q* are currently locked by T_i.
- Observe that locks are acquired in root-to-leaf order, whereas they are released in leaf-to-root order.
- Lock granularity escalation: in case there are too many locks at a particular level, switch to higher granularity S or X lock

17 a) Recursive Queries

Recursive queries are used to query hierarchical data. The SQL standard defines a special syntax for common table expressions to enable recursive processing.

```
   with recursive empl (employee_name, manager_name ) as (
         select employee_name, manager_name
         from   manager
     union
         select manager.employee_name, empl.manager_name
         from   manager, empl
         where manager.manager_name = empl.employe_name)
  select *
  from   empl
```

17 b) Storage Structure

Storage structure is the memory structure in the system. It is mainly divided into two categories :

Volatile Memory: It is that the quite hardware that stores information quickly. it's additionally referred as temporary memory. The information within the volatile memory is hold on solely till the ability is provided to the system, once the system is turned off the information gift within the volatile memory is deleted mechanically. RAM (Random Access Memory) and Cache Memory are the common example of the volatile memory. It's quite quick and economical in nature and may be accessed apace.
Non-Volatile Memory: It is the type of memory in which data or information remains keep within the memory albeit power is completed. ROM (Read Only Memory) is the most common example of non-volatile memory. it's not that a lot of economical and quick in nature as compare to volatile memory however stores information for the longer amount. Non-volatile memory is slow concerning accessing. All such information that must be hold on for good or for a extended amount is hold on in non-volatile memory. Non-volatile memory has a huge impact on a system's storage capacity.

17 c) Thomas write rule
Thomas Write Rule does not enforce *Conflict Serializablity* but rejects fewer Write Operations by modifying the check Operations for W_item(X)

1. If **R_TS(X) > TS(T)**, then abort and rollback T and reject the operation.
2. If **W_TS(X) > TS(T)**, then don't execute the Write Operation and continue processing. This is a case of *Outdated or Obsolete Writes*. Remember, outdated writes are ignored in Thomas Write Rule but a Transaction following Basic TO protocol will abort such a Transaction.
3. If neither the condition in 1 or 2 occurs, then and only then execute the W_item(X) operation of T and set W_TS(X) to TS(T)