RELATIONAL MODEL & RELATIONAL DATABASE DESIGN

Multiple Choice Type Questions

1. Full form of SQL is [WBUT 2003, 2010]
   a) Structured Query Language b) Server Query Language
   c) Simple Query Language d) Simple Quotation Language
   Answer: (a)

2. SQL is a [WBUT 2016]
   a) procedural language b) non-procedural language
   c) complex language d) none of these
   Answer: (b)

3. For relations r_1 (A, B, C), r_2 (C, D, E) and r_3 (E, F), assume r_1 has 1000 tuples and r_2 has 1500 tuples and r_3 has 750 tuples. Maximum size of $r_1 \bowtie r_2 \bowtie r_3$ is [WBUT 2018]
   a) 1000 tuple b) 750 tuple c) 1500 tuple d) 1500750 tuple
   Answer: (a)

4. What operator performs pattern matching is SQL? [MODEL QUESTION]
   a) Except b) Intersect c) Like d) All of these
   Answer: (d)

5. The column of a table is referred to as [MODEL QUESTION]
   a) tuple b) attribute c) entity d) degree
   Answer: (b)

6. If two relations have 5 & 10 rows respectively, then what will be the no. of tuples in Cartesian product? [MODEL QUESTION]
   a) 5 b) 10 c) 15 d) 50
   Answer: (d)

7. SQL is relationally [MODEL QUESTION]
   a) Complete language b) Incomplete language
   c) Cannot handle certain relations d) Sound language
   Answer: (a)

8. The operation on certain relation X, produces Y such that Y contains only selected attributes of X, such an operation is [MODEL QUESTION]
   a) Projection b) Selection c) Union d) Difference
   Answer: (a)

9. A table can be logically connected to another table by defining a [MODEL QUESTION]
   a) a hyperlink b) Common field c) Primary key d) Foreign key
   Answer: (d)

10. DDL stands for [MODEL QUESTION]
    a) data-dictionary language b) dictionary defined language
    c) data defined language d) data definition language
    Answer: (d)

11. What operator performs pattern matching in SQL? [MODEL QUESTION]
    a) except b) intersect c) like d) all of these
    Answer: (d)

12. The primary key is selected from the: [MODEL QUESTION]
    a) Composite keys b) Determinants
    c) Candidate keys d) Foreign keys
    Answer: (a)

13. A key of table in RDBMS: [MODEL QUESTION]
    a) Must always be composed of two or more columns
    b) Can only be one column
    c) Identifies a row
    d) identifies a column
    Answer: (c)

14. A recursive relationship is a relationship between an entity and _____ [MODEL QUESTION]
    a) Itself b) A subtype entity
    c) An archetype entity d) An instance entity
    Answer: (a)

Short Answer Type Questions

1. What is the difference between Super key, Candidate key and Primary key? [WBUT 2004]

   OR,

   What do you mean by Super Key, Candidate key and Primary key? [WBUT 2011, 2012]

   Answer:

   The **primary key** of a relational table uniquely identifies each record in the table. It can either be a normal attribute that is guaranteed to be unique (such as Social Security Number in a table with no more than one record per person) or it can be generated by the DBMS (such as a globally unique identifier, or GUID, in Microsoft SQL Server). Primary keys may consist of a single attribute or multiple attributes in combination.

   Example:

CUSTOMER {custno, Name, Address}
ORDERS {Order No, Order Date, custno, part-no, project-no}
Here, the relation CUSTOMER having the key attribute custno, the ORDERS relation holds the custno as a non key attribute.
Thus custno in ORDERS is a foreign key w.r.t CUSTOMER

Refer to Question No. 3(b) of Short Answer Type Questions.

2. a) What is a View? Give one example. **[WBUT 2005]**
b) What is integrity constraint? List them. **[WBUT 2005]**
OR,
What do you mean by integrity constraint? Describe. **[WBUT 2011]**

Answer:

a) A view is a relation (virtual rather than base) and can be used in query expressions, that is, queries can be written using the view as a relation. In other words, a view is a named table that is represented, not by its own physically separate stored data, but by its definition in terms of other named tables (base tables or views). The base relations on which a view is based are sometimes called the existing relations. The definition of a view in a create view statement is stored in the system catalog. The syntax to create a view is:

CREATE [OR REPLACE] VIEW <view_name> [(<aliases>)] AS
<query> WITH {READ ONLY|CHECK OPTION [CONSTRAINT
<constraint_name>]}

b) An integrity constraint defines a business rule for a table column. When enabled, the rule will be enforced by oracle (and so will always be true.) To create an integrity constraint all existing table data must satisfy the constraint.

Default values are also subject to integrity constraint checking (defaults are included as part of an INSERT statement before the statement is parsed.)
If the results of an INSERT or UPDATE statement violate an integrity constraint, the statement will be rolled back.

Integrity constraints are stored as part of the table definition, (in the data dictionary.)
If multiple applications access the same table they will all adhere to the same rule.
There are two types of integrity constraints,
i) Entity integrity constraints and
ii) Referential integrity constraints

3. a) "All primary key is the super key but the converse is not true" — clarify. **[WBUT 2007, 2009, 2010, 2012, 2014, 2016, 2017]**
Define multi-valued attribute and composite attribute with suitable example. **[WBUT 2017]**
b) Define Candidate key, Alternate key with example. **[WBUT 2007, 2010]**

Answer:

a) 1st Part:

A super-key is a property where the set of attributes, which uniquely identifies a tuple of a relation. Let us now define the concept of super-key in a more formal way:
Let us consider two tuples t_1, and t_2, and for a given set of attributes R,
if $t_1[A] = t_2[A]$, for all $A \varepsilon R$ but $t_1 \# t_2$ then R cannot be a super key.
A set of attributes R is a super-key, if $t_1[A] = t_2[A]$ for all $A \varepsilon R$, the two tuples $t_1 = t_2$ are equal.
Now primary key is formed from the super key which includes minimum no of attributes.
It can be also said that primary key is minimal superkey.
So, there is no doubt that "All primary key is the super key but the converse is not true".

2nd Part:

Multi-Valued Attributes:

Occasionally, an attribute seems to hold multiple values simultaneously.
For example, consider an entity type PERSON, which has Language Spoken as an attribute. A distinguishing characteristic of a PERSON is, that they speak many languages. So, the attribute Language Spoken would seem to require multiple attribute values for each entity of PERSON.
An attribute that seems to require multiple values simultaneously is named a multivalued attribute.

Composite Attributes:

A composite attribute is one that is made up of other attributes.
An example of a composite attribute is the attribute Phone Number of the entity type CUSTOMER. Decomposing Phone Number into its elements is unnecessary in most business situations. If, however, the business happens to be a telephone company, there would be a need for Phone Number to be a composite attribute. This allows Phone Number to be treated as a single attribute in some cases, while still making it possible to inspect and modify its elements individually whenever necessary.

b) ***Candidate key***

In the relational model of databases, a **candidate key** of a relation is a minimal superkey for that relation; that is, a set of attributes such that

1. the relation does not have two distinct tuples with the same values for these attributes (which means that the set of attributes is a superkey)
2. there is no proper subset of these attributes for which (1) holds (which means that the set is minimal).

Since a relation contains no duplicate tuples, the set of all its attributes is a superkey if NULL values are not used. It follows that every relation will have at least one candidate key.

The candidate keys of a relation tell us all the possible ways we can identify its tuples. As such they are an important concept for the design database schema
Among the candidate keys which are not primary key are **alternate keys.**

4. "Minimal super key is candidate key". With a suitable example, justify the statement. **[WBUT 2013]**

Answer:
A key of a relational schema is equal to the minimal subset of a superkey that is still a key and is equal to a candidate key

Roll	First Name	Last Name
1	kabbir	Bal
2	Lovely	Chatterjee
3	Larry	Gomes

Now here we have the following as super keys
1. Roll
2. Roll | First Name
3. Roll | First Name | Last Name
Any unique key with some non unique key combination is the super key of the relationship. Thus 1,2, and 3 are the super keys.
A candidate key of a relationship is a set of attributes of that relationship such that there are no two distinct tuples with the same values for these attributes. In simple example candidate key is a minimal super key, i.e. a super key of which no proper subset is also a super key. So Roll is a candidate key. RDBMSs usually holds in each relation one of its candidate keys is declared as the primary key.

5. Explain Relational Algebra using the operators $\{\partial, \Pi, \cup, -, X$ **and show that:** $A \cap B = A \cup B - ((A-B) \cup (B-A))$.
Answer: **[WBUT 2014]**
First part:
The select operation: - to identify a set of tuples which is a part of a relation and to extract only these tuples out. The select operation selects tuples that satisfy a given predicate or condition.

- It is a unary operation defined on a single relation.
- It is denoted as **σ**.

Consider the following table "Book":

Code:

Acc-no	Yr-pub	title
734216	1982	Algorithm design
237235	1995	Database systems
631523	1992	Compiler design
543211	1991	Programming
376112	1992	Machine design

Example1:- Select from the relation "Book" all the books whose year of publication is 1992.
Code:

```
σ Yr-pub=1992(Book)
```

Example2:- Select from the relation "Book" all the books whose Acc-no is greater than equal to 56782.
Code:

```
σ Acc-no>=56782(Book)
```

The project operation: returns its argument relation with certain attributes left out.

- It is a unary operation defined on a single relation
- It is denoted as **Π**.

Example:- List all the Title and Acc-no of the "Book" relation.
Code:

```
Π Acc-no, Title (Book)
```

The union operation: is used when we need some attributes that appear in either or both of the two relations.

- It is denoted as **U**.

Example:
Borrower (customer-name, loan-number)
Depositor (customer-name, account-number)
Customer (customer-name, street-number, customer-city)

List all the customers who have either an account or a loan or both
Code:

```
Π customer-name (Borrower) U Π customer-name (Depositor)
```

For a union operation **r U s** to be valid, two conditions must hold:

- The relation r and s must be of the same arity, i.e. they must have the same number of attributes.
- The domains of the ith attribute of r and the ith attribute of s must be the same for all i.

The set difference operation: finds tuples in one relation but not in other.

- It is denoted as –

Example:
Find the names of all customers who have an account but not a loan.
Code:

```
Π customer-name (Depositor) - Π customer-name (Borrower)
```

The Cartesian product operation: - allows combining information from two relations.

- It is denoted as **r X s** where r and s are relations.

Consider the following relation or table "r":
Code:

```
+--------+-------+
| A      | B     |
+--------+-------+
|  a     |  1    |
|  b     |  2    |
|  a     |  2    |
+--------+-------+
```

Consider another relation or table "s":
Code:

```
+--------+-------+
| B      | C     |
+--------+-------+
|  3     |  1a   |
|  2     |  2b   |
+--------+-------+
```

Therefore, rXs gives:
Code:

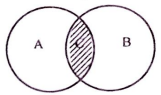
```
+----------+---------+---------+----------+
| r.A      | r.B     | s.B     | s.C      |
+----------+---------+---------+----------+
|  a       |  1      |  3      |  1a      |
|  a       |  1      |  2      |  2b      |
|  b       |  2      |  3      |  1a      |
|  b       |  2      |  2      |  2b      |
|  a       |  2      |  3      |  1a      |
|  a       |  2      |  2      |  2b      |
+----------+---------+---------+----------+
```


2nd part:
From the Venn diagram, the intersection of the two sets A & B in the shaded part i.e., C

$A \cup B = ((A - B) \cup (B - A)) \cup C$

$\Rightarrow (A \cup B) - ((A - B) \cup (B - A)) = C$

$\Rightarrow A \cap B - (A \cup B) - ((A - B) \cup (B - A))$ [Proved]



6. "All super keys are not candidate keys but the vice-versa is true". --- Justify the statement. **[WBUT 2018]**

Answer:

Super Key: A super key is a basic key of any relation. It is defined as a key that can identify all other attributes in a relation. Super key can be a single attribute or a set of attributes. Two entities do not have the same values for the attributes that compose a super key. There is at least one or more than one super keys in a relation.

A minimal super key is also called candidate key. So we can say some of the super keys get verified for being a candidate key.

Let Suppose R is a relation with the below attributes:

R (A, B, C, D, E, F) and it has the following dependencies:

Functional Dependencies	Super Key
AB → CDEF	✓
CD → AEF	✓
CB → DF	×
ABD → CEF	✓
DF → ABCE	✓
D → BC	×
DEF → ABC	✓

Using key, AB rest of the attributes (i.e. CDEF) of the table can be identified. Similarly, using keys CD, ABD, DF, and DEF we can identify remaining attributes of the table R. So all these are super keys.

But using a key CB we can only find values for attribute D and F, we cannot find the value for attributes A and E. Hence, CB is not a super key. Same is the case with key D, we cannot find the values of all attributes in a table using key D. So, D is not a super key.

Candidate Key: A super key that is a proper subset of another super key of the same relation is called a minimal super key. The minimal super key is called Candidate key. Like super key, a candidate key also identifies each tuple in a table uniquely.

In above example, we have found the Super keys for relation R. Now, let us check all the super keys for being Candidate key.

Functional Dependencies	Super Key	Candidate key
AB → CDEF	✓	✓
CD → ABEF	✓	✓
CB → DF	×	–

ABD → CEF	✓	✗
DF → ABCE	✓	✓
D → BC	✗	–
DEF → ABC	✓	✗

Super key AB is a proper subset of super key ABD. So, when a minimal super key AB alone can identify all attributes in a table, then we do not need bigger key ABD. Hence, super key AB is a candidate key while ABD will only be super key.
Similarly, a super key DF is also a proper subset of super key DEF. So, when DF is alone capable of identifying all attributes in a relation why do we need DEF. Hence, super key DF becomes a candidate key while DEF is only a super key.
The super key CD is not a proper subset of any other super key. So, we can say CD is a minimal super key that identifies all attributes in a relation. Hence, CD is a candidate key. Whereas key CB and D are not super key so, they cannot be the candidate key even. Viewing above table you can conclude that each candidate key is a super key but the inverse is not true.

7. Write the algorithm to find out the candidate key from given a relation. **[WBUT 2018]**

Answer:
For a given set of functional dependencies (F) on a relation R, the algorithm can be described as:

1. Find the attributes that are neither on the left and right side
2. Find attributes that are only on the right side
3. Find attributes that are only on the left side
4. Combine the attributes on step 1 and 3
5. Test if the closures of attributes on step 4 constitutes all the attributes. If yes, it is a candidate key.
6. If not, find the relation exteriors, that is the attributes not included in step 4 and step 2.
7. Now test the closures of attributes on step 4 + one attribute in step 6 one at a time. All those combinations are candidate keys if their closures constitute all the attributes.

8. Consider the relational database as given below and write down expressions in relational algebra for the following queries.
Material_Master (item_id, item name, reorder level)
Material_Dts (item_id, Supplier_id, Pharchase_date, Qty, Utcost)
i) Select the quantities of each purchased material alphabetically.
ii) Select the names of materials which have the highest total quantity.
iii) Replace the material name 'power supply' with 'UPS'. **[MODEL QUESTION]**

Answer:

i) $\tau_{\text{item\_name}}(\pi_{\text{item\_name, Qty}}\sigma_{\text{item\_id=item\_id}}$ (Material_Master ⋈ Material_Dts).

ii) χ Select m.item_name, Max(sum(d.qty)) from **Material-Master m, Material-Dts d where m.** item-id=d. item-id group by d. item-id ;

$_{\text{item-id}}\chi_{\text{SUM(Qty),item\_name}}(\sigma_{\text{item\_id=item\_id}}$(Material_Master ⋈ Material_Dts))

iii) $\pi_{\text{item\_id, reorder level, item\_name} \leftarrow \text{'UPS'}}(\sigma_{\text{item\_name='power supply'}}$(material_master))

9. Answer the following queries in relational algebra using the given database scheme: **[MODEL QUESTION]**
EMP (Eno, Ename, Eadd, Bdate, Super_no)
DEPT (Dno, Dname, Mgno)
PROJECT(Pno, Pname,Dno,Plocation)
WORKS_NO(Eno, Dno, Hours)
i) List the employee no, name, address of all employees working in the 'Research' Department.
ii) For all projects in 'Kolkata' print the project no, location, controlling department number and its manager's name, address and birthday.

