CS/IT 315

Hall Ticket Number:

III/IV B.Tech (Supplementary) DEGREE EXAMINATION

November, 2016 **Common for CSE & IT**

Fifth Semester **Database Management Systems**

Time: Three Hours **Maximum :** 60 Marks

Answer Question No.1 compulsorily. (1X12 = 12 Marks)

Answer ONE question from each unit. (4X12=48 Marks)

1. Define the following (1X12=12 Marks)

a) Database
b) Key
c) Weak Entity
d) DCL commands
e) Trigger
f) View
g) ALTER command syntax
h) Atomicity
i) Shared lock
j) Normalization
k) Recovery
l) Save point

UNIT I

2. a) What is DBMS? What are the advantages of DBMS? `(7M)
   b) Explain the concept of data independence (5M)

(OR)

3. a) Explain in detail about database architecture. (6M)
   b) Define E-R Diagram. Construct E-R Model for university database and explain in detail. (6M)

UNIT II

4. a) Explain different types of integrity constraints with examples (6M)
   b) Explain in detail about Database languages with examples. (6M)

(OR)

5. a) Explain in detail about relational algebra operators with examples. (4M)
   b) Consider the following relational schema : (8M)
   WORKS (Emp-name, Comp-name, Salary, Ph-Num)
   LIVES IN (Emp-name, Street, City)
   LOCATED-IN (Comp-name, City)
   MANAGER - OF (Mgr-name, Emp-name)
   Write the SQL Queries for the following:
   i) Find the names of the employees, who live and work in the same city.
   ii) Find the names of the employees, who do not work for the company "SBM".
   iii) Find the names of the employees and their salary, whose manager is "Fleming".

iv) Delete all the employees, who work for the company "SBM".

UNIT III

6. a)Explain about the B+ tree, its operations and its structure in detail with an example. (6M)
   b) What is Normalization? Explain in detail about 1NF , 2NF , 3NF with examples. (6M)

(OR)

7. a) Explain about sorted and unsorted file records in detail. (6M)
   b) What are the problems caused by redundancy? Explain about Multivalued and join dependencies (6M)

UNIT IV

8. a) What is a transaction? Explain in detail about ACID Properties. (6M)
   b) Explain about 2-Phase Locking Protocol (6M)

(OR)

9. a) What is shadow paging? Explain recovery technique based on immediate update. (6M)
   b) What is Recoverability? Explain about mandatory access control. (6M)

CS/IT 315

Hall Ticket Number:

III/IV B.Tech (Supplementary) DEGREE EXAMINATION

November, 2016 **Common for CSE & IT**

Fifth Semester **Database Management Systems**

Time: Three Hours **Maximum :** 60 Marks

Answer Question No.1 compulsorily. (1X12 = 12 Marks)

Answer ONE question from each unit. (4X12=48 Marks)

1. Define the following (1X12=12 Marks)

m) Database

A database is a collection of related data.1 By data,we mean known facts that can be recorded and that have implicit meaning.

n) Key

A key is an attribute or a set of attributes in a relation that identifies a tuple in a relation. The keys are defined in a table to access or sequence the stored data quickly and smoothly.They are also used to create relationship between different tables.

o) Weak Entity

The entity set which does not have sufficient attributes to form a primary key is called as Weak entity set.

p) DCL commands

Data Control Language(DCL) is used to control privilege in Database.
DCL defines two commands,
Grant : Gives user access privileges to database.
Revoke : Take back permissions from user.

q) Trigger

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events:

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

r) View

A view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

s) ALTER command syntax

```
ALTER TABLE table_name ADD column_name datatype
```

t) Atomicity

In database systems, atomicity is one of the ACID transaction properties. A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all.

u) Shared lock

Shared locks exist when two transactions are granted read access. One transaction gets the shared lock on data and when the second transaction requests the same data it is also given a shared lock. Both transactions are in a read-only mode, updating the data is not allowed until the shared lock is released.

v) Normalization

Normalization is the process of efficiently organizing data in a database with two goals in mind
First goal: eliminate redundant data for example, storing the same data in more than one table.
Second Goal: ensure data dependencies make sense for example, only storing related data in a table

w) Recovery

Data recovery is the process of restoring data that has been lost, accidentally deleted, corrupted or made inaccessible for any reason.

x) Save point

A savepoint is a way of implementing subtransactions (also known as nested transactions) within a relational database management system by indicating a point within a transaction that can be "rolled back to" without affecting any work done in the transaction before the savepoint was created

UNIT I

10. a) **What is DBMS? What are the advantages of DBMS?** **(7M)**

DBMS **Definition of DBMS---2M**

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

Advantages of DBMS **Any 5 advantages-----5M**

- **Controlling Data Redundancy**

  In non-database systems each application program has its own private files. In this case, the duplicated copies of the same data is created in many places. In DBMS, all data of an organization is integrated into a single database file. The data is recorded in only one place in the database and it is not duplicated.

- **Sharing of Data**

  In DBMS, data can be shared by authorized users of the organization. The database administrator manages the data and gives rights to users to access the data. Many users can be authorized to access the same piece of information simultaneously. The remote users can also share same data. Similarly, the data of same database can be shared between different application programs.

- **Data Consistency**

  By controlling the data redundancy, the data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value is immediately available to all users. If the DBMS has controlled redundancy, the database system enforces consistency.

- **Integration of Data**

  In Database management system, data in database is stored in tables. A single database contains multiple tables and relationships can be created between tables (or associated data entities). This makes easy to retrieve and update data.

- **Integration Constraints**

  Integrity constraints or consistency rules can be applied to database so that the correct data can be entered into database. The constraints may be applied to data item within a single record or the may be applied to relationships between records.

- **Data Security**

  Form is very important object of DBMS. You can create forms very easily and quickly in DBMS. Once a form is created, it can be used many times and it can be modified very easily. The created forms are also saved along with database and behave like a software component. A form provides very easy way (user-friendly) to enter data into database, edit data and display data from database. The non-technical users can also perform various operations on database through forms without going into technical details of a database.

- **Control Over Concurrency**

In a computer file-based system, if two users are allowed to access data simultaneously, it is possible that they will interfere with each other. For example, if both users attempt to perform update operation on the same record, then one may overwrite the values recorded by the other. Most database management systems have sub-systems to control the concurrency so that transactions are always recorded with accuracy.

- **Backup and Recovery Procedures**

In a computer file-based system, the user creates the backup of data regularly to protect the valuable data from damage due to failures to the computer system or application program. It is very time consuming method, if amount of data is large. Most of the DBMSs provide the 'backup and recovery' sub-systems that automatically create the backup of data and restore data if required.

b) Explain the concept of data independence (5M)

The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. We can define two types of data independence:

1. **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).
2. **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema.

(OR)

11. a) Explain in detail about database architecture. (6M)

Writing any relevant points can be considered--4M

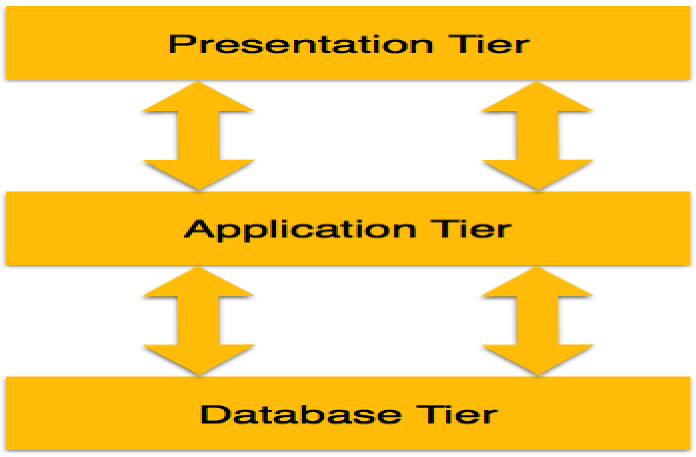
The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-tier Architecture

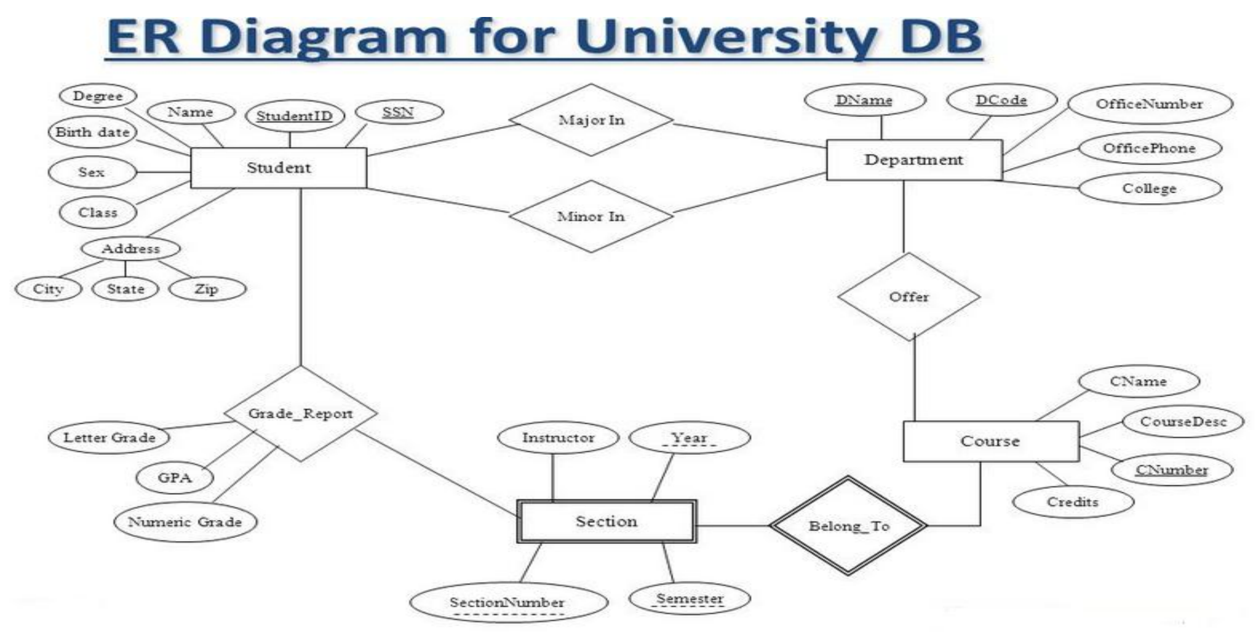
A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.



- **Database (Data) Tier** − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

b) Define E-R Diagram. Construct E-R Model for university database and explain in detail.(6M).

- An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure



UNIT II

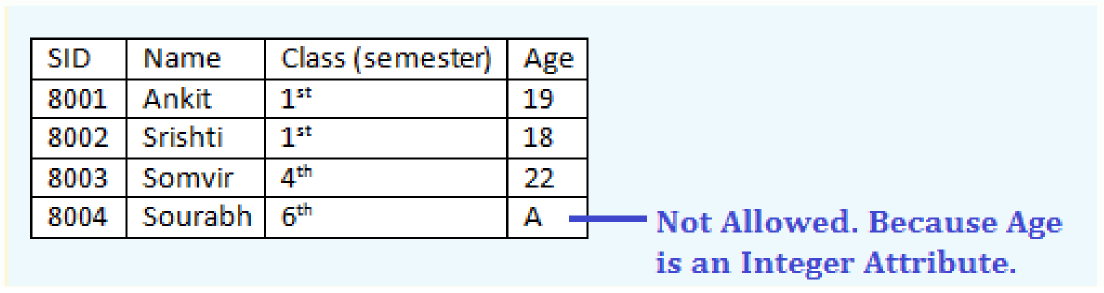
12. a) Explain different types of integrity constraints with examples (6M)

[Examples --2 M]
[Any 4 Constraints explanation ---4M]

- **Domain Constraints**
- **Key Constraints**
- **Single Value Constraints**
- **Entity Integrity Constraint**
- **Referential Integrity Constraint**

Domain Constraints

Domain Constraints specifies that what set of values an attribute can take. Value of each attribute X must be an atomic value from the domain of X.
The data type associated with domains include integer, character, string, date, time, currency etc. An attribute value must be available in the corresponding domain. Consider the example below **[Any relevant example can be considered]**



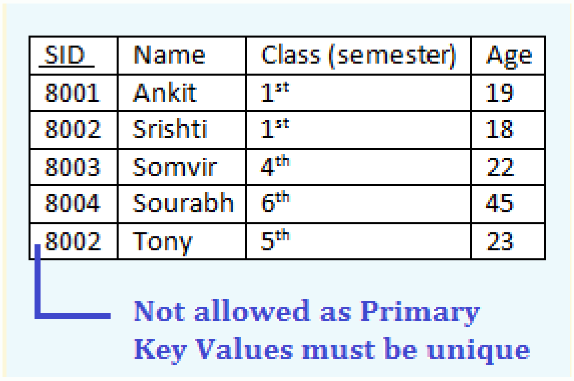
SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	1st	18
8003	Somvir	4th	22
8004	Sourabh	6th	A

Not Allowed. Because Age is an Integer Attribute.

Key Constraints

Keys are attributes or sets of attributes that uniquely identify an entity within its entity set. An Entity set E can have multiple keys out of which one key will be designated as the primary key. Primary Key must have unique and not null values in the relational table. In an subclass hierarchy, only the root entity set has a key or primary key and that primary key must serve as the key for all entities in the hierarchy.

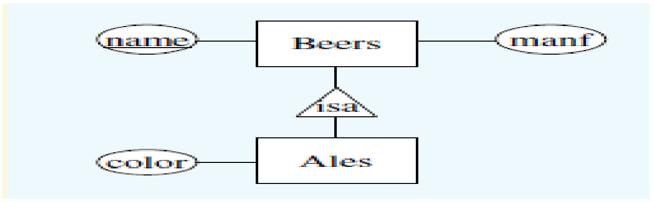
[Any relevant example can be considered]



SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	1st	18
8003	Somvir	4th	22
8004	Sourabh	6th	45
8002	Tony	5th	23

Not allowed as Primary Key Values must be unique

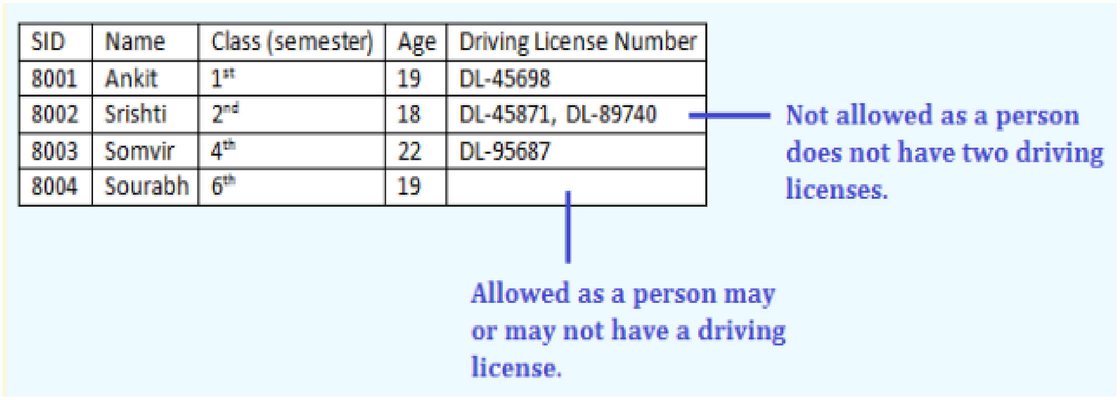
Example of Key Constraints in an subclass hierarchy –



Single Value Constraints

Single value constraints refers that each attribute of an entity set has a single value. If the value of an attribute is missing in a tuple, then we cal fill it with a "null" value. The null value for a attribute will specify that either the value is not known or the value is not applicable. Consider the below example

[Any relevant example can be considered]



SID	Name	Class (semester)	Age	Driving License Number
8001	Ankit	1st	19	DL-45698
8002	Srishti	2nd	18	DL-45871, DL-89740
8003	Somvir	4th	22	DL-95687
8004	Sourabh	6th	19	

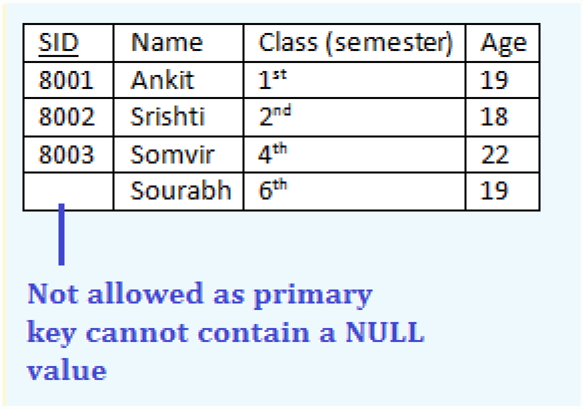
Not allowed as a person does not have two driving licenses.

Allowed as a person may or may not have a driving license.

Entity Integrity Constraint

The Integrity Rule 1 is also called Entity Integrity Rule or Constraint. This rule states that no attribute of primary key will contain a null value. If a relation have a null value in the primary key attribute, then uniqueness property of the primary key cannot be maintained. Consider the example below

[Any relevant example can be considered]



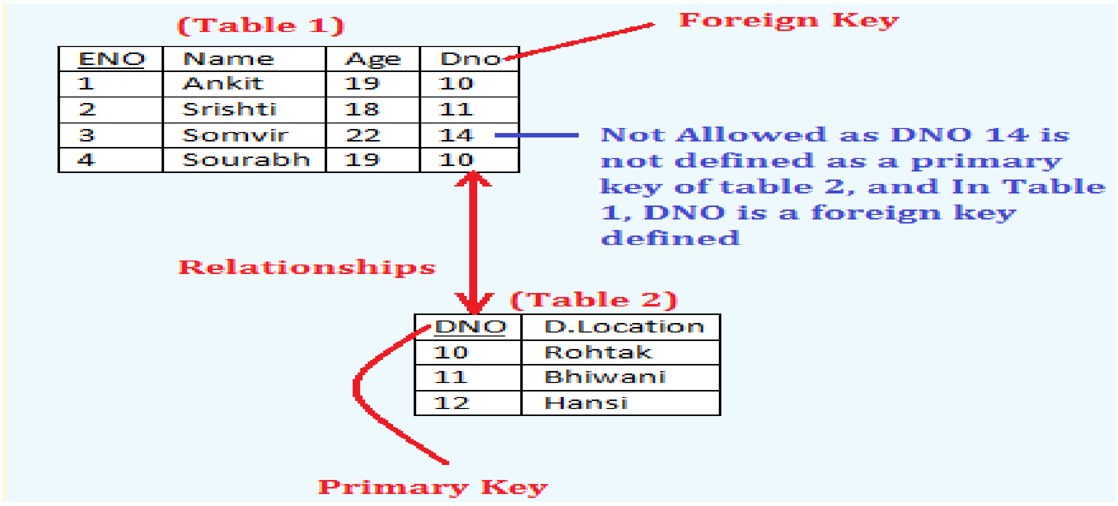
SID	Name	Class (semester)	Age
8001	Ankit	1st	19
8002	Srishti	2nd	18
8003	Somvir	4th	22
	Sourabh	6th	19

Not allowed as primary key cannot contain a NULL value

[Any relevant example can be considered]

Referential Integrity Constraint

The integrity Rule 2 is also called the Referential Integrity Constraints. This rule states that if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2. For example,



(Table 1)

ENO	Name	Age	Dno
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	14
4	Sourabh	19	10

(Table 2)

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

Foreign Key
Not Allowed as DNO 14 is not defined as a primary key of table 2, and In Table 1, DNO is a foreign key defined
Relationships
Primary Key

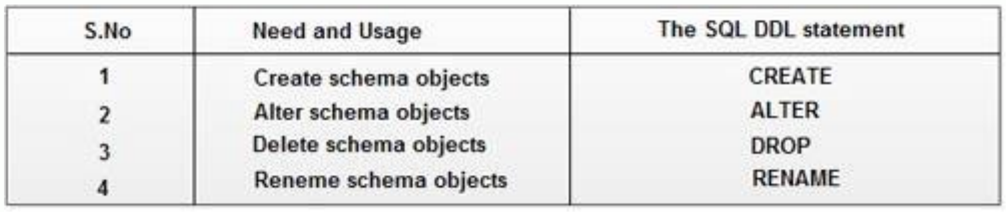
b) Explain in detail about Database languages with examples. (6M)

Data Definition Language (DDL) **[DDL--2M]**

It is a language that allows the users to define data and their relationship to other types of data. It is mainly used to create files, databases, data dictionary and tables within databases.

It is also used to specify the structure of each table, set of associated values with each attribute, integrity constraints, security and authorization information for each table and physical storage structure of each table on disk.

The following table gives an overview about usage of DDL statements in SQL

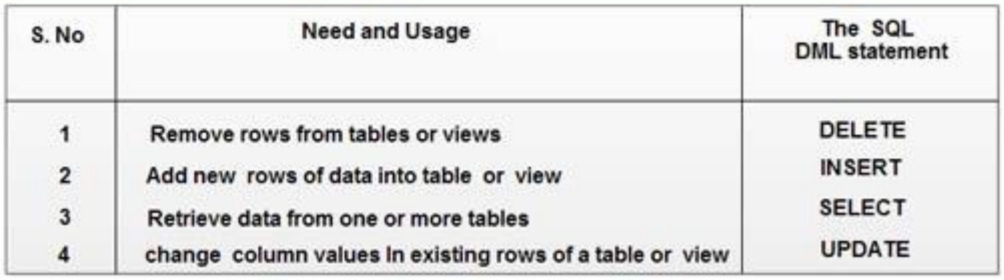


S.No	Need and Usage	The SQL DDL statement
1	Create schema objects	CREATE
2	Alter schema objects	ALTER
3	Delete schema objects	DROP
4	Reneme schema objects	RENAME

Data Manipulation Language (DML) **[DML--2M]**

It is a language that provides a set of operations to support the basic data manipulation operations on the data held in the databases. It allows users to insert, update, delete and retrieve data from the database. The part of DML that involves data retrieval is called a query language.

The following table gives an overview about the usage of DML statements in SQL:

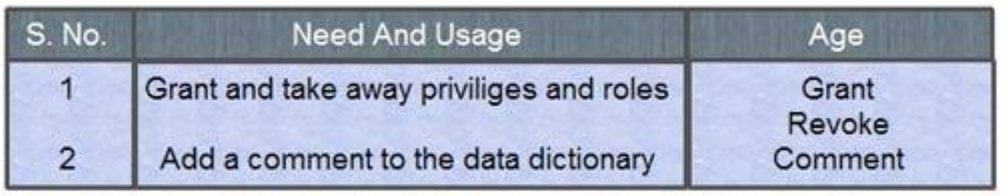


S. No	Need and Usage	The SQL DML statement
1	Remove rows from tables or views	DELETE
2	Add new rows of data into table or view	INSERT
3	Retrieve data from one or more tables	SELECT
4	change column values In existing rows of a table or view	UPDATE

Data Control Language (DCL) **[DCL--2M]**

DCL statements control access to data and the database using statements such as GRANT and REVOKE. A privilege can either be granted to a User with the help of GRANT statement. The privileges assigned can be SELECT, ALTER, DELETE, EXECUTE, INSERT, INDEX etc. In addition to granting of privileges, you can also revoke (taken back) it by using REVOKE command.

The following table gives an overview about the usage of DCL statements in SQL:



S. No.	Need And Usage	Age
1	Grant and take away priviliges and roles	Grant Revoke
2	Add a comment to the data dictionary	Comment

(OR)

13. a) Explain in detail about relational algebra operators with examples. (4M)

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows −

- Select
- Project
- Union
- Set different
- Cartesian product
- Rename

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation − σp(r)

Where σ stands for selection predicate and r stands for relation. p is prepositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like − =, ≠, ≥, < , >, ≤.

For example **[Any relevant example]**

```
σsubject = "database"(Books)
σsubject = "database" and price = "450"(Books)
```

Project Operation (∏)

It projects column(s) that satisfy a given predicate.
Notation − ∏A1, A2, An (r)
Where A1, A2 , An are attribute names of relation r.
Duplicate rows are automatically eliminated, as relation is a set.

For example −

```
∏subject, author (Books)
```

Selects and projects columns named as subject and author from the relation Books.

Union Operation (∪)

It performs binary union between two given relations and is defined as −

```
r ∪ s = { t | t ∈ r or t ∈ s}
```

Notation − r U s

Where r and s are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

```
∏ author (Books) ∪ ∏ author (Articles)
```

Output − Projects the names of the authors who have either written a book or an article or both.

Set Difference (−)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation − (r − s)

Finds all the tuples that are present in r but not in s.

```
∏ author (Books) − ∏ author (Articles)
```

Output − Provides the name of authors who have written books but not articles.

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** *ρ*.

Notation − ρ x (E)

Where the result of expression E is saved with name of x.

5 b) **Consider the following relational schema :** **(8M)**

WORKS (Emp-name, Comp-name, Salary, Ph-Num)
LIVES IN (Emp-name, Street, City)
LOCATED-IN (Comp-name, City)
MANAGER - OF (Mgr-name, Emp-name)

Write the SQL Queries for the following:

v) Find the names of the employees, who live and work in the same city.
select emp-name from works, Livesin,locatedin where works.emp-name=livesin.emp-name and works.compname=locatedin.compname and locatedin.city=livesin.city

vi) Find the names of the employees, who do not work for the company "SBM".
select emp-name from works where comp-name!="SBM";

vii) Find the names of the employees and their salary, whose manager is "Fleming".
select emp-name,salary from works,livein, managerof where works.emp-name=livesin.emp-name and livesin.empname=managerof.emp-name and mgr-name="Fleming";

viii) Delete all the employees, who work for the company "SBM".
delete emp-name from works where comp-name="SBM"

UNIT III

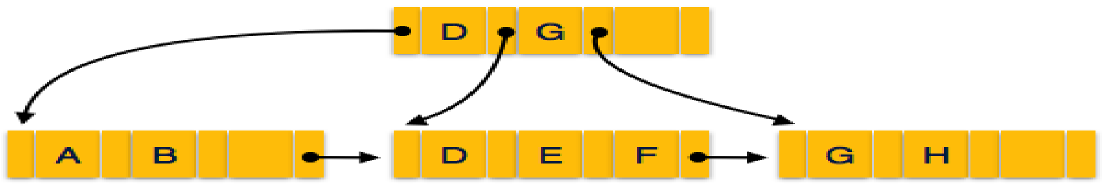
14. a)Explain about the B+ tree, its operations and its structure in detail with an example. (6M)

B$^+$ Tree

A B$^+$ tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B$^+$ tree denote actual data pointers. B$^+$ tree ensures that all leaf nodes remain at the same height, thus balanced. Additionally, the leaf nodes are linked using a link list; therefore, a B$^+$ tree can support random access as well as sequential access.

Structure of B$^+$ Tree

Every leaf node is at equal distance from the root node. A B$^+$ tree is of the order **n** where **n** is fixed for every B$^+$ tree.



Internal nodes –

- Internal (non-leaf) nodes contain at least ⌈n/2⌉ pointers, except the root node.
- At most, an internal node can contain **n** pointers.

Leaf nodes

- Leaf nodes contain at least ⌈n/2⌉ record pointers and ⌈n/2⌉ key values.
- At most, a leaf node can contain **n** record pointers and **n** key values.
- Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.

B$^+$ Tree Insertion

- B^+ trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows −
  - Split node into two parts.
  - Partition at **$i = \lfloor (m+1)_{/2} \rfloor$.**
  - First **i** entries are stored in one node.
  - Rest of the entries (i+1 onwards) are moved to a new node.
  - ***i^{th}*** key is duplicated at the parent of the leaf.
- If a non-leaf node overflows −
  - Split node into two parts.
  - Partition the node at **$i = \lceil (m+1)_{/2} \rceil$**.
  - Entries up to **i** are kept in one node.
  - Rest of the entries are moved to a new node.

B^+ Tree Deletion

- B^+ tree entries are deleted at the leaf nodes.
- The target entry is searched and deleted.
  - If it is an internal node, delete and replace with the entry from the left position.
- After deletion, underflow is tested,
  - If underflow occurs, distribute the entries from the nodes left to it.
- If distribution is not possible from left, then
  - Distribute from the nodes right to it.
- If distribution is not possible from left or from right, then
  - Merge the node with left and right to it.

b) What is Normalization? Explain in detail about 1NF , 2NF , 3NF with examples. (6M)

Normalization is the process of efficiently organizing data in a database with two goals in mind
First goal: eliminate redundant data for example, storing the same data in more than one table
Second Goal: ensure data dependencies make sense for example, only storing related data in a table .
Advantages of Normal forms
Less storage space
Quicker updates
Less data inconsistency
Clearer data relationships
Easier to add data
Flexible Structure

First Normal Form **[Any relevant example can be considered]**

'A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only.'

Second Normal Form

‘A relation R is in second normal form (2NF) if and only if it is in 1NF and every nonkey attribute is fully functionally dependent on the primary key.’

Third Normal Form

‘A relation R is in third normal form (3NF) if and only if it is in 2NF and every nonkey attribute is nontransitively dependent on the primary key. OR

A relation schema R is in third normal form (3NF) if, whenever a nontrivial functional dependency X->A holds in R, either (a) X is a superkey of R, or (b) A is a prime attribute of R.

(OR)

15. a) Explain about sorted and unsorted file records in detail. (6M)

Writing any relevant points can be considered

In an ordered file, the records are sequenced on some field in the record (like StudentId or StudentName etc). In an unordered file, the records are not in any particular order.

Ordered File	Unordered File
RecordA	RecordM
RecordB	RecordH
RecordG	RecordB
RecordH	RecordN
RecordK	RecordA
RecordM	RecordK
RecordN	RecordG

Records are placed in the file in the order in which they are inserted, so new records are inserted at the end of the Such an organization is called a heap or pile file. This organization is often used with additional access paths, such as the secondary indexes. Inserting a new record is very efficient. The last disk block of the file is copied into a buffer, the new record is added, and the block is then rewritten back to disk. The address of the last file block is kept in the file header. However, searching for a record using any search condition involves a linear search through the file block by block an expensive procedure.

To delete a record, a program must first find its block, copy the block into a buffer, delete the record from the buffer, and finally rewrite the block back to the disk. This leaves unused space in the disk block. Deleting a large number of records in this way results in wasted storage space.

An ordered Sequential file, is a file ordered upon some key field. The ordering of the records in the file makes it possible to process an ordered file in ways that are not available to us with unordered files. While it is not really possible to apply batch updates or deletes to an unordered file these are possible with ordered files. Ordered records have some advantages over unordered files. First, reading the records in order of the ordering key values becomes extremely efficient because no sorting is required. Second, finding the next record from the current one in order of the ordering key usually requires no additional block accesses because the next record is in the
same block as the current one .

b) What are the problems caused by redundancy? Explain about Multivalued and join dependencies (6M)

Problems caused by redundancy [Explanation--2M]

Disadvantages of data redundancy include an unnecessary increase in size of databases, and the likelihood of data corruption being a direct result of redundancy. Other disadvantages include the likelihood of inconsistency of data as well as decreased efficiency of a database.
A change or modification, to redundant data, requires that you make changes to multiple fields of a database. While this is the expected behaviour for flat file database designs and spreadsheets.
Because of Data redundancy lots of storage space are wasted. It's difficult to update the database because there is duplicate data in table. Data redundancy is inefficient and worthless for several aspects and databases designers try to eliminate it as far as possible by using a technique known as normalization.

[**Multivalued dependency--2M]**

A multivalued dependency $X \twoheadrightarrow Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t_1 and t_2 exist in r such that $t_1[X] = t_2[X]$, then two tuples t_3 and t_4 should also exist in r with the following properties,15 where we use Z to denote $(R – (X ∗ Y))$:16

■ $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.
■ $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$.
■ $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$.

Whenever $X \twoheadrightarrow Y$ holds, we say that X **multidetermines** Y.

[**Join Dependecy --2M]**

A **join dependency** (**JD**), denoted by JD(R_1, R_2, ..., R_n), specified on relation schema R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have a nonadditive join decomposition into R_1, R_2, ..., R_n. Hence, for every such r we have

$$(\pi R1(r), \pi R2(r), \pi R3(r), \dots\dots\dots\dots \pi Rn(r)) = r$$

UNIT IV

16. a) What is a transaction? Explain in detail about ACID Properties. (6M)

Definition--2M

ACID Properties---4M

A transaction is typically implemented by a computer program, which includes database commands such as retrievals, insertions, deletions, and updates.Transactions should possess several properties, often called the **ACID** properties; they should be enforced by the concurrency control and recovery methods of the DBMS. The following are the ACID properties:

Atomicity. A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all.

Consistency preservation. A transaction should be consistency preserving, meaning that if it is completely executed from beginning to end without interference from other transactions, it should take the database from one consistent state to another.

Isolation. A transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executing.

Durability or permanency. The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.

b) Explain about 2-Phase Locking Protocol (6M)

In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories

- Lock based protocols
- Time stamp based protocols

Lock-based Protocols

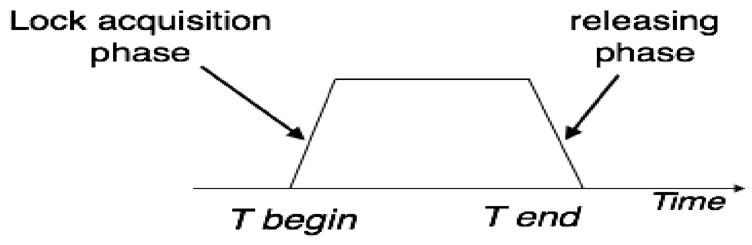
Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it.

Locks are of two kinds

- **Binary Locks −** A lock on a data item can be in two states; it is either locked or unlocked.
- **Shared/exclusive −** This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

Two-Phase Locking 2PL

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

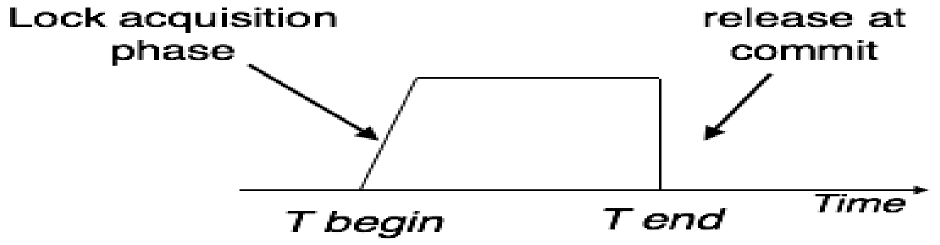


- Two-phase locking has two phases, one is **growing**, where all the locks are being acquired by the transaction; and the second phase is **shrinking,** where the locks held by the transaction are being released.

- To claim an exclusive (write) lock, a transaction must first acquire a shared (read) lock and then upgrade it to an exclusive lock.

Strict Two-Phase Locking

- The first phase of Strict-2PL is same as 2PL. After acquiring all the locks in the first phase, the transaction continues to execute normally. But in contrast to 2PL, Strict-2PL does not release a lock after using it. Strict-2PL holds all the locks until the commit point and releases all the locks at a time.



.

(OR)

17. a) What is shadow paging? Explain recovery technique based on immediate update. (6M)

Shadow paging ---2M

Shadow paging is an alternative to log-based recovery techniques, which has both advantages and disadvantages. It may require fewer disk accesses, but it is hard to extend paging to allow multiple concurrent transactions. The paging is very similar to paging schemes used by the operating system for memorymanagement. The idea is to maintain two page tables during the life of a transaction: the current page table and the shadow page table. When the transaction starts, both tables are identical. The shadow page is never changed during the life of the transaction. The current page is updated with each writeoperation. Each table entry points to a page on the disk. When the transaction is committed, the shadow page entry becomes a copy of the current page table entry and the disk block with the old data is released. If the shadow is stored in nonvolatile memory and a system crash occurs, then the shadow page table is copied to the current page table. This guarantees that the shadow page table will point to the database pages corresponding to the state of the database prior to any transaction that was active at the time of the crash, making aborts automatic.

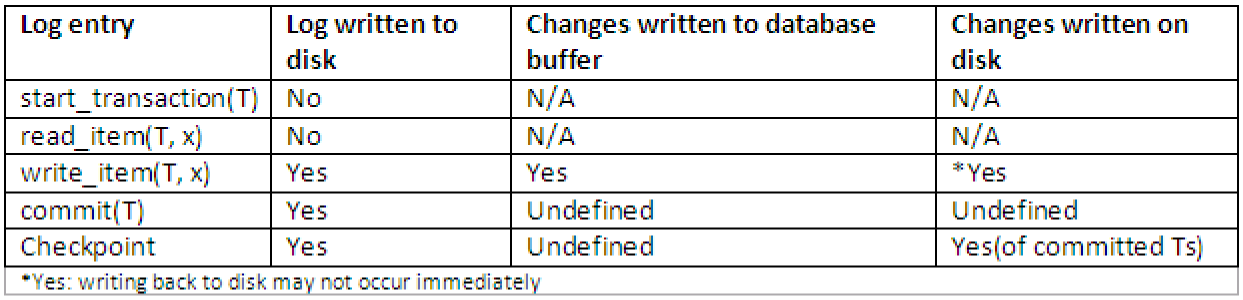
Recovery Techniques Based on Immediate Update **[Explanation--4M]**

In the immediate update techniques, the database may be updated by the operations of a transaction immediately, before the transaction reaches its commit point. However, these operations are typically recorded in the log on disk by force writing before they are applied to the database so that recovery is possible.

When immediate update is allowed, provisions must be made for undoing the effect of update operations on the database, because a transaction can fail after it has applied some updates to the database itself. Hence recovery schemes based on immediate update must include the capability to roll back a transaction by undoing the effect of its write operations.

1. When a transaction starts, write an entry start_transaction(T) to the log;
2. When any operation is performed that will change values in the database, write a log entry write_item(T, x, old_value, new_value);
3. Write the log to disk;
4. Once the log record is written, write the update to the database buffers;
5. When convenient write the database buffers to the disk;
6. When a transaction is about to commit, write a log record of the form commit(T);
7. Write the log to disk.

The protocol and how different entries are affected can be best summarised in Figure 10.7 below.



Log entry	Log written to disk	Changes written to database buffer	Changes written on disk
start_transaction(T)	No	N/A	N/A
read_item(T, x)	No	N/A	N/A
write_item(T, x)	Yes	Yes	*Yes
commit(T)	Yes	Undefined	Undefined
Checkpoint	Yes	Undefined	Yes(of committed Ts)

*Yes: writing back to disk may not occur immediately

.

b) What is Recoverability? Explain about mandatory access control. (6M)

Definition--2M

Recoverability refers to the ability to restore your deployment to the point at which a failure occurred. The ability to recover quickly from a system failure or disaster depends not only on having current backups of your data, but also on having a predefined plan for recovering that data on new hardware

Explanation--4M

Mandatory Access Control (MAC) is is a set of security policies constrained according to system classification, configuration and authentication. MAC policy management and settings are established in one secure network and limited to system administrators.

MAC advantages and disadvantages depend on organizational requirements, as follows:

- MAC provides tighter security because only a system administrator may access or alter controls.
- MAC policies reduce security errors.
- MAC enforced operating systems (OS) delineate and label incoming application data, which creates a specialized external application access control policy.

Signature of HOD