Answer:

i) $\prod_{\text{Eno,Ename,Eadd}}(\sigma_{\text{p\_name = "DBMS"}}((\text{project})\ (\text{work\_on}))\ (\text{Employee}))$

ii) $A = \prod_{\text{p\_no, Plocation}}(\sigma_{\text{Plocation = "Kolkata"}}(\text{project}))$

$B = \pi_{\text{Dno, Dname, Mgno, Bdate}}$ (DEPT $\underset{\text{Mgno = Eno}}{}$ EMP)

Result = $\pi_{\text{p\_no, Plocation, Dno, Dname, Mgno, Bdate}}$ (A B)

10. What are the fundamental operations in relational Algebra? Explain each operation with example. **[MODEL QUESTION]**

Answer:

a) Relation-Oriented operation covers data manipulation through the, Selection, projection, Joins and division operations.

Selection: The selection operation selects tuples from a relation that satisfy the given condition and the operation is symbolized by σ. The syntax of the query is $\sigma_{\text{<condition>}}$ <Relation Name>

The output of the query is also a relation. Logical connectives and comparison operators can be used as selection conditions while executing the query.

The valid logical connectives are: AND(^) , OR (V) , NOT (≠) .The valid Comparison operations are: $<, >, =, \geq, \leq, \neq$

For example, let us execute the following queries using the relation **Employee (EMP#, name, Dept, Manager):**

Find the employees whose EMP# is greater than 100 and less than 104 and Working under the manager Patel

$\sigma_{\text{Emp>100 ∧ Emp<104 ∧ Manager='Patel'}}$ (EMPLOYEE)

Projection Operator: The projection operation projects all or an attribute or a set of selective attributes from a relation. It is a vertical subset of the given relation. The operation is symbolized by π.

The syntax of the projection query is:

π <List of attribute names/ offset position> (Relation Name)

For example, let us execute the following queries using the relation **Employee**
Find the employees whose EMP# is greater than 100 and less than 104 and Working under the manager Patel and show all the attributes

. $\Pi_{\text{emp\#, Name, Dept, Manager}}(\sigma_{\text{Emp>100 ^ Emp<104^Manager='Patel'}}(\text{EMPLOYEE}))$

Assignment operation

Sometimes it is useful to be able to write a relational algebra expression in parts using a temporary relation variable.

The assignment operation, denoted ← works like assignment in a programming language. We write to define assignment as:

$$\text{temp} \leftarrow \pi_{n-s}(r)$$

$$\text{temp} \leftarrow \pi_{n-s}((\text{temp} \times s) - r)$$

No extra relation is added to the database, but the relation variable created can be used in subsequent expressions. Assignment to a permanent relation would constitute a modification to the database.

Join operation: Join operation combines two or more relations to form a single new relation. The operation is symbolized by '⋈'. The join operation is a combination of Cartesian product, selection and projection operation. The conditional join is called **θ-join**. When equality condition holds in a θ-join, then it is called **equi-join**. When the attributes, equated in a join, have the same name in two relations the attribute name under the join operator need not required to be mentioned and is identified as **Natural join.**

The outer join operation is of three types namely left outer join, right outer join and full outer join.

Left Outer Join: In this type of outer join, the resulting relation retains all the tuples of the left relation. Symbolically left outer join is represented as '⟕'

Right Outer Join: In this type of outer join, the resulting relation retains all the tuples of the right relation. Symbolically right outer join is represented as '⟖'

Full Outer Join: In this type of outer join, the resulting relation retains all the tuples of both the relations and null values are padded with. Symbolically Full-Outer-Join is represented as '⟗'. The operation is primarily a union of two relations and are union compatible.

e.g.,

A.

No	Name
100	Smith
101	Jones
102	Evan
105	Raju
107	Robin
109	Suresh

B.

No	Name	Salary
100	Smith	5000
105	Raju	4000
107	Robin	7000

C.

Emp-No	Dept
101	Operation
107	Security
109	Accounts

Long Answer Type Questions

1. Considering the following tables serve the SQL queries, Relational Algebra & Relational Calculus: **[WBUT 2008]**

MARKS

ROLL	NAME	SUB	MARKS
01	ABHISHEK	PHYSICS	89
02	ANAND	CHEM	58
01	ABHISHEK	CHEM	80
03	BADAL	PHYSICS	65
03	BADAL	CHEM	75
02	ANAND	PHYSICS	80

ACC

ROLL	NAME	FEES_PAID
01	ABHISHEK	YES
02	ANAND	NO
03	BADAL	YES
04	RAJIB	YES

STUDENT

ROLL	NAME	STREAM
01	ABHISHEK	BCA
02	ANAND	BCA
03	BADAL	BCA
04	RAJIB	BCA

a) Find out the name of the students from STUDENT table whose 2nd letter of the name is 'A'. **[WBUT 2008]**

Answer:

SQL: Select Name from student where name like '_A%';

Algebra: $\pi_{\text{name}}\ O_{\text{name like '\_A\%'}}$ (Student)

Calculus: { Name | Nme ∈ student ∧ *Name* like '_A%'}

b) Find out the total marks obtained in the exam by ABHISHEK. **[WBUT 2008]**

Answer:
SQL: Select Name, Roll, marks from student where name='ABHISHEK ';
Algebra: $\pi_{name,roll,marks}\ O_{name='ABHISHEK'}$ (Student)
Calculas:{ Name[t],Roll[t],marks[t] | t ∈ student ∧ *Name* ='ABHISHEK'}

c) Find out the ROLL, NAME, STREAM, and FEES_PAID status of every student in ascending order. **[WBUT 2008]**

Answer:
SQL: Select Acc.Roll, Acc. Name, stream, fees_paid from ACC, STUDENT where Acc.Roll = student.Roll order by Acc.Roll;
Algebra: $\pi_{roll,\ name,\ stram,fees\text{-}paid}$ (ACC ⋈ Student)
Acc.Roll = student.Roll
Calculas: {Name[t],Roll[t],stream [s] | t ∈ ACC, s ∈ student ∧ Acc.Roll =student.Roll'}

d) Who obtained the highest marks in PHYSICS? Display ROLL, NAME, SUB, MARKS. **[WBUT 2008]**

Answer:
SQL: Select ROLL, NAME, SUB, MARKS from MARKS where SUB='PHYSICS' and MARKS=MAX(MARKS);
Algebra: $\pi_{roll,\ name,\ sub,\ marks}\ O_{sub='physics'\ and\ marks=max(marks)}$ (MARKS)
Calculas:{roll[t], name[t], sub[t], marks[t] | t ∈ Marks ∧ SUB[t]='PHYSICS' ∧ MARKS[t]=MAX(MARKS)}
Note: In calculas max function does not exist. As it is non procedural language, an attempt is made to execute the query in calculas.

e) Who obtained the highest marks in PHYSICS and CHEM? **[WBUT 2008]**

Answer:
SQL: Select ROLL, NAME, SUB, MARKS from MARKS where (SUB='PHYSICS' and MARKS=MAX(MARKS)) AND (SUB='CHEMISTRY' and MARKS=MAX(MARKS));
Algebra:
$\pi_{roll,\ name,\ sub,\ marks}\ O_{(sub='physics'\ and\ marks=max(marks))\ and\ (sub='chemistry'\ and\ marks=max(marks))}$ (MARKS)
Calculas:
{ roll[t], name[t], sub[t], marks[t] | t ∈ Marks ∧ (SUB[t]='PHYSICS' ∧ MARKS[t]=MAX(MARKS)) ∧ (SUB[t]='CHEMISTRY' ∧ MARKS[t]=MAX(MARKS))}

2. Considering the following tables serve the SQL queries. Relational Algebra and Relational calculus. **[WBUT 2009]**

Run

Jersy_ No.	Player_ Name	Against	Run
11	Tendulkar	Sri Lanka	89
23	Mendis	India	58
11	Tendulkar	Pakistan	80
15	Akram	India	58
15	Akram	Sri Lanka	38
23	Mendis	Pakistan	38

Team

Jersy_ No.	Player_ Name	Team
11	Tendulkar	India
23	Mendis	Sri Lanka
15	Akram	Pakistan

a) Find out the names of the players from Team table whose 2nd letter of the name is *E*. **[WBUT 2009]**

Answer:
SQL: Select Player_Name from Team where Player_Name like '_e%';
Relational Algebra: $Pi_{player\_name}(sigma_{player\text{-}name\ like\ '\_e\%'}(Team))$
Tuple Relational Calculus :
{s. Player_Name | Team(s)^ s. Player_Name like'_e%'}

b) Find out the total runs scored in the tournament by Tendulkar. **[WBUT 2009]**

Answer:
SQL: Select Sum(Run) from Run group by Player_Name where Player_Name = 'Tendulkar';
Relational Algebra:
$_{player\_name}Gamma_{sum(Run)}(Sigma_{player\_name='Tendulkar'}(RUN))$
Tuple Relational Calculus:
{SUM (*s*.run) | RUN(*s*) ∧ *s*. Player_Name = 'Tendulkar '}

c) Find out the Jersy_No., Player_Name, Team of every player in ascending order. **[WBUT 2009]**

Answer:
Select Jersey_No, Player_Name, Team from Team Order by Jersey_No;
Relational Algebra:
Tao $_{Jersey\_No}$(Pi $_{Jersey\_No,\ Player\_Name,\ Team}$(Team))

Tuple Relational Calculus:
Cannot specify order by in relational calculus

d) Who scored the highest run against India? Display Jersy_No, Player_Name and Run. **[WBUT 2009]**

Answer:
SQL:
Select Jersey_No, Player_Name, Max(run) from run where Against='INDIA';
Relational Algebra:
Gamma $_{Jersey\_No,\ Player\_Name\ ,Max(Run)}$(Sigma $_{Against='INDIA'}$(RUN))

e) Who scored the highest run against India and Sri Lanka? Display. [WBUT 2009]

Answer:

SQL: Select select Jersy_No, Player_Name, Max(run),Against from run where where Against= 'INDIA' or; Against= 'SRI LANKA';

Relational Algebra:

Gamma $_{\text{Jersey\_No, Player\_Name ,Max(Run)}}$(Sigma $_{\text{Against='INDIA' or Against='SRI LANKA'}}$ (RUN))

Relational calculus does not provide aggregate functions, so 'd' and 'e' can't be answered.

3. Define functional dependencies.

[WBUT 2006, 2009, 2010, 2014-Short Note, 2016]

Explain Armstrong rules. [WBUT 2006, 2010]

OR,

Define functional dependency. Describe Armstrong's axioms. [WBUT 2008, 2011]

OR,

State Armstrong's Axioms. [WBUT 2015]

Answer:

a) A functional dependency (FD) is a constraint between two sets of attributes in a relation from a database.

Given a relation R, a set of attributes X in R is said to functionally determine another attribute Y, also in R, (written $X \rightarrow Y$) if and only if each X value is associated with precisely one Y value. Customarily we call X the determinant set and Y the dependent attribute. Thus, given a tuple and the values of the attributes in X, one can determine the corresponding value of the Y attribute. For the purposes of simplicity, given that X and Y are sets of attributes in R, $X \rightarrow Y$ denotes that X functionally determines each of the members of Y—in this case Y is known as the dependent set. Thus, a candidate key is a minimal set of attributes that functionally determine all of the attributes in a relation.

A set of inference rule is said to be sound if we cannot obtain any dependency that is not in F^+

Let us consider a relation $R=\{A_1, A_2, A_3, \ldots\ldots, A_n\}$ and let r be a instance of R

Reflexivity

Let $Y, X \in R$. Then for any two tuples t_1 and t_2 or r

$t_1[x] = t_2[x]$ and we can also say $t_1[Y] = t_2[Y]$ hence $X \rightarrow Y$

Augmentation

Let us suppose that r satisfies $X \rightarrow Y$ and let $Z, W \in R$.

Then for any two tuples t_1 and t_2 or r such that $t_1[xW] = t_2[xW]$

Since r satisfies $X \rightarrow Y$ and X values of t_1 and t_2 are equal $t_1[Y] = t_2[Y]$.

Also W values of t_1 and t_2 being $t_1[Z] = t_2[Z]$

Therefore, $XW \rightarrow YZ$ holds in r.

Union

To prove that: if $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ then $\alpha \rightarrow \beta\gamma$

We derive:

$\alpha \rightarrow \beta$ given

$\alpha\alpha \rightarrow \alpha\beta$ augmentation rule

$\alpha \rightarrow \alpha\beta$ union of identical sets

$\alpha \rightarrow \gamma$ given

$\alpha\beta \rightarrow \gamma\beta$ augmentation rule

$\alpha \rightarrow \beta\gamma$ transitivity rule and set union commutativity

Transitivity

Suppose that r satisfies $X \rightarrow Y$ and $Y \rightarrow Z$, then for any two tuples t_1 and t_2 of r , if $t_1[x] = t_2[x]$ by $X \rightarrow Y$, $t_1[Y] = t_2[Y]$

Again $Y \rightarrow Z$, requiers $t_1[Z] = t_2[Z]$. Hence r satisfies $X \rightarrow Z$

Reflexivity rule leads to trivial dependencies. Trivial dependencies does not depend on the given set of dependencies.

The use of augmentation and the transitivity depends on the given set of dependencies and these rules can be repeatedly used to infer new non-trivial dependencies.

Decomposition Rule

If $\alpha \rightarrow \beta\gamma$ then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$

Let $\alpha \rightarrow \beta\gamma$ --------- (A)

Now, $\gamma \subseteq \beta\gamma$ $\quad \therefore \beta\gamma \rightarrow \gamma$ --------- (B)

(From Reflexivity Rule)

from (A) and (B) through Transitivity Rule

$\alpha \rightarrow \gamma$

In manner in can be shown $\alpha \rightarrow \beta$

Since, $\beta \subseteq \beta\gamma$.

4. Consider the following relational schema: [WBUT 2007]

Sailors (sid, sname, rating, age)
Reserved (sid, bid, day)
Boats (bid, bname, color)

Give an expression to express each of the following queries with specified language:

(i) Find the names of the sailors who have reserved a red or a green boat. (Relational Algebra)

(ii) Find the names of the sailors who have reserved at least two boats (Tuple Relational Calculus)

(iii) Find the names of the sailors with age over 20 who have not reserved a red boat. (Relational Algebra)

(iv) Find the names of the sailors who have reserved all boats. (Tuple Relational Calculus)

(v) Find the colour of boats reserved by Smith. (SQL Query)

Answer:

i) $\Pi_{\text{sname}}\, \sigma_{\text{colour='Red' or colour='green'}}$ ((SAILORS Theta join RESERVES) Equijoin Boats)

ii) $\Pi_{sname}\ \sigma_{bid>=2}$ (SAILORS Theta join RESERVES).
iii) $\Pi_{sname}\ \sigma_{colour\neq'Red'\ \wedge\ age>20}$ ((SAILORS Theta join RESERVES) Equijoin Boats)
iv) $\Pi_{sname}\ \sigma_{bid\neq Null}$ (SAILORS Theta join RESERVES)
v) $\Pi_{sname,\ colour}\ \sigma_{sname='Smith'}$((SAILORS Theta join RESERVES) Equijoin Boats).

5. a) Explain Codd's rules of DBMS.
b) What is Relational Algebra? Explain Select, Projection, Set Union, Set Difference, Set Intersection with example. [WBUT 2009]

Answer:

a) Codd's12 rules are a set of thirteen rules (numbered zero to twelve) proposed by Edgar F. Codd is designed to define what is required from a database management system in order for it to be considered *relational*, i.e., a relational database management system RDBMS

The rules

Rule 0: The system must qualify as *relational*, as a *database*, and as a *management system*.

For a system to qualify as a relational database management system (RDBMS), that system must use its *relational* facilities (exclusively) to *manage* the *database*.

Rule 1: The *information rule:*

All information in the database is to be represented in one and only one way, namely by values in column positions within rows of tables.

Rule 2: The *guaranteed access rule:*

All data must be accessible. This rule is essentially a restatement of the fundamental requirement for primary keys. It says that every individual scalar value in the database must be logically addressable by specifying the name of the containing table, the name of the containing column and the primary key value of the containing row.

Rule 3: *Systematic treatment of null values:*

The DBMS must allow each field to remain null (or empty). Specifically, it must support a representation of "missing information and inapplicable information" that is systematic, distinct from all regular values (for example, "distinct from zero or any other number", in the case of numeric values), and independent of data type. It is also implied that such representations must be manipulated by the DBMS in a systematic way.

Rule 4: *Active online catalog based on the relational model:*

The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language. That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.

Rule 5: The *comprehensive data sublanguage rule:*

The system must support at least one relational language that

1. Has a linear syntax
2. Can be used both interactively and within application programs,
3. Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations (begin, commit, and rollback).

Rule 6: The *view updating rule:*

All views that are theoretically updatable must be updatable by the system.

Rule 7: *High-level insert, update, and delete:*

The system must support set-at-a-time *insert*, *update*, and *delete* operators. This means that data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

Rule 8: *Physical data independence:*

Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.

Rule 9: *Logical data independence:*

Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure. Logical data independence is more difficult to achieve than physical data independence.

Rule 10: *Integrity independence:*

Integrity constraints must be specified separately from application programs and stored in the catalog. It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications.

Rule 11: *Distribution independence:*

The distribution of portions of the database to various locations should be invisible to users of the database. Existing applications should continue to operate successfully:

1. when a distributed version of the DBMS is first introduced; and
2. when existing distributed data are redistributed around the system.

Rule 12: The *nonsubversion rule:*

If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint.

b) A relational algebra expression of a query specifies only partially how to evaluate a query, there are several ways to evaluate an relational algebra expression. For example, consider the query:

SELECT Salary FROM EMPLOYEE WHERE Salary >= 50,000;

The possible relational algebra expressions for this query are:

- $\Pi_{Salary}(\sigma_{Salary>=50000}(EMPLOYEE))$
- $\sigma_{Salary>=50000}(\Pi_{Salary}EMPLOYEE$

Selection: The selection operation selects tuples from a relation that satisfy the given condition and the operation is symbolized by σ. The syntax of the query is $\sigma_{<condition>}$

For example,

$\sigma_{Dept='Accounts'}$ **(EMPLOYEE)**

Projection: The projection operation projects all or an attribute or a set of selective attributes from a relation. It is a vertical subset of the given relation. The operation is symbolized by π. The syntax of the projection query is:

$\pi_{<\text{List of attribute names/ offset position}>}$ (Relation Name)

For example,

$\pi_{emp\#, name}$(EMPLOYEE)

Set Union:

$R = A \cup B$ s.t. $R = \{t \mid t \in A \vee t \in B\}$

Set Difference:

$R = A - B$ s.t. $R = \{t \mid t \in A \wedge t \notin B\}$

Set Intersection:

$R = A \cap B$ s.t $R = \{t \mid t \in A \wedge t \in B\}$

6. What do you mean by Super key, Candidate key and Primary key? Take a single example of a database and explain the relationship between primary key, candidate key, foreign key in the same example. **[WBUT 2015]**

Answer:

1st Part: ***Refer to Question No. 1 of Short Answer Type Questions.***

2nd part:

Consider the following table

Candidate key	Candidate key		
Student_ID	First_name	Last_name	Course_id
X001	Joy	Roy	CS02
X002	Mohit	Sen	IT01
Xoo3	Raja	Das	M03
X004	Soma	Basu	B06

In this example, we might have a student_id that uniquely identifies the student in student table. They should be candidate key. But sometime in the above table we might have the student First_name and Last_name that also when combined, uniquely identify the student in the table. So, both of these would be candidate key.

In order to eligible candidate key it must past same criteria.

i) It must contain unique value

ii) It must not contain null value

iii) It contain minimum number of fields to ensure uniqueness

iv) It must uniquely identify each record in the table.

Once candidate keys have been identified then we can select are to be as primary key.

Primary key			
Student_ID	First_name	Last_name	Course_id
X001	Joy	Roy	CS02
X002	Mohit	Sen	IT01
Xoo3	Raja	Das	M03
X004	Soma	Basu	B06

In the above table, Student_id might be a primary key.

Primary key are mandatory for every table each record must have a value for its primary key. When choosing a primary key form pool of a candidate keys always choose a single simple key over a composite key.

Foreign keys

A foreign key is a primary they from one table that appears as a field in another where the first table has a relationship to the second.

In the example student table their contains the Course_id the student is attending another table list of courses offered with the Course_id begin the primary key. The table 2 linked with the table 1 through Course_id. So Course_id is the foreign key.

Candidate key	Candidate key		Foreign key
Student_ID	First_name	Last_name	Course_id
X001	Joy	Roy	CS02
X002	Mohit	Sen	IT01
X003	Raja	Das	M03
X004	Soma	Basu	B06

Relationship

Primary key

Course_id	Course_name
CS02	Computer Science (Semester 1)
IT01	Information Technology (Semester 2)
M03	MBA (Semester 3)
B06	BCA (Semester 6)

7. Explain with two examples why the set $\{\sigma, \pi, \cup, -, X\}$ is called the complete set of relational algebra operation. **[WBUT 2015]**

Answer:

The set of operations including select σ project π, Union $\cup$, set difference $-$, and Cartesian product X is called a complete set because any other relation algebra expression can be expressed by combination of these find operations.

Example

- $R \cap S = (R \cup S) - (R - S) \cup (S - R)$
- $R \bowtie_{<\text{join operation}>} S = \sigma_{<\text{join operation}>}(E \times S)$

8. Write SQL Query to perform the following operations on a Relation.
i) Add a constraint to an attribute
ii) Remove a constraint from an attribute
iii) Remove an attribute
iv) Grant DBA authority to user
v) Revoke authorities on a relation from any user
vi) Find second max sal of a salary attribute in a relation
vii) Insert a Date
viii) Insert system date
ix) Finding grouped average and filtering some groups out of all groups
x) Deleting duplicate tuples in a relation. **[MODEL QUESTION]**

Answer:

i) alter table <table_name> add constraint <constr_name(<spec)>; example:
alter table dept add constraint chk_deptno check(deptno between 10 and 50);

ii) alter table <table_name> drop constraint constr_name;
example:-
alter table dept drop constraint chk_deptno;

iii) alter table table_name drop column col_name;
[note:-col_name is an existing column]

iv) grant dba to user_name;
[note:-current user must have dba role]

v) revoke update on object_name from username;
[note:-object_name may be a table/view]

vi) select sal from
(select sal from emp
minus
select max(sal) from emp order by sal desc)
where rownum=1;

vii) insert into table_name values(, 'dd-mon-yyyy');
[note:- dd- Day number mon- month name in 1st 3 character
yyyy-4 digit year]

viii) insert into table_name values(sysdate);

ix) select avg(sal),deptno from table 1 group by deptno having dept='D2';

Table: 1

Empno	Sal	Deptno
E0	10	D1
E1	20	D1
E2	30	D2
E3	30	D2
E4	15	D2
E5	10	D3
E6	20	D3

The data table is assumed and the above query is framed.

x) Delete from table 2 where rowid not in (select min(rowid) from table 2 group by c1,c2);

Table: 2

C1	C2
2	ff
3	gg
3	rr
4	ff
2	fff
3	fdg
3	hhj
3	hhj
3	fdg

9. Write short notes on the following:
a) Use of Relational algebra in DBMS **[WBUT 2005]**
OR,
Relational Algebra **[WBUT 2008]**
b) Codd's rule **[WBUT 2011, 2016]**
c) Selection and Projection **[WBUT 2011, 2018]**
d) Tuple Relational Calculus **[MODEL QUESTION]**
e) Domain relational Calculus **[MODEL QUESTION]**

Answer:

a) Relational algebra:

Relational algebra is a procedural language used for the evaluation of the relational database queries. It specifies the operations to be performed on the existing relations to derive result relations. It expresses the complete scheme of each of the result relations. The relational algebra operations can be divided into two broad categories:

1. Set-Oriented operation
2. Relation-Oriented Operation

Set-oriented Basic Operations are traditional operations which includes union, difference, intersection and Cartesian product. The operations union, difference, intersection are union compatible.

Relation-Oriented operation covers data manipulation through the, Selection, projection, Joins and division operations.

b) Codds rule: ***Refer to Question No. 5(a) of Long Answer Type Questions.***

c) Selection and Projection:

Refer to Question No. 5(b) of Long Answer Type Questions.

d) Tuple Relational Calculus:

The tuple relational calculus is a non procedural language (declarative language). It describes the desired information without giving a specific procedure for obtaining that information. A query in the tuple relational calculus is expressed as $\{t \mid P(t)\}$ i.e. it is the set of all tuples t such that predicate P is true for t. P is a well formed formula. A well

formed formula consists of a set of atoms, logical connectives and quantifiers connect those items.

Safe Expression: It is possible to write tuple calculus expression that generates infinite relations.

For example, $\{ t \mid \neg t[Name] = 'Asutosh\ Saxena' \}$.

This expression only specifies all tuples corresponding to anyone except Asutosh Saxena to generate. As a result there can be infinite number of tuples. The **safe expressions** guard such problems. Safe expressions in tuple relational calculus are relationally complete.

Equivalence Expression:

- $P_1 \wedge P_2 = \neg(\neg P_1 \vee \neg P_2)$
- $\forall t \in r(P(t)) = \neg \exists t \in r(\neg P(t))$
- $P_1 \Rightarrow P_2 = \neg P_1 \vee P_2$.

Deletion Operation: Deletion is expressed in much the same way as a query. Instead of displaying, the selected tuples are removed from the database. We can only delete whole tuples. In relational algebra, a deletion is of the form $r \leftarrow r - E$ where r is the relation and E is a relational algebra query. Tuples in r for which E is true are deleted. For example; Delete all of Ramesh's account records.

deposit ← deposit – s_{cname}="*Ramesh*"(deposit)

Insertion Operation

1. To insert data into a relation, we either specify a tuple, or write a query whose result is the set of tuples to be inserted. Attribute values of or inserted tuples must be members of the attribute's domain.
2. An insertion is expressed by $r \leftarrow r \cup E$

where r is a relation and E is a relational algebra expression.

3. Example:

To insert a tuple for Ramesh who has Rs1200/- in his account at the kolkata Branch.

deposit ← deposit ∪ {"Kolkata","*Ramesh*",1200}

Update Operation

Updating allows us to change some values in a tuple without necessarily changing all.

Let us call the update operator as δ and the operation is of the form

$$\delta_A \leftarrow E(R)$$

where R is a relation with attribute A, which is assigned the value of Expression E.

The Expression E is any arithmetic expression involving constants and attributes in relation R. For example; increase the deposit by 16%

$\delta_{balance}$ ←balance +.16(deposit), applied to every tuple in deposit.

e) Domain Relational Calculus:

The **domain relational calculus** has domain variables i.e., variables that range over the underlying domains instead of over relation. A domain variable consists of the domains on which the relations are defined. Each query is an expression of the form:

{ < x1,x2,……..,xn >| P(x1,x2,……..,xn)} where x1,x2,…,xn are domain variables and P is a predicate represents a formula composed of atoms. An atom in the domain relational calculus has one of the following forms: -

(1) $<x_1, x_2, \ldots\ldots, x_n> \in r$, where r is a relation on x attributes and $x_1, x_2, \ldots\ldots, x_n$ are domain variable or domain constants.

(2) $x \Theta y$, where x and y are domain variables and Θ is a comparison operator ($<, \leq, =, \neq, >, \geq$). We require that attributes x and y have domains that can be compared by Θ

(3) $x \Theta c$, where x is a domain variable, Θ is a comparison operator and c is a constant in the domain of the attribute for which x is a domain variable.

NORMALIZATION

Multiple Choice Type Questions

1. Normalization of database is needed to [WBUT 2006, 2007, 2015]
   a) make data more intelligible to humans b) remove error in data entry
   c) eliminate redundant data d) all of these
Answer: (c)

2. Given a relation R, attribute A is on B if each value of A in R is associated with precisely one value of B. [WBUT 2007]
   a) Multi valued dependency b) Data independence
   c) Functional dependency d) Transitive dependency
Answer: (c)

3. When all the columns in a relation describe and depend upon the primary key. The relation is said to be in [WBUT 2008]
   a) 1NF b) 2NF c) 3NF d) 4NF
Answer: (b)

4. Whenever two independent one –to-many relationships are mixed in the same relation, a ..... arise. [WBUT 2008]
   a) functional dependency b) multivalued dependency
   c) transitive dependency d) none of these
Answer: (a)

5. A normal form in which every determinant is a key is [WBUT 2009]
   a) 2 NF b) 3NF c) BCNF d) 4NF
Answer: (c)

6. Given the relation schema Bank (Bank ID, Account Numb, Balance, Customer) with FDs:
{BankID, AccountNumb → Balance; BankID, AccountNymb → Customer; Customer → BankID}
What is the highest normal form for the relation schema Bank? [WBUT 2009]
   a) First b) Second c) Third d) Boyce Code
Answer: (d)

7. A relation is considered to be in second normal form if it is in first normal form and it has no ..... dependencies [WBUT 2009, 2012]
   a) referential b) functional c) partial key d) transitive
Answer: (c)

8. BCNF is a type of [WBUT 2010]
   a) Indexing b) DFD c) Normalization d) None of these
Answer: (c)

9. Normalization removes [WBUT 2012, 2017]
   a) dependency of data b) uniqueness of data
   c) redundancy of data d) none of these
Answer: (c)

10. Functional dependency is the dependency between [WBUT 2013]
   a) Tuples b) Attributes c) Values d) None of these
Answer: (b)

11. 2NF is based on _____ dependency [WBUT 2014]
   a) transitive b) total c) partial d) functional
Answer: (b)

12. If $X \supseteq Y$ then $X \rightarrow Y$ is an example of _____ dependency [WBUT 2014]
   a) partial b) join c) non trivial d) trivial
Answer: (d)

13. Which of the following is true? [WBUT 2015]
   a) A relation in BCNF is always in 3NF
   b) A relation in 3NF is always in BCNF
   c) BCNF and 3NF are same
   d) A relation in BCNF is not in 3NF
Answer: (a)

14. Consider a schema $R(A, B, C, D)$ and functional dependencies $A \rightarrow B$ and $C \rightarrow D$. Then the decomposing $R_1(A, B)$ and $R_2(C, D)$ is [WBUT 2015]
   a) dependency preserving but not lossless join
   b) dependency preserving and lossless join
   c) lossless join but not dependency preserving
   d) lossless join
Answer: (a)

15. $R = (A, B, C, D)$, $F = (AB \rightarrow C, C \rightarrow D)$. Find candidate key [WBUT 2015]
   a) AB b) ABC c) $ABCD$ d) none of these
Answer: (a)

16. Which form has a relation that possesses data about an individual entity? [WBUT 2018]
   a) 2NF b) 3NF c) 4NF d) 5NF
Answer: (c)

17. Consider the following R1 relation and functional dependencies. [WBUT 2018]
R1 (a, b, c, d, e, f)
FD1 : a → b, c, d, f
FD2 : d → f
a) relation R1 is in Boyce Codd normal form.
b) relation R1 is in Third normal form.
c) relation R1 is not in Third normal form due to transitive dependency
d) None of the above
Answer: (c)

Short Answer Type Questions

1. a) What in normalization? [WBUT 2003]

OR,

Note on 'Normalization'. [WBUT 2008, 2011]
b) Explain 1NF, 2NF and 3NF with examples. [WBUT 2003, 2006, 2016]
Answer:
a) Normalization
Relations resulting from a well-written E-R design may still have some undesirable properties, e.g. composite attributes, multi-valued attributes, data redundancy etc. In order to remove these undesirables we apply a process called **normalization**. The central concept is the notion of *functional dependency*.

b) 1NF
A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only
Example: 1NF but not 2NF
FIRST (supplier_no, status, city, part_no, quantity)
Functional Dependencies:
(supplier_no, part_no) → quantity
(supplier_no) → status
(supplier_no) → city
city → status (Supplier's status is determined by location)
Non-key attributes are not mutually independent (city ® status).
Non-key attributes are not fully functionally dependent on the primary key (i.e., status and city are dependent on just part of the key, namely supplier_no).

2NF
A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key
SECOND (supplier_no, status, city)
Functional Dependencies:
supplier_no → status
supplier_no → city
city → status

3NF
A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. An attribute C is transitively dependent on attribute A if there exists an attribute B such that: A → B and B → C. Note that 3NF is concerned with transitive dependencies which do not involve candidate keys. A 3NF relation with more than one candidate key will clearly have transitive dependencies of the form: primary_key → other_candidate_key → any_non-key_column
SUPPLIER_PART (supplier_no, supplier_name, part_no, quantity)

Functional Dependencies:
We assume that supplier_name's are always unique to each supplier. Thus we have two candidate keys:
(supplier_no, part_no) and (supplier_name, part_no)

Thus we have the following dependencies:
(supplier_no, part_no) → quantity
(supplier_no, part_no) → supplier_name
(supplier_name, part_no) → quantity
(supplier_name, part_no) → supplier_no
supplier_name → supplier_no
supplier_no → supplier_name

2. Define BCNF and show that if a relation schema is in BCNF, and then it is also in 3NF. [WBUT 2004]

OR,

Define BCNF with example. [WBUT 2016]
Answer:
Boyce-Codd normal form (or **BCNF** or **3.5NF**) is a normal form used in database normalization. It is a slightly stronger version of the third normal form (3NF). A table is in Boyce-Codd normal form if and only if for every one of its nontrivial dependencies $X \rightarrow Y$, X is a superkey—that is, X is either a candidate key or a superset thereof.
A relation R is in 3NF if and only if every dependency A->B satisfied by R meets at least ONE of the following criteria:
1. A->B is trivial (i.e. B is a subset of A)
2. A is a superkey
3. B is a subset of a candidate key

BCNF doesn't permit the third of these options. Therefore a relation schema is in BCNF, and then it is also in 3NF.
Consider the following relation and determinants.
R(a,b,c,d)
a,c -> b,d
a,d -> b

Here, the first determinant suggests that the primary key of R could be changed from a,b to a,c. If this change was done all of the non-key attributes present in R could still be determined, and therefore this change is legal. However, the second determinant indicates that a,d determines b, but a,d could not be the key of R as a,d does not determine all of the non key attributes of R (it does not determine c). We would say that the first determinate is a candidate key, but the second determinant is not a candidate key, and thus this relation is not in BCNF (but is in 3rd normal form).

3. Consider the relation R = {A, B, C, D},
Functional Dependencies F = {A —> B, C —> D} and
the Decomposition ρ = {R1, R2} where R1= {A,B} and R2 ={C,D}. Show that the decomposition is not lossless but it preserves dependencies. [WBUT 2007]

Answer:

Loss less decomposition can be proved by the matrix method as follows:
The initial matrix is

	A	B	C	D
R1	a1	a2	b13	b14
R2	b21	b22	a3	a4

Now we apply the relation A->B on the initial matrix, over R1 and R2

	A	B	C	D
R1	a1	a2	b13	b14
R2	~~b21~~ a1	~~b22~~ a2	a3	a4

Now the element of row R2 has attained all a's. Thus the decomposition is lossless.

	A	B	C	D
R1	a1	a2	b13	b14
R2	a1	a2	a3	a4

A decomposition is dependency preserving if the closure of
F'= (F1 ¿ F2.....)
F+ identical to F+2F2
F+ ={(a,b) ¿ (c,d)}={a,b,c,d} [Thus dependency preserving]

4. Given the relation R (A, B, C, D, E) and functional dependencies: [WBUT 2008]

$A \rightarrow BC$
$CD \rightarrow E$
$B \rightarrow D$
$E \rightarrow A$

Check whether
i) decomposition R1 = (A, B, C) and R2 = (A, D, E) is lossless or not.
ii) depending is preserved or not.

Answer:

Closure

Let the given dependencies are F then the closure is denoted by F^+
F+={ABC} , Using B →D
F+ <new> = {ABCD}, using CD → E
F+ <new> = {ABCDE}

	A	B	C	D	E
R1	a1	a2	a3	b14	c14
R2	a1	b21	b23	a4	a5

In order to prove the loss less decomposition, we form the initial matrix. Now we apply the relation A → BC on R1 and R2 and the transformed matrix stands

	A	B	C	D	E
R1	a1	a2	a3	b14	c14
R2	a1	~~b21~~ a2	~~b23~~ a3	a4	a5

The elements of row R2 converted to all a's. Thus the decomposition is loss less.
A decomposition is dependency preserving if the closure of F'= (F1 ⋃ F2.....)
F^+ ={(a,b,c) ⋃ (c,d,e)}= (a,b,c,d,e} [Thus dependency preserving]

5. What is meant by canonical cover of a set of functional dependencies? Is the canonical cover unique? [WBUT 2008]

Answer:

Given that X, Y, and Z are sets of attributes in a relation R, one can derive several properties of functional dependencies. Among the most important are Armstrong's axioms, which are used in database normalization:
Subset Property (Axiom of Reflexivity): If Y is a subset of X, then $X \rightarrow Y$
Augmentation (Axiom of Augmentation): If $X \rightarrow Y$, then $XZ \rightarrow YZ$
Transitivity (Axiom of Transitivity): If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
From these rules, we can derive these secondary rules:
Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
Pseudotransitivity: If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$
Equivalent sets of functional dependencies are called covers of each other. Every set of functional dependencies has a canonical cover.
"Yes" Canonical cover is unique.

6. Explain multi-valued dependency and join dependency. **[WBUT 2008]**

Answer:

Formally, the multivalued dependency $X \rightarrow\rightarrow Y$ is said to hold for a relation $R(X, Y, Z)$ if for a given set of value (set of values if X is more than one attribute) for attributes X, there is a set of (zero or more) associated values for the set of attributes Y and the Y values depend only on X values and have no dependence on the set of attributes Z.

Now, more formally, $X \rightarrow\rightarrow Y$ is said to hold for $R(X, Y, Z)$ if $t1$ and $t2$ are two tuples in R that have the same values for attributes X and therefore with $t1[x] = t2[x]$ then R also contains tuples $t3$ and $t4$ (not necessarily distinct) such that

t1[x] = t2[x] = t3[x] = t4[x]
t3[Y] = t1[Y] and t3[Z] = t2[Z]
t4[Y] = t2[Y] and t4[Z] = t1[Z]

In other words if t1 and t2 are given by
t1 = [X, Y1, Z1], and
t2 = [X, Y2, Z2]
then there must be tuples t3 and t4 such that
t3 = [X, Y1, Z2], and
t4 = [X, Y2, Z1]

Fourth normal form

e.g.,

List the non-trivial multi-valued dependencies

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c1

The FDs and MVDs are: $A \rightarrow B, A \rightarrow\rightarrow C, B \rightarrow\rightarrow C$

A join dependency *{A, B, … Z} on R is implied by the candidate key(s) of R if and only if each of A, B, …, Z is a superkey for R.

Let us consider an example involving agents, companies, and products. If agents represent companies, companies make products, and agents sell products, then we might want to keep a record of which agent sells which product for which company. This information could be kept in one record type with three fields:

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	GM	truck

7. Define 2nd NF, 3rd NF and BCNF with example. **[WBUT 2010, 2012]**

Answer:

Refer to Question No. 1(b) and 2 of Short Answer Type Questions.

8. What do you mean by Data Dependency? What do you mean by redundant data? **[WBUT 2011]**

Answer:

1st Part:

Refer to Question No. 12 of Short Answer Type Questions.

2nd Part:

Redundancy is attained when the same data values are stored more than once in a table, or when the same values are stored in more than one table. One of the biggest disadvantages of data redundancy is that it increases the size of the database unnecessarily.

9. What is lossless decomposition? **[WBUT 2011, 2015]**

Answer:

Refer to Question No. 1(1st Part) of Long Answer Type Questions.

10. Consider the following tables with their functional dependencies –
Professor (Professor_code, Department, Head_of_dept, Percent_time)
(Department, Professor_code) → (Head_of_dept, Percent_time)
Department → (Head_of_dept)
(Head_of_dept, Professor_code) → (Department, Percent_time)
It is assumed that –
i) A professor can work in more than one department
ii) The percentage of the time he spends in each department is given
iii) Each department has only one head_of_dept.
Normalize the table up to BCNF. **[WBUT 2012, 2017]**

Answer:

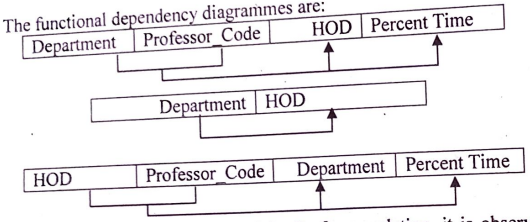
It is assumed that for the relation
PROFESSOR(Prof-code, department, Head-of-dept, Parent-time)
A professor can work in more than one department
The percentage of the time spend in each dept is given
Each department has only one HOD

The sample data and the FDs are presented below:

Prof- Code	Department	Head-of-dept	Parent-time
T1	CSE	BASU	50
T1	ECE	DUTTA	50
T2	MCA	SARKAR	25
T2	CSE	BASU	75
T3	ECE	DUTTA	100

Relation: **PROFESSOR**

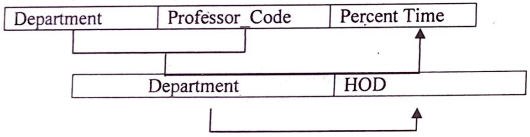
The functional dependency diagrammes are:



From the dependency diagram of the Professor relation, it is observed that the relation has more than one possible key. The composite keys have common attributes. If an attribute of a composite key is dependent on an attribute of the other composite key, then we call the normalized relation is in BCNF. The dependency diagram is clearly showing that the attribute HOD is a component of the key in the first dependency diagram and a non key attribute in the second dependency diagram.

Though the relation is in 3NF but still anomalies exists. Like the names of Dept. and HOD are duplicated. If professor T2 resigns, **row 3** and **row 4** both will be deleted and as a result Prof : **Sarkar** will be deleted.

Which can be removed by creating a new relation for department and HOD and deleting the HOD from the professor relation. The revised FDs are:



Dependency diagram of Professor relation

The sample data in the normalized relations are:

Prof- Code	Department	Parent-time
T1	CSE	50
T1	ECE	50
T2	MCA	25
T2	CSE	75
T3	ECE	100

Relation: **PROFESSOR**

Department	Head-of-dept
CSE	BASU
ECE	DUTTA
MCA	SARKAR

Relation: **DEPARTMENT**

Normalized Professor relation in BCNF

11. What is spurious tuple? How spurious tuples are generated? [WBUT 2012]

Answer:

A **join dependency (JD)**, denoted by JD (R_1, R_2, ..., R_n) is satisfied by a relation R over the union of the sets of attributes R_1 U R_2...U Rn iff the joins of the projections of R onto R_i's equals R. i.e. every legal instance r of R should have a lossless join decomposition into R_1,R_2..Rn such that the following holds:

$\mathbf{R_1(r) \bowtie R_2(r) \bowtie ..... \bowtie R_n(r)) = R}$

e.g.,

Let us consider an example: Supplier-item-project, Where the following MVDs holds S# ***→→I#, I#→→P and S#→→P#***

S#	I#	P#
S1	I1	P2
S1	I2	P1
S2	I1	P1
S1	I1	P1

We make the following decomposition:

SI

S#	I#
S1	I1
S1	I2
S2	I1

IP

P#	I#
P2	I1
P1	I2
P1	I1

SP

S#	P#
S1	P2
S1	P1
S2	P1

NOW IF WE JOIN SI AND IP OVER I#

S#	I#	P#
S1	I1	P2
S1	I1	P1
S1	I2	P1
S2	I1	P1
S2	I1	P2

Spurious tuple

So we observe that joining of this two relation produces a extra tuple which we call Spurious tuple.

12. Distinguish between logical and physical data dependency. **[WBUT 2012]**

Answer:

Physical data independence

The ability to modify the physical schema without re-writing the application programs is known as physical independence. Modifications at the physical level are occasionally necessary to improve performance of the application system.

Logical data independence

The ability to modify the logical schema without causing application programs to be rewritten. Modifications at the logical level are necessary whenever the logical structure of the database is altered (for example, when money-market accounts are added to a banking system).

Logical data independence is more difficult to achieve than is physical data independence, since application programs are heavily dependent on the logical structure of the data that they access.

The concept of data independence is similar in many respects to the concept of abstract data types in modern programming languages. Both hide implementation details from the users, to allow users to concentrate on the general structure, rather than on low-level implementation details.

13. $R(A,B,C,D,E)$ and $A \rightarrow BC, B \rightarrow E, CE \rightarrow D$ in R. **Find the candidate key for R.** **[WBUT 2013]**

Answer:

A→BC [using B→E] implies A→BCE [using CE→D] implies A→BCDE

[A is a candidate key]

14. How does BCNF differ from 3NF? Why is it considered stronger than 3NF? **[WBUT 2015]**

OR,

Proof with an example that a relation in BCNF is in 3NF, but the converse is not true. **[WBUT 2017]**

OR,

Explain with example why BCNF is stricter than 3NF? **[WBUT 2018]**

Answer:

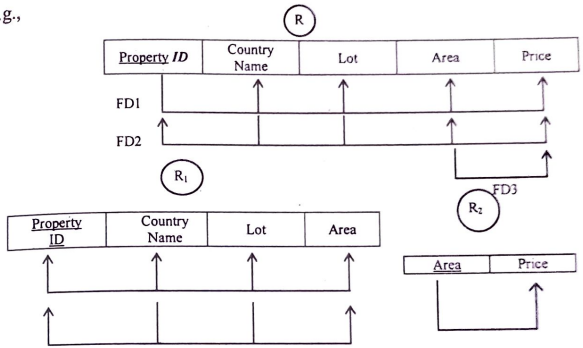
BCNF is Stronger than 3NF

A relation schema R is in Third Normal form if, whenever a nontrivial functional dependency X→A holds in R either

(a) X is a super key of R, or

(b) A is a prime attribute of R

e.g.,



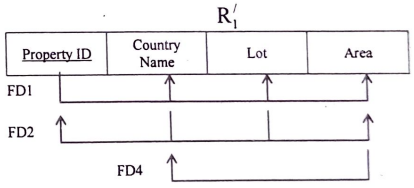
In the above relation FD3 violates 3NF because Area is not a super key and price is not a prime attribute.

After decomposing relation R into R1 & R2 both R1 & R2 are 3rd Normal form.

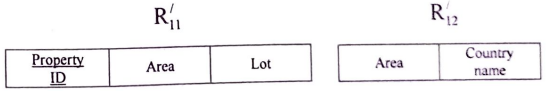
According to definition of BCNF

A relation schema R is in BCNF if whenever a nontrivial functional dependency X→A holds in R then X is a super key of R. The only difference between 3NF and BCNF is that 3NF allows A to be prime is absent from BCNF.

Suppose in the relation R1 lot sizes in country India 0.5, 0.6, 0.7 ... 1.0 acres where as lot sizes in Australia are restricted to 1.1, 1.2, 1.3 ... 2.0 acres so in the relation R1 we will have an additional functional dependency FD4:Area→country name.



Relation R′1 is in 3rd NF but not in BCNF



The above two Relation R″11 and R″12 after decomposing R′1 is in BCNF.

From the above examples we can conclude that any relation that is in BCNF is above the 3NF but the relation in 3NF necessarily in BCNF. Hence BCNF is stronger than 3NF.

15. Discuss the different database anomalies. **[WBUT 2015]**

Answer:

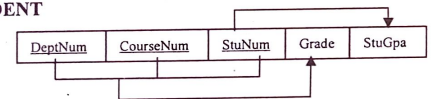
An anomaly is a variation with respect to a database maintenance. In databases when **update, insert, delete** operations are performed, it is desirable that these operations should be in a minimum time, cost effective, predictable and efficient.

However, there are some aspect of the relation that may be difficult to maintain. In order to change the value of one attribute of an entity, ideally that change require only one tuple to be updated. When the relations are not fully normalized they exhibit update anomalies.

Let us consider the relation structure and sample records:

DeptNum	CourseNum	StuNum	Grade	StuGpa
CSE	CS101	742101	A	8.04
CSE	CS101	662771	E	8.75
CSE	EC222	662771	E	8.75
CSE	EE101	742101	A	8.04

Relation: STUDENT



Functional dependency diagram

The relation STUDENT keeps the information about the student enrolled in the courses, the grade assigned for the course, and the student's overall grade point average. The functional dependency diagram above shows relations among the attributes:

Update Anomaly:

What if the student's stuGpa changes? We then need to change **stuGpa** in several records. We call this type of difficulty as an **update anomaly** – the simple change of a student's stuGpa affects, not just one record, but potentially several records in the database. The update operation is more complex than necessary, and this means it is more expensive to do, resulting in slower performance.

Delete Anomaly:

Let us assume that the above relation STUDENT is the only information about stuGpa of a student and for stuNum equals to **'662771'**, the stuGpa is required to be deleted. What happens to the student's stuGpa? The delete operation has to be executed two times and both the course information EC222 and CS101 will be deleted from the relation. We refer the case as **deletion anomaly**.

Insertion Anomaly:

Suppose we need to add a new student and assume a new student's GPA is 0. How do we add this information? Before we could add a row to this relation, we need to enrol the student into a particular course. i.e. course number is a must. This type of inefficient database design is referred to as Insertion Anomaly.

e.g.,

Name	Course	Phone#	Major	Prof	Grade
X1	353	2556-1818	CSE	P1	O
X 2	329	2592-0584	IT	P2	B
X1	328	2556-1818	CSE	P3	B
X3	456	2563-3193	ECE	P4	E
X1	492	2556-1818	CSE	P8	D
X5	379	2357-5684	Eng.	P9	E

Relation: STUDENT

Normalization is a process by which such anomalies can be removed.

16. What is meant by spurious tuple? **[WBUT 2018]**

Consider the following relational schema:

r(A, B, C, D, E, F, G)

The following functional dependency is held in the relation

A –> B ; B –> C ; C –> D, E ; F –> G

Maintaining the functional dependency the relation r is broken as follows:

r1(A, B) ; r2(B, C) ; r3 (C, D, E) ; r4 (F, G)

Verify whether this decomposition is lossless or not.

Answer:

1st Part:

Refer to Question No. 11 (1st Part) of Short Answer Type Questions.

2nd Part:

To check for lossless join decomposition using FD set, following conditions must hold:

1. Union of Attributes of R1, R2, R3 and R4 must be equal to attribute of R.
2. Intersection of attributes of R1, R2, R3 and R4 must not be NULL.
3. Common attribute must be a key for at least one relation (R1 or R2 or R3 or R4).

Now, for the given relation r(A, B, C, D, E, F, G), the first condition holds true means that Union of attributes of r1, r2, r3 and r4 is equal to attribute of r.

But the second and third condition do not satisfy because there is no common attribute in r4 with rest of the relations. So the decomposition is not lossless.

17. Consider the following relation: **[WBUT 2018]**

$$r(R) = \{A, B, C, D, E, F, G, H, I\}$$

Functional dependency is given below:

$$F = \{A \rightarrow B,\ C \rightarrow DE,\ F \rightarrow G,\ B \rightarrow GH,\ AF \rightarrow C,\ E \rightarrow I\}$$

Determine the current normal form of the given relation. Decompose it up to BCNF.

Answer:

A or F cannot be derived from any other attribute. So there will be 1 candidate key $\{AF\}$ and another will be $\{A, F, C\}$.

$(AF)^+ = \{A, B, C, D, E, F, G, H, I\}$, $(AFC)^+ = \{A, B, C, D, E, F, G, H, I\}$

Now, we know that, prime attributes are those which are part of candidate key. In this care $\{A, F, C\}$ are prime attribute and rest of the attributes are non-prime attribute.

Now, the relation R is in 1st Normal form as the relational DBMS does not allow multivalued or composite attribute.

But the relation is not in 2NF, because there are partial dependencies present in the relation.

In $A \rightarrow B$, A is a proper subset of prime attribute and B is a non-prime attribute.

Similarly for $F \rightarrow G$ partial dependency exists.

∴ The highest normal form of the relation is 1NF.

Decomposition to 2NF:

By the definition, to be in 2NF, all the partial dependencies should be removed.

Therefore, $R_1 = \{ABGH\}$

$R_2 = \{CDEI\}$

$R_3 = \{FG\}$

$R_4 = \{AFC\}$

Decomposition to 3NF:

By the definition to be in 3NF, all the transitive dependencies should be removed.

Therefore, $R_{11} = \{AB\}$, $R_{12} = \{BGH\}$

$R_{21} = \{CDE\}$, $R_{22} = \{EI\}$

$R_3 = \{FG\}$

$R_4 = \{AFC\}$

This relation is also in BCNF, because there is no prime to prime dependency. In case of $AF \rightarrow C$, $\{AF\}$ can be treated as superkey, since $\{AF\}$ alone can find all other attributes. So this does not violate the condition of BCNF.

18. Compare between 3NF and BCNF with example. **[MODEL QUESTION]**

Answer:

3NF	BCNF
i) In case of 3NF all the determinant cannot be candidate key.	i) In case of BCNF all the determinant must be candidate key.
ii) 3NF is less stricter than BCNF.	ii) BCNF is more stricter than 3NF.
iii) 3NF is normal form in which the table is in 2NF and every non-prime attribute is non-transitively dependent on every key in the table.	iii) BCNF is a normal form in which for every one of a table's non-trivial functional dependencies, is a super key.

19. Discuss the advantages of centralized and distributed databases. **[MODEL QUESTION]**

Answer:

Centralized database

Early data processing systems worked with a single centralized database.

Smaller local systems started to store local databases as well as do local data processing.

Move to central database distributed to local processors as long as:

Accurate updating of data

Integrity of Data

Sharing of Data

Central Administrative controls

Could all be assured and maintained.

For advantages of distributed database:

1. Enriched Modeling Capabilities.
2. Extensibility.
3. Removal of Impedance Mismatch.
4. More Expressive Query Language.
5. Support for Schema Evolution.
6. Support for Long Duration Transactions.
7. Applicability to Advanced Database applications.
8. Improved Performance

20. A table Supply (sno, city, status, pno, qty) is defined with the FD'S (i) sno → city (ii) city → status (iii) (sno, pno) → qty.
1. Find the key for the schema.
2. Reduce it to 3NF. **[MODEL QUESTION]**

Answer:

Sno and Pno is the combined primary key of the given FD's

Non-key attributes are not mutually independent (city → status).

Non-key attributes are not fully functionally dependent on the primary key (i.e., status and city are dependent on just part of the key, namely supplier no).

The 3NF will be:

Supply (sno, pno, qty) with FD (sno, pno) → qty

City(city, status) with FD city → status

Serial(sno , city) with sno → city

21. a) Define 'Project-join normal form' & 'Domain-key normal form' with suitable example. **[MODEL QUESTION]**

Answer:

Project-join normal form (PJ/NF) or Fifth normal form **(5NF)** is a level of database normalization designed to reduce redundancy in relational databases recording multi-valued facts by isolating semantically related multiple relationships. A table is said to be in the 5NF if and only if every join dependency in it is implied by the candidate keys.

A join dependency *{A, B, ... Z} on R is implied by the candidate key(s) of R if and only if each of A, B, ..., Z is a super key for R.

Let us consider an example involving agents, companies, and products. If agents represent companies, companies make products, and agents sell products, then we might want to keep a record of which agent sells which product for which company. This information could be kept in one record type with three fields:

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	GM	truck

This form is necessary in the general case. For example, although agent Smith sells cars made by Ford and trucks made by GM, he does not sell Ford trucks or GM cars. Thus we need the combination of three fields to know which combinations are valid and which are not.

But suppose that a certain rule was in effect: if an agent sells a certain product, and he represents a company making that product, then he sells that product for that company.

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	Ford	truck
Smith	GM	car
Smith	GM	truck
Jones	Ford	car

In this case, it turns out that we can reconstruct all the true facts from a normalized form consisting of three separate record types, each containing two fields:

AGENT	COMPANY
Smith	Ford
Smith	GM
Jones	Ford

COMPANY	PRODUCT
Ford	car
Ford	truck
GM	car
GM	truck

AGENT	PRODUCT
Smith	car
Smith	truck
Jones	car

These three record types are in fifth normal form, whereas the corresponding three-field record shown previously is not.

Domain/key normal form (DKNF) is a normal form used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints.

A domain constraint specifies the permissible values for a given attribute, while a key constraint specifies the attributes that uniquely identify a row in a given table.

The domain/key normal form is achieved when every constraint on the relation is a logical consequence of the definition of keys and domains, and enforcing key and domain restraints and conditions causes all constraints to be met. Thus, it avoids all non-temporal anomalies.

In the following relation

CUSTOMER (CUST_ID, ACCT_LEVEL, ACCT_TYPE, FEE) where CUST_ID = Customer identifier, ACCT_TYPE = Account type, FEE = FEE charged based on type of account.

CUST_ID functionally determines ACCT_LEVEL, ACCT_TYPE, and FEE so CUST_ID is the key. Our business rules in this relation place domain constraints on each of the attributes and we know that ACCT_TYPE => FEE. In accordance with the rules for DK/NF, we need to make the functional dependency ACCT_TYPE => FEE a logical consequence of keys. If ACCT_TYPE were a key attribute, ACCT_TYPE => FEE would be a logical consequence of the key.

ACCT_TYPE cannot be a key in Customer because more than one customer can have the same account type. But it can be a key of its own relation. So, we define the following domain definition, relations, and attributes:

DOMAIN DEFINITIONS CUST_ID in DDDDDD, where D is a decimal digit >0<999999 ACCT_LEVEL in ("PERS", "CML", "INTB") ACCT_TYPE in CHAR (3) FEE in DEC (4)RELATION AND KEY DEFINITIONS CUSTOMER (CUST_ID, ACCT_LEVEL, ACCT_TYPE) KEY: CUST_ID ACCT_FEE (ACCT_TYPE, FEE)KEY: ACCT_TYPE.

b) If 'D' be the set of all functional and multivalued dependencies then write the rules to compute the'D+ (Closureof D). **[MODEL QUESTION]**

Answer:

We can find all of D+ by applying Armstrong's Axioms:

- if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ **(reflexivity)**
- if $\alpha \rightarrow \beta$, then $\gamma\ \alpha \rightarrow \gamma\ \beta$ **(augmentation)**
- if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(transitivity)**

22. Explain the following terms 'Fully functional dependency' and 'non transitive dependency' with examples. **[MODEL QUESTION]**

Answer:

An attribute is **fully functionally** dependent on a set of attributes X if it is

i) functionally dependent on X, and

ii) not functionally dependent on any proper subset of X. {Employee Address} has a functional dependency on {Employee ID, Skill}, but not a *full* functional dependency, because it is also dependent on {Employee ID}.

A **non-transitive dependency** is one in which an attribute is dependent on a non-key attribute in that table. There is no dependency between non-prime key attributes.

Consider a relation student having prime key attribute (Roll) and non-prime key attributes(Name, semester, hostel) where for each semester there is a different hostel. Here, Name is non-transitively dependent on Roll.

Long Answer Type Questions

1. What do you mean by lossless decomposition and preservation of dependencies? Consider the relation R= {RN, POR, PI, PN, DATE, MA, MT} and Functional Dependencies: {RN —> POR: P1 -> PN, RN; P1, DATE -> MA, MT, POR}. In which normal form this belongs to? Decompose the relation so that it can be belong to 3NF. **[WBUT 2007]**

Answer:

Lossless Decomposition

Let R having the decompositions R1 and R2 with given set of FD's say F, then R1and R2 to become lossless then at least one of the following functional dependencies should hold in F+

$R1 \cap R2 \longrightarrow R1$

$R1 \cap R2 \longrightarrow R2$

Dependency Preservation

A decomposition is dependency preserving if the closure of

$F' = (F1 \cup F2.....) = F+$

$RN \longrightarrow POR$

$PI \longrightarrow PN$ $\quad$ $PI \longrightarrow RN$

$PI \longrightarrow PN, RN, POR$ .... (1)

$PI, DATE \longrightarrow MA, MT, POR$ .... (2)

From (1) and (2), it can be inferred that the presence of the attribute POR in (1) and in (2) shows that a partial dependency exist. Thus, the following dependencies are in 2nd NF. In order to convert it in to 3nf the revised FD's should be

$PI \longrightarrow PN, RN, POR$ .... (1)

$PI, DATE \longrightarrow MA, MT$ .... (2)

2. a) Describe Inference Rules for functional dependencies. **[WBUT 2008]**

Answer:

For any given relation, there can be many FDs. Let us denote F as the set of FDs of a given relation R. There can be many other FDs that can be derived (or, inferred) from F.

1. Reflexive Rule: If $Y \subseteq X$, then $X \rightarrow Y$

If Y is a subset of attributes of set X, then $X \rightarrow Y$

2. Augmentation Rule: If $X \rightarrow Y$, then $XZ \rightarrow YZ$

For a pair of tuples t1 and t2,

If t1[X] = t2[X] then it implies that t1[Y] = t2[Y]

And it can also be inferred that

If t1[XZ] = t2[XZ] then it implies that t1[YZ] = t2[YZ]

3. Transitive Rule: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

To generalize the all possible FDs the above three rules are sufficient. Rule four, five and rule six can be derived.

4. Decomposition: If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

5. Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

6. **Pseudo transitivity rule:** If $X \rightarrow Y$ holds and $ZY \rightarrow W$ holds then $XZ \rightarrow W$ holds

b) Write down the attribute closure algorithm. Compute the closure of following set F of functional dependencies for the given relational schema
R = (A, B, C, D, E)
The FD's: A → BC, CD → E, B → D, E → A. **[WBUT 2008]**

Answer:

- To test whether a set of attributes α is a superkey, we need to find the set of attributes functionally determined by α.
- Let α be a set of attributes. We call the set of attributes determined by α under a set F of functional dependencies the **closure** of α under F, denoted α^+.
- The following algorithm computes α^+:

```
Result: = α
        While (changes to result) do
              For each functional dependency β→γ
                                        In F do
                        Begin
                              If β⊆ result
                        Then result: = result ∪γ ;
                  End
```

Let the given dependencies are F then the closure is denoted by F+

F+={ABC}, Using B →D

F+ <new> = {ABCD}, using CD → E

F+ <new> = {ABCDE}

3. What is the highest NF of each of the following relations?
i) R1 (J, K, L) with FDs are J → K, J → L, K → L
ii) R2 (J, K, L, M) with FDs are J → KL, LM → K **[WBUT 2009, 2012, 2014]**

Answer:
i) The relation R=1 is in 2NF as there is no partial dependency. But there is transitive dependency.
So, the highest NF is 2NF.

ii) The highest NF of the relation R2 is 1NF.

4. a) What do you mean by closure?
b) Suppose that we decompose the schema, R = (A, B, C, D, E) into (A, B, C) and (A, D, E). Show that this decomposition is loss less decomposition, if the following set F of FDs holds-
A → BC
CD → E
B → D
E → A **[WBUT 2009]**

Answer:
a) The closure of a set F of functional dependencies is the set of all functional dependencies logically implied by F. We denote the closure of F by $F+$.

b)

	A	B	C	D	E
R1	a1	a2	a3	b14	b15
R2	a1	b22	b23	a4	a5

The initial partitions are organized in a [2 x 5] matrix. Using the given FD's, the objective is to transform all elements of any one of the rows with 'a' to infer that the decomposition is lossless.

Using the relation A→BC, the transformed matrix becomes

	A	B	C	D	E
R1	a1	a2	a3	b14	b15
R2	a1	~~b22~~	~~b23~~	a4	a5
		a2	a3		

Now R2 is transformed in terms of 'a' only and hence the decomposition is loss less.

5. Compute the closure of the set F of FDs for the relation schema,
R = (A, B, C, D, E)
A → BC
CD → E
B → D
E → A
List the candidate keys for R. **[WBUT 2011, 2015]**

Answer:
R = {A,B,C,D,E}
F, the set of functional dependencies
A→BC, CD→E, B→D, E→A
Compute the closure for each β in β → γ in F

Closure for A

Iteration	result	using
1	A	
2	ABC	A→BC
3	ABCD	B→D
4	ABCDE	CD→E
5	ABCDE	

A^+ = ABCDE, Hence A is a super key

Closure for CD

Iteration	result	using
1	CD	
2	CDE	CD→E
3	ACDE	E→A
4	ABCDE	A→BC
5	ABCDE	

CD^+ = ABCDE, Hence CD is a super key

Closure for B

Iteration	result	Using
1	B	
2	BD	B→D
3	BD	

B^+ = BD, Hence B is NOT a super key
Applying Armstrong axioms, to find alternate keys.
B→D, BC→CD (by Armstrong's augmentation rule)

Closure for BC

Iteration	result	using
1	BC	
2	BCD	BC→CD
3	BCDE	CD→E
4	ABCDE	E→A

BC^+ = ABCDE, Hence BC is a super key

Closure for E Iteration	result	using
1	E	
2	AE	E→A
3	ABCE	A→BC
4	ABCDE	B→D
5	ABCDE	

E^+ = ABCDE
A and E are minimal super keys.

To see whether CD is a minimal super key, check whether its subsets are super keys.
$C^+ = C$, $D^+ = D$

Since C and D are not super keys, CD is a minimal super key.
To see whether BC is a minimal super key, check whether its subsets are super keys.
$B^+ = BD$, $C^+ = C$
Since B and C are not super keys, BC is a minimal super key.
Since A, BC, CD, E are minimal super keys, they are the candidate keys.

6. a) What is data anomaly? Describe minimal set of FDs. **[WBUT 2011]**
b) Consider the universal relation
R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies are
F = { {A, B} – > {C}, {A} – > {D, E}, {B} – > {F}, {F} – > {G, H}, {D} – > {I, J} }.
The set represents which normal form?
Answer:
a) Data anomaly:
Refer to Question No. 15 of Short Answer Type Questions.

Minimal set of FDs:
A set E of FDs that is irreducible and is equivalent to some other set of FDs is said to be an irreducible cover for F. It can be also referred as minimal set of FDs.

b) Candidate Keys: {AB}
Minimal Cover:
AB→C, A→D, A→E, B→F, F→G, F→H, D→I, D→J
3NF

R1 ABC	A→BC
R2 ADE	A→DE
R3 BF	B→F
R4 FGH	F→GH
R5 DIJ	D→IJ

7. What do you mean by fully functional dependency? A relation R (A, B, C) having FDs – A →B, A → C, C → B. Is the relation in 2NF? Can it be decomposed to 3NF? Justify your answer. **[WBUT 2013]**
Answer:
A full functional dependency occurs when you already meet the requirements for a functional dependency and the set of attributes on the left side of the functional dependency statement cannot be reduced any farther.

Examples: E.g. "{SSN, age} → name" is a functional dependency, but it is not a full functional dependency because you can remove age from the left side of the statement without impacting the dependency relationship.

Consider the set of FD: $A \rightarrow B$ and $A \rightarrow C$ *implies* $A \rightarrow BC$. *A* is obviously a key for this relation. It is also a primary key since there are no smaller subsets of keys that hold over *R(A,B,C)*. The FD: A→*C*, *C* →*B violates* 3NF but not 2NF since:

8. Consider the universal relation:
$R = \{A, B, C, D, E, F, G, H, I, J\}$ **and the set of functional dependencies:**
$AB \rightarrow C$
$A \rightarrow DE$
$B \rightarrow F$
$F \rightarrow GH$
$D \rightarrow IJ$
For the above relation R and functional dependencies, consider the decomposition
$D = \{R1, R2, R3\}$ **where**
$R1 = \{A, B, C, D, E\}$
$R2 = \{B, F, G, H\}$
$R3 = \{D, I, J\}$
Find out whether this decomposition is lossless or lossy. **[WBUT 2013]**
Answer:

	A	B	C	D	E	F	G	H	I	J
R1	a	a	a	a	a	~~b16~~	~~b17~~	~~b18~~	~~b19~~	~~b110~~
R2	b21	a	b23	b24	b25	a	a	a	b29	b210
R3	b31	b32	b33	a	b31	b36	b37	b38	a	a

Using $B \rightarrow F$ and $F \rightarrow GH$
By Transitivity B→GH and D→IJ
We re-write the matrix as follows:

	A	B	C	D	E	F	G	H	I	J
R1	~~a~~	a	a	a	a	a	a	a	a	~~a~~
R2	b21	a	b23	b24	b25	a	a	a	b29	b210
R3	b31	b32	b33	a	b31	b36	b37	b38	a	a

The row R1 is now fully converted in terms of 'a' and hence the decomposition is loss less.

9. i) Why certain functional dependencies are called trivial functional dependencies?
ii) Use Armstrong's axioms to prove the soundness of the union rule.
iii) Compute the closure of the following set F of FDs for each relation schema R = (A, B, C, D, E).
A → BC
CD → E
B → D
E → A.
List the candidate key for R. **[WBUT 2014]**

Answer:

i) A trivial functional dependency occurs when you describe a functional dependency of an attribute on a collection of attributes that includes the original attribute. For example, "{A, B} -> B" is a trivial functional dependency, as is "{name, SSN} -> SSN". This type of functional dependency is called trivial because it can be derived from common sense. It is obvious that if you already know the value of B, then the value of B can be uniquely determined by that knowledge.

ii) Here we can use the augmentation rule to show that, if $\alpha \rightarrow \beta$, then $\alpha \rightarrow \alpha\beta$. Then apply the augmentation rule again, using $\alpha \rightarrow \gamma$, and then apply the transitivity rule.
To prove that: if $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ then $\alpha \rightarrow \beta\gamma$,

we derive: $\alpha \rightarrow \beta$	given
$\alpha\alpha \rightarrow \alpha\beta$	augmentation rule
$\alpha \rightarrow \alpha\beta$	union of identical sets
$\alpha \rightarrow \gamma$ given $\alpha\beta \rightarrow \gamma\beta$	augmentation rule
$\alpha \rightarrow \beta\gamma$	transitivity rule and set union commutativity

iii) Compute the closure of the following set F of functional dependencies for relation schema R = (A, B, C, D, E).
$A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$.

List the candidate keys for R. Note: It is not reasonable to expect students to enumerate all of F +. Some shorthand representation of the result should be acceptable as long as the nontrivial members of F + are found.
Starting with $A \rightarrow BC$, we can conclude: $A \rightarrow B$ and $A \rightarrow C$.
Since $A \rightarrow B$ and $B \rightarrow D$, $A \rightarrow D$ (decomposition, transitive) Since $A \rightarrow CD$ and $CD \rightarrow E$, $A \rightarrow E$ (union, decomposition, transitive) Since $A \rightarrow A$, we have (reflexive)
$A \rightarrow ABCDE$ from the above steps (union)
Since $E \rightarrow A$, $E \rightarrow ABCDE$ (transitive)
Since $CD \rightarrow E$, $CD \rightarrow ABCDE$ (transitive)
Since $B \rightarrow D$ and $BC \rightarrow CD$, $BC \rightarrow ABCDE$ (augmentative, transitive)
Also, $C \rightarrow C$, $D \rightarrow D$, $BD \rightarrow D$, etc.

Therefore, any functional dependency with A, E, BC, or CD on the left hand side of the arrow is in F +, no matter which other attributes appear in the FD.
Allow * to represent any set of attributes in R, then F + is $BD \rightarrow B$, $BD \rightarrow D$, $C \rightarrow C$, $D \rightarrow D$, $BD \rightarrow BD$, $B \rightarrow D$, $B \rightarrow B$, $B \rightarrow BD$, and all FDs of the form $A * \rightarrow \alpha$, $BC * \rightarrow \alpha$, $CD * \rightarrow \alpha$, $E * \rightarrow \alpha$ where α is any subset of {A, B, C, D, E}. The candidate keys are A, BC, CD, and E.

10. For relation R(L, M, N, O, P) the following FD's hold: **[WBUT 2017]**
M→O, NO→P, P→L, L→MN
R is decomposed into R1 = (L, M, N, P) and R2 = (M, O)
i) Is the above decomposition lossless-join decomposition? Explain
ii) Is the above decomposition dependency preserving? Explain

Answer:

i) $R1 \cap R2 = \{M\}$
$M \rightarrow O$. so it gives $R1 \cap R2 = \{M, O\}$.
But without N, nothing else can be obtained. Thus it is a lossy decomposition. It is said to be lossless if and only if $R1 \cap R2 = R$.

ii) A decomposition is dependency preserving when $R1 \cup R2 = R$.
Considering R1 = (L, M, N, P):
$L \rightarrow MN$
$M \rightarrow O$
$NO \rightarrow P$
Thus $(R1)+ = R$
Considering R2 = (M, O):
$M \rightarrow O$
Now, $(R1 \cup R2)+ = R$
Thus it is a dependency preserving decomposition.

11. Explain different types of database anomalies and how normalization removes them. **[WBUT 2018]**

Answer:

1st Part: ***Refer to Question No. 15 of Short Answer Type Questions***

2nd Part: ***Refer to Question No. 1(a) & (b) of Short Answer Type Questions.***

12. Write short notes on the following:
a) Functional Dependency **[WBUT 2014]**
b) 4NF **[MODEL QUESTION]**

Answer:

a) Functional Dependency:
Functional dependency is a relationship that exists when one attribute uniquely determines another attribute.

If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X→Y, which specifies Y is functionally dependent on X. Here X is a determinant set and Y is a dependent attribute. Each value of X is associated precisely with one Y value.

Functional dependency in a database serves as a constraint between two sets of attributes. Defining functional dependency is an important part of relational database design and contributes to aspect normalization.

A functional dependency is trivial if Y is a subset of X. In a table with attributes of employee name and Social Security number (SSN), employee name is functionally dependent on SSN because the SSN is unique for individual names. An SSN identifies the employee specifically, but an employee name cannot distinguish the SSN because more than one employee could have the same name.

Functional dependency defines Boyce-Codd normal form and third normal form. This preserves dependency between attributes, eliminating the repetition of information. Functional dependency is related to a candidate key, which uniquely identifies a tuple and determines the value of all other attributes in the relation. In some cases, functionally dependent sets are irreducible if:

The right-hand set of functional dependency holds only one attribute

The left-hand set of functional dependency cannot be reduced, since this may change the entire content of the set

Reducing any of the existing functional dependency might change the content of the set

An important property of a functional dependency is Armstrong's axiom, which is used in database normalization. In a relation, R, with three attributes (X, Y, Z) Armstrong's axiom holds strong if the following conditions are satisfied:

Axiom of Transivity: If X→Y and Y→Z, then X→Z

Axiom of Reflexivity (Subset Property): If Y is a subset of X, then X→Y

Axiom of Augmentation: If X→Y, then XZ→YZ

Related Terms

b) 4NF:

Fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce-Codd Normal Form (BCNF). It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

FILE ORGANIZATION & QUERY OPTIMIZATION

Multiple Choice Type Questions

1. The first phase of query processing method is more efficient? [WBUT 2008, 2015]

a) decomposition **b) restructuring** **c) analysis** **d) none of these**

Answer: (d)

2. Which of the following query processing method is more efficient? [WBUT 2008]

a) Pipe lining **b) Materialization** **c) Tunneling** **d) None of these**

Answer: (d)

3. One of the cause of the failure of file system is [WBUT 2009]

a) Data availability **b) Fixed records**

c) Sequential records **d) Lack of security**

Answer: (d)

4. Files of unordered records are called [WBUT 2013]

a) heap files **b) stored files**

c) hash files **d) None of these**

Answer: (a)

5. In which phase of Query processing multiple plans are considered [MODEL QUESTION]

a) parsing **b) translation** **c) optimitation** **d) evaluation**

Answer: (c)

Short Answer Type Questions

1. What is Query Optimization? [WBUT 2005]

Answer:

Having long-running queries not only consumes system resources that makes the server and application run slowly, but also may lead to table locking and data corruption issues. So, query optimization(QO) becomes an important task. As there are many equivalent transformations of same high-level query, aim of QO is to choose one that minimizes resource usage. In general it also reduces total execution time of query. Since the problem is computationally intractable with large number of relations, so strategy adopted is reduced to finding near optimum solution.

2. Explain the disadvantages of file oriented approach. **[WBUT 2013]**

Answer:

- **Data Storage** - creates excessive storage costs of paper documents and/or magnetic form
- **Data Updating** - any changes or additions must be performed multiple times
- **Currency of Information** - potential problem of failing to update all affected files
- **Task-Data Dependency** - user's inability to obtain additional information as his or her needs change.

3. Compare static and dynamic query optimization techniques. **[MODEL QUESTION]**

Answer:

Static Query Optimization Technique	Dynamic Query Optimization Technique
i) Static query optimization utilizes information about the relations to be operated on that is available prior to execution time i.e. compilation time.	i) Dynamic query optimization involves optimization of a query as part of its execution during execution stage.
ii) The access plan is not optimized further.	ii) The access plan is further optimized by optimizing the order in which certain operations are executed.
iii) Selects an access plan with the minimum projected cost.	iii) Cost is maximized in this technique.

Long Answer Type Questions

1. Explain the query optimization technique with relevant examples. **[WBUT 2012, 2015]**

Answer:

Query Optimization:

Refer to Question No. 1 of Short Answer Type Questions.

Overview of Query Processing

The main steps in processing a high-level query are illustrated in figure below.

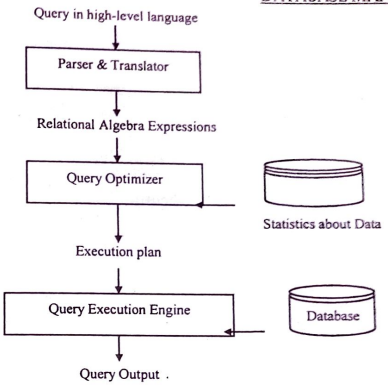


Fig: Steps in query processing process

The functions of Query Parser is parsing and translating a given high-level language query into its immediate form such as relational algebra expressions. The parser need to check for the syntax of the query and also check for the semantic of the query (it means verifying the relation names, the attribute names in the query are the names of relations and attributes in the database). A parse-tree of the query is constructed and then translated into relational algebra expression.

A relational algebra expression of a query specifies only partially how to evaluate a query, there are several ways to evaluate an relational algebra expression. For example, consider the query:

SELECT Salary FROM EMPLOYEE WHERE Salary >= 50,000;

The possible relational algebra expressions for this query are:

1) $\Pi_{Salary}(\sigma_{Salary>=50000}(EMPLOYEE))$
2) $\sigma_{Salary>=50000}(\Pi_{Salary}EMPLOYEE)$

Further, each relational algebra operation can be executed using various algorithms. For example, to implement the preceding selection, we can do a linear search in the EMPLOYEE file to retrieve the tuples with Salary >= 50000. However, if an index available on the Salary attribute, we can use the index to locate the tuples. Different algorithms might have different cost.

Thus, in order to specify fully how to evaluate a query, the system is responsible for constructing a query execution plan which made up of the relational algebra expression and the detailed algorithms to evaluate each operation in that expression. Moreover, the selected plan should minimize the cost of query evaluation. The process of choosing a

suitable query execution plan is known as query optimization This process is perform by Query Optimizer.

One aspect of optimization occurs at relational algebra level. The system attempt to find an expression that is equivalent to the given expression but that is more efficient to execute. The other aspect involves the selection of a detail strategy for processing the query, this relates to choosing the processing algorithm, choosing the indices to use and so on.

Once the query plan is chosen, the Query Execution Engine lastly take the plan, executes that plan and return the answer of the query.

2. Write short notes on the following:

a) Indexed sequential file organization **[WBUT 2006, 2007]**

b) Query optimization technique **[WBUT 2007, 2010, 2011, 2016, 2018]**

c) Query processing and optimization **[WBUT 2014]**

d) Database approach and the file based approach **[WBUT 2017]**

Answer:

a) Indexed Sequential Organization:

It facilitates direct access to a sequential file organization. Indexed- sequential file organization provides facilities for accessing records both sequentially and directly on the fields on which it is indexed.

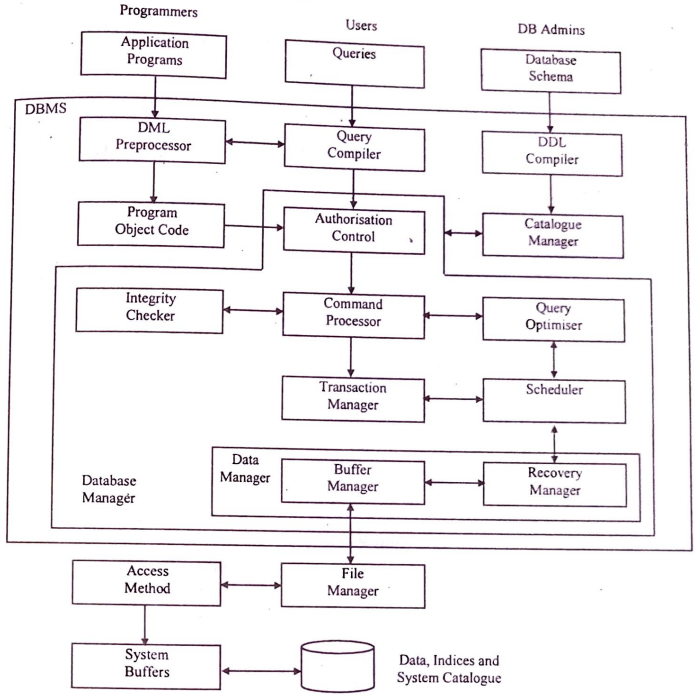
- Records are stored by primary key or any orderly field
- An index of record locations is stored on the disk.
- A separate Index structure is maintained
- An index is maintained that points to the location on disk where the record is found.
- Insertion is made at the end of the primary(data) file
- Corresponding record is inserted in the index file in proper order

Operation on the data file as well as a set of index files can be made. The index files are small in size. Insert operation takes constant time for the data itself plus $\log_2 n$ for the index, Select, Update, Delete operations take $\log_2 n$ lookup on the index followed by constant time to access data record. As the index files are very short in size the value of $\log_2 n$ becomes negligible small. This value is then use to search the physical record directly.

b) Query optimization technique:

Refer to Question No. 1 of Long Answer Type Questions.

c) Query processing and optimization:



Basic Query Processing Steps ...

- Query parsing and translation (query compiler)
  - check the syntax (e.g. SQL for relational DBMS)
  - verify that the mentioned relations do exist and replace views
  - transform the SQL query to a query plan represented by a relational algebra expression (for relational DBMS)
  - different possible relational algebra expressions for a single query
- Query optimization (query optimiser)
  - transform the initial query plan into the best possible query plan based on the given data set

- specify the execution of single query plan operations (evaluation primitives), e.g., which algorithms and indices to be used
- the query execution plan is defined by a sequence of evaluation primitives

- Query evaluation (command processor)
  - execute the query execution plan and return the result.

d) Database approach and the file based approach:
A database management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient. Database systems are designed to manage large bodies of information. A management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.

The main characteristics of the database approach versus the file-processing approach are as follows –
Self-describing Nature of a Database System – The database system contains not only the database itself but also a complete definition or description of the database structure and constraints. In traditional file processing, data definition is the part of the application programs themselves. Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs. Whereas file-processing software can access only specific database DBMS software can access diverse database by extracting the database definitions from the catalog and then using these definitions.
Insulation between Programs and Data, and Abstraction – In traditional file processing, the structure of data files is embedded in the access programs, so any changes to the structure of a file may require changing all programs that access this file. By contrast, DBMS access programs do not require such changes. The structure of data files is stored in the DBMS catalog separately from the access programs.

INDEXING

Multiple Choice Type Questions

1. One approach for standardizing data storage is [WBUT 2006, 2007, 2011]
a) MIS
b) CODASYL specification
c) structured programming
d) data storage cannot be standardized
Answer: (b)

2. Which index is specified on the non-ordering fields of a file? [WBUT 2008, 2011]
a) Primary
b) Clustering
c) Secondary
d) None of these
Answer: (c)

3. An index on the search key is called a [WBUT 2010]
a) primary index
b) secondary index
c) multi-level index
d) all of these
Answer: (a)

4. The main goal of indexing is to [WBUT 2013, 2017]
a) search an item faster from a table
b) insert an item faster into a table
c) delete an item faster from a table
d) none of these
Answer: (a)

5. The primary key indexing techniques do not allow [WBUT 2014]
a) Sets of relations
b) Multiple attributes
c) duplicate data
d) Many to many relation
Answer: (c)

6. Which of these is not a feature of hierarchical model? [WBUT 2018]
a) Organizes the data in tree-like structure.
b) Parent node can have any number of child nodes
c) Root node does not have any parent.
d) Child node can have any number of parent nodes.
Answer: (d)

Short Answer Type Questions

1. Define dynamic hashing. [WBUT 2005, 2011]
Answer:
Dynamic hashing is a technique, where the number of buckets is not fixed but it grows and shrinks as needed. In case of static hashing the bucket size is fixed. However, a dynamic hashing starts with a single bucket and when it is full an attempt is made to insert the record into a new bucket. Extendable hashing is one form of dynamic hashing.

It splits and coalesces buckets as database size changes. It imposes some performance overhead, but space efficiency is maintained. Overhead is reasonably low as re-organization is within one bucket at a time.

2. What is multi-level index? **[WBUT 2012]**

Answer:

With multilevel indexes, the idea is to reduce the part of the index that we continue to search by a larger factor, the blocking factor of the index, where the bfr_i is greater than 2. The blocking factor of the index, bfr_i is called the **fan-out**, or fo of the multilevel index. Searching a multilevel index requires approximately $(\log_{fo} b_i)$ block accesses which is a smaller number than for a binary search if the fan-out is larger than 2. If the fan out is equal to 2, there is no difference in the number of block accesses.

The index file is called the first level of a multilevel index. It is an ordered file with a distinct value for each key value K(i). Therefore we can create a ***primary index*** for the first level. This index to the first level is called the second level of the multilevel index.

The second level is a primary index, therefore we can use block anchors, and the second level has ***one entry for each block*** in the first level index. The blocking factor for all other levels is the same as for the first level, because the size of each index entry is the same. Each entry has one field value, and one block address.

If the first level has r_1 entries, and the blocking factor (which is also the fan out) for the index is bfr_i = fo, then the first level needs $\lceil r_1/fo \rceil$ blocks, which is therefore the number of entries r_2 needed at the second level of the index.

If necessary, this process can be repeated at the second level. The third level, which is an index for the second level, has an entry for each second level block, so the number of third level entries is $r_3 = \lceil r_2/fo \rceil$.

We only require a second level only if the first level needs more than one block of disk storage, and we require a third level only if the second level requires more than one block as well.

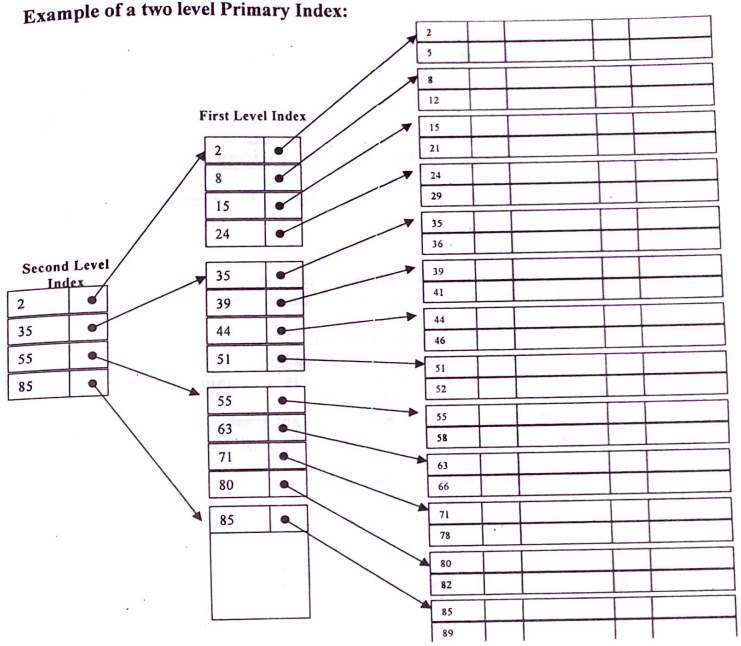
This process can be repeated until all the index entries at some level t fit in a single block. The block at the t^{th} level is the top-level index. Each level reduces the number of entries from the previous level by a factor of fo (the index fan out or blocking factor of the index).

A multilevel index with r1 first level entries will have approximately t levels, where:

$t = \lceil \log_{fo}(r_1) \rceil$.

The multilevel indexes can be used on any type of index, primary, clustering, or secondary, as long as the first level index has distinct values for K(i), and fixed length entries.

Example of a two level Primary Index:



An indexed file approach keeps a small part of each row, and some kind of "pointer" to the row's location within the data file. This allows a search to use the index, which is ordered by the index and much smaller and therefore much faster than scanning the entire data file for the indexed data.

A hashing algorithm uses some of the data in the record to compute a "hash" value. This value is a unique or at least relatively unique value. The hash value determines where the record is stored in the file.

Hashes are generally very fast. A simple algorithm will immediately determine the hash value for the record. This is both their strength, and their weakness. The strength comes from the speed and the fact that they don't need disk I/O to find anything. The weakness comes from the limitation of having only one hash value for a record. If one needs to order the list by employee number, last name, and first name, it will be problematic. Note that nothing prevents anyone from both hashing and indexing, but that adds a lot of extra I/O to the process of modifying the file.

A good hash algorithm needs to provide relatively unique hash values. If one gets too many rows that produce the same hash value, the "collisions" will drastically slow down the performance of both inserts and searches. Another requirement is that while minimizing collisions, the algorithm needs to keep the hash range small, which means that the hash output values are relatively tightly clustered.

Long Answer Type Questions

1. What is the difference between primary index and secondary index? What is hashing?
Draw the ERD:

Consider a hospital:

Patients are treated in a single ward by the doctors assigned to them. Usually each patient will be assigned a single doctor, but in rare cases they will have two.
Healthcare assistants also attend to the patients; a number of these are associated with each ward. Initially the system will be concerned solely with drug treatment. Each patient is required to take a variety of drugs a certain number of times per day and for varying lengths of time.
The system must record details concerning patient treatment and staff payment. Some staff are paid part time and doctors and card assistants work varying amounts of overtime at varying rates (subject to grade).
The system will also need to track what treatments are required for which patients and when and it should be capable of calculating the cost of treatment per week for each patient (though it is currently unclear to what use this information will be put). **[WBUT 2011]**

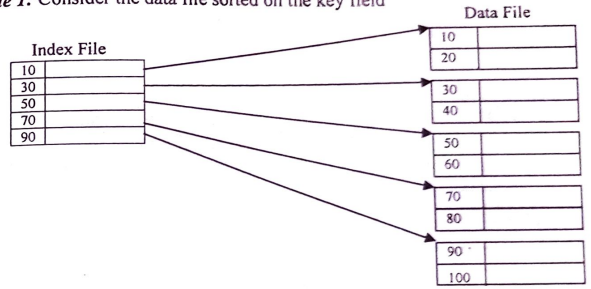
Answer:

Primary index and secondary index:

Primary Indexes

These indexes require a sequential file (i.e. the file should be sorted on the search key field). When the search key is a key of the relation, we call the index as *primary* index, and when the search key is not a key of the relation, the index is called *clustering* index. The following examples will make things clear:

Example 1: Consider the data file sorted on the key field



- Primary index requires that the ordering field of the data file have a distinct value for each record.
- Primary index is sparse
- Contains as many records as there are blocks* in the data file (there are 5 blocks in this example and each block can hold only 2 records).
- The first record in each block of the data file is called *anchor record* of the block, or simply *block anchor.*
- There can be only one primary index on a table

A dense index on the above data file will have 10 records, one for each key value, and record pointers instead of block pointers.

Secondary Indexes

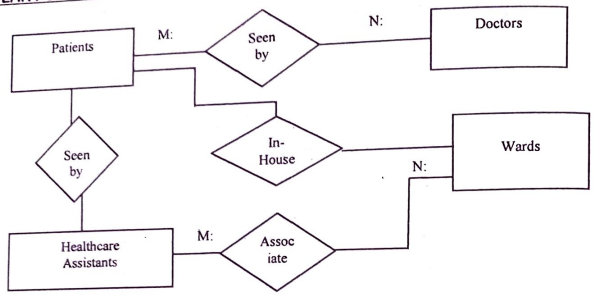
Secondary indexes do not require the data file to be sorted on the indexed field. They can be created on both key and non-key fields.

Properties of Index Types

Type of Index	Number of Index Entries	Dense or Sparse	Block Anchoring
Primary	No. of blocks in data file	Sparse	Yes
Clustering	No. of distinct index field values	Sparse	Yes/no*
Secondary (key)	Number of records in data file	Dense	No
Secondary (nonkey)	No. of records** No. of distinct index field values***	Dense Sparse	No

Hashing:

Refer to Question No. 5(a) of Long Answer Type Questions.



E-R Diagram of the Hospital

2. Describe dense and sparse indices with diagram. **[WBUT 2014]**

Answer:

Dense and Sparse Indices

1. There are Two types of ordered indices:
   Dense Index:
   - An index record appears for every search key value in file.
   - This record contains search key value and a pointer to the actual record.

   Sparse Index:
   - Index records are created only for some of the records.
   - To locate a record, we find the index record with the largest search key value less than or equal to the search key value we are looking for.
   - We start at that record pointed to by the index record, and proceed along the pointers in the file (that is, sequentially) until we find the desired record.
2. Figures 1 and 2 show dense and sparse indices for the deposit file.

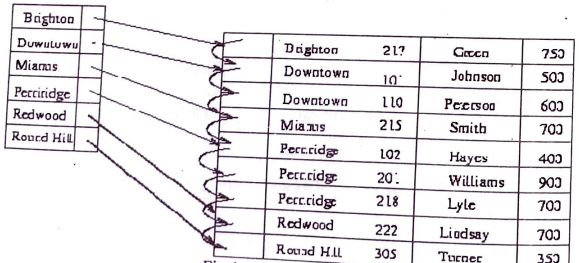


Fig 1: Dense index

3. Notice how we would find records for Perryridge branch using both methods. (Do it!)

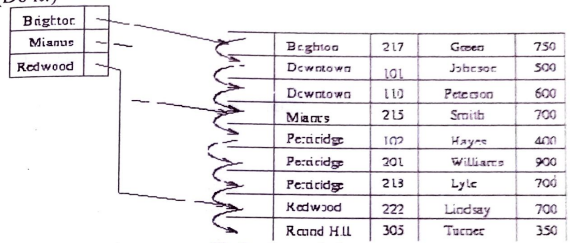


Fig 2: Sparse index

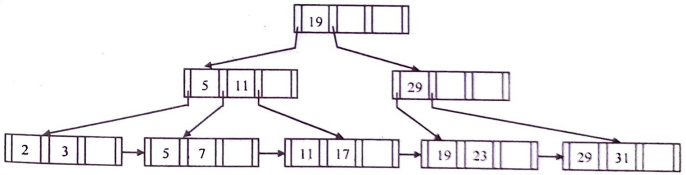
4. Dense indices are faster in general, but sparse indices require less space and impose less maintenance for insertions and deletions.
5. A good compromise: to have a sparse index with one entry per block.

3. Construct a B+ tree for the following set of values: **[WBUT 2014]**
(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)
Assume that the tree is initially empty and values are added in ascending order. Contract B+ tree the cases where the number of pointers that will fit in one node is as follows
a. Four
b. Six
c. Eight

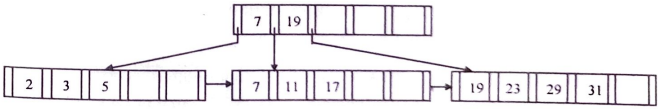
Answer:

The following were generated by inserting values into the B+ tree in ascending order. A node (other than the root) was never allowed to have fewer than [n/2] values/pointers.

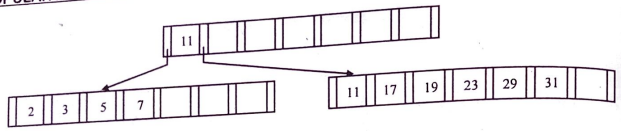
a)



b)



c)



4. What is index? Define clustering indices, hash indices, dense indices and Primary-secondary index. **[WBUT 2015]**

Answer:

1st Part:

We know that information in the DBMS files is stored in form of records. Every record is equipped with some key field, which helps it to be recognized uniquely.

Indexing is a data structure technique to efficiently retrieve records from database files based on some attributes on which the indexing has been done. Indexing in database systems is similar to the one we see in books.

2nd Part:

Indexing is defined based on its indexing attributes. Indexing can be one of the following types:

- Primary Index: If index is built on ordering 'key-field' of file it is called Primary Index. Generally it is the primary key of the relation.
- Secondary Index: If index is built on non-ordering field of file it is called Secondary Index.
- Clustering Index: If index is built on ordering non-key field of file it is called Clustering Index.

Ordering field is the field on which the records of file are ordered. It can be different from primary or candidate key of a file.

Ordered Indexing is of two types:

- Dense Index
- Sparse Index

Dense Index

In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself. Index record contains search key value and a pointer to the actual record on the disk.

Sparse Index

In sparse index, index records are not created for every search key. An index record here contains search key and actual pointer to the data on the disk. To search a record we first proceed by index record and reach at the actual location of the data. If the data we are looking for is not where we directly reach by following index, the system starts sequential search until the desired data is found.

Multilevel Index

Index records are comprised of search-key value and data pointers. This index itself is stored on the disk along with the actual database files. As the size of database grows so does the size of indices. There is an immense need to keep the index records in the main

memory so that the search can speed up. If single level index is used then a large size index cannot be kept in memory as whole and this leads to multiple disk accesses.

Multi-level Index helps breaking down the index into several smaller indices in order to make the outer most level so small that it can be saved in single disk block which can easily be accommodated anywhere in the main memory.

5. a) Discuss about possible situations where hashing is preferred over indexing to search values. **[WBUT 2018]**

Answer:

Hashing is a technique that is used to uniquely identify a specific object from a group of similar objects. Some examples of how hashing is used in our lives include:

- In universities, each student is assigned a unique roll number that can be used to retrieve information about them.
- In libraries, each book is assigned a unique number that can be used to determine information about the book, such as its exact position in the library or the users it has been issued to etc.

In both these examples the students and books were hashed to a unique number.

Assume that one wants to assign a key to an object to make searching easy. To store the key/value pair, one can use a simple array like data structure where keys (integers) can be used directly as an index to store values. However, in cases where the keys are large and cannot be used directly as an index, one should use hashing.

In hashing, large keys are converted into small keys by using hash functions. The values are then stored in a data structure called hash table. The idea of hashing is to distribute entries (key/value pairs) uniformly across an array. Each element is assigned a key (converted key). By using that key one can access the element in O(1) time. Using the key, the algorithm (hash function) computes an index that suggests where an entry can be found or inserted.

b) Discuss about possible situations where is indexing preferred over hashing to search values. **[WBUT 2018]**

Answer:

Indexes can be created on columns to speed up queries. Indexes provide faster access to data for operations that return a small portion of a table's rows.

In general, one should create an index on a column in any of the following situations:

- The column is queried frequently.
- A referential integrity constraint exists on the column.
- A UNIQUE key integrity constraint exists on the column.

One can create an index on any column; however, if the column is not used in any of these situations, creating an index on the column does not increase performance and the index takes up resources unnecessarily.

c) Why is secondary index defined? Discuss some of the possible applications. **[WBUT 2018]**

Answer:

1st Part: Secondary indexing provides a way to meet the different processing requirements of various applications. Secondary indexing allows to have an index based on any field in the database, not just the key field in the root segment.

When an application program accesses a database through a secondary index, the database records are processed in an alternative sequence.

2nd Part: Secondary indexes are typically defined to improve performance regarding residual conditions in the where clause. There are multiple ways to improve performance of these conditions, including but not limited to row level partitioning, secondary indexes etc.

Secondary indexes are mostly used to improve access on the queries that use a non-primary index column in search conditions.

Following are some common uses of Secondary Index:

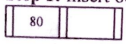
1. If a non-primary index column is being used in where clause often, define Secondary Index on it.
2. One can use Unique Secondary Index to enforce uniqueness in a PPI table where partition columns are not part of Primary Index.
3. One can define Non-Unique Secondary Index in a PPI table to make a Primary Index access single amp.

d) Show the steps of Insertion in a B+ Tree of order 3 for following values: 80, 50, 20, 70, 30, 100, 120, 60 **[WBUT 2018]**

Answer:

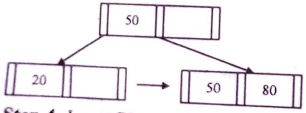
Since the B+ tree is of order 3 that means the B+ tree can hold up to 3 pointers and 2 keys. Assuming that the insertion will be of ascending order, the steps are as follows:

Step 1: Insert 80

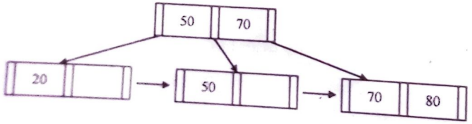


Step 2: Insert 50

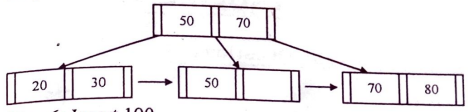
Step 3: Insert 20



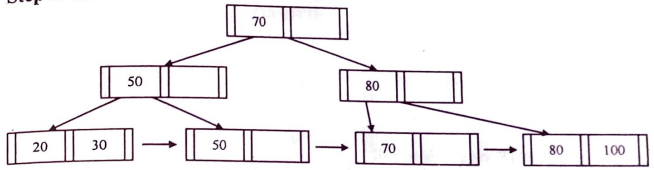
Step 4: Insert 70



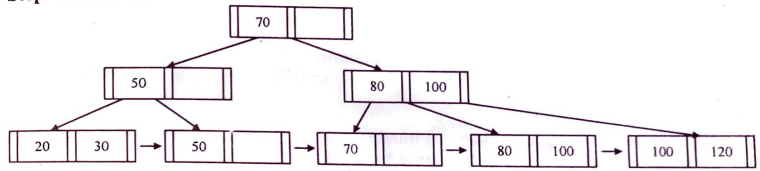
Step 5: Insert 30



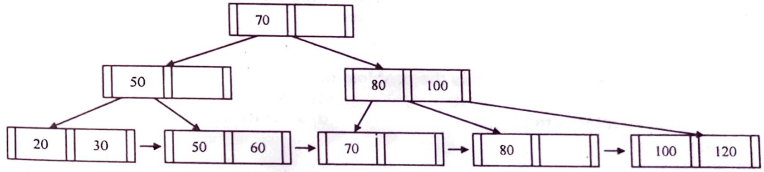
Step 6: Insert 100



Step 7: Insert 120



Step 8: Insert 60



6. Write short notes on the following:

a) Indexing **[WBUT 2014]**

b) Multi-level index **[WBUT 2016, 2018]**

c) Primary Indexing **[WBUT 2017]**

Answer:

a) Indexing: ***Refer to Question No. 4 of Long Answer Type Questions.***

b) Multi-level index: ***Refer to Question No. 2 of Short Answer Type Questions.***

c) Primary Indexing: ***Refer to Question No. 1(1st Part) of Long Answer Type Questions.***

TRANSACTION MANAGEMENT

Multiple Choice Type Questions

1. What are the transaction properties? [WBUT 2007]
   a) ABCD property b) ACID property
   c) DEADLOCK d) READ - WR property
   Answer: (b)

2. COMMIT is a ..... Statement. [WBUT 2013]
   a) TCL b) DCL c) DML d) DQL
   Answer: (a)

3. In transaction, a WRITE operation will [WBUT 2016]
   a) read data from table and write on buffer b) write on table
   c) depend on application d) lock a data for updating
   Answer: (a)

4. Grant and revoke are ..... statements. [WBUT 2017]
   a) DDL b) TCL c) DCL d) DML
   Answer: (b)

5. In 2-phase locking a transaction must: [MODEL QUESTION]
   a) release all its locks at the same time
   b) NOT obtain any new locks once it has started releasing locks
   c) Only obtain locks on items not used by any other transactions
   d) Ensure that deadlocks will never occur
   Answer: (d)

6. The concurrency control has the problem of [MODEL QUESTION]
   a) lost updates b) dirty read
   c) unrepeatable read d) all of these
   Answer: (d)

7. Advantage of locking algorithms in concurrent execution of DB transaction is ..... [MODEL QUESTION]
   a) deadlock b) concurrency c) consistency d) none of these
   Answer: (b)

8. Which phase is not part of a two phase locking protocol? [MODEL QUESTION]
   a) Growing phase b) Shrinking phase
   c) Stabilization phase d) None of these
   Answer: (c)

9. Which is not an ACID property? [MODEL QUESTION]
   a) Atomicity b) Integrity c) Consistency d) Durability
   Answer: (b)

10. There is a conflict in a schedule if [MODEL QUESTION]
    a) two transactions work on the same data item
    b) the operations are from different transactions
    c) at least one of the operations is write
    d) all of these
    Answer: (d)

11. Time stamp is used for [MODEL QUESTION]
    a) serialization b) deadlock control
    c) transaction log d) both (b) & (c)
    Answer: (d)

12. What type of lock forbids any other user to access of the data in any way? [MODEL QUESTION]
    a) Shared b) Exclusive c) Limited d) Concurrent
    Answer: (b)

13. Which to the following makes the transaction permanent in the database? [MODEL QUESTION]
    a) view b) commit c) rollback d) flashback
    Answer: (b)

14. Lock point in a two phase locking protocol denotes [MODEL QUESTION]
    a) start of growing phase b) end of growing phase
    c) start of shrinking phase d) anywhere in the phase
    Answer: (a)

15. Precedance graph helps to find a [MODEL QUESTION]
    a) serializable schedule b) recoverable schedule
    c) deadlock free schedule d) cascadeless schedule
    Answer: (a)

16. If a transaction T has obtained an exclusive lock on item Q, then T can [MODEL QUESTION]
    a) read Q b) write Q
    c) write Q but not read Q d) both read and write Q
    Answer: (d)

Short Answer Type Questions

1. Explain ACID properties of transactions. [WBUT 2009, 2012, 2014, 2016, 2017]
   OR,
   Note on ACID Property. [WBUT 2008, 2010]

Answer:

Properties of Transactions

With the help of the abbreviated word ACID the properties of a transaction can be explained as follows:

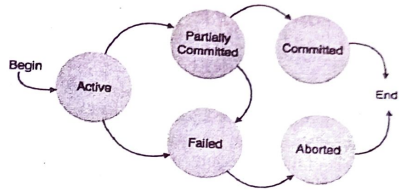
1. *Atomicity*
A transaction is either performed in its entirety or not performed at all. The system guarantees this using logging, shadowing, distributed commit.
2. ***Consistency (Preservation)***
The complete execution of a transaction takes the database from one consistent state to another. The application guarantees this by correctly marking transaction boundaries.
3. *Isolation*
The execution of a transaction is not interfered with by any other (concurrent) transactions. It should not make its update values visible to other transactions until it is completed successfully. This is guaranteed through locking or timestamp-based concurrency control.
4. ***Durability (Permanency)***
Changes applied to the database by a committed transaction must persist in the database even if the system crashes before all changes are reflected by writing updates to stable storage.

2. Explain all the states of a transaction with example for each state. [WBUT 2017]
Answer:
Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

A's Account
Open_Account(A)
Old_Balance = A.balance
New_Balance = Old_Balance - 500
A.balance = New_Balance
Close_Account(A)

B's Account
Open_Account(B)
Old_Balance = B.balance
New_Balance = Old_Balance + 500
B.balance = New_Balance
Close_Account(B)

A transaction in a database can be in one of the following states –
Active – In this state, the transaction is being executed. This is the initial state of every transaction.
Partially Committed – When a transaction executes its final operation, it is said to be in a partially committed state.
Failed – A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

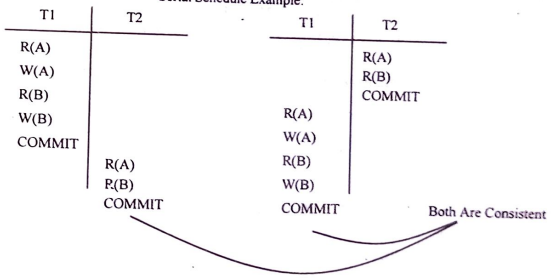


Aborted – If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts –

Re-start the transaction
Kill the transaction
Committed – If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

3. What is a schedule? Give an example of a serial schedule with two transactions.
[WBUT 2017]
Answer:
A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.

Serial Schedule Example:



4. What is two phase locking protocol? How does it guarantee serializability? [MODEL QUESTION]

Answer:

Two-phase locking

According to the **two-phase locking** protocol, a transaction handles its locks in two distinct, consecutive phases during the transaction's execution:

b) **Expanding phase** (aka Growing phase): locks are acquired and no locks are released (the number of locks can only increase).

c) **Shrinking phase:** locks are released and no locks are acquired.

The serializability property is guaranteed for a schedule with transactions that obey the protocol. The 2PL *schedule class* is defined as the class of all the schedules comprising transactions with data access orders that could be generated by the 2PL protocol (or in other words, all the schedules that the 2PL protocol can generate).

Typically, without explicit knowledge in a transaction on end of phase-1, it is safely determined only when a transaction has entered its *ready* state in all its processes (processing has ended, and it is ready to be committed; no additional data access and locking are needed and can happen). In this case phase-2 can end immediately (no additional processing is needed), and actually no phase-2 is needed. Also, if several processes (two or more) are involved, then a synchronization point (similar to atomic commitment) among them is needed to determine end of phase-1 for all of them (i.e., in the entire distributed transaction), to start releasing locks in phase-2 (otherwise it is very likely that both 2PL and Serializability are quickly violated). Such synchronization point is usually too costly (involving a distributed protocol similar to atomic commitment), and end of phase-1 is usually postponed to be merged with transaction end (atomic commitment protocol for a multi-process transaction), and again phase-2 is not needed. This turns 2PL to SS2PL (see below). All known implementations of 2PL in products are SS2PL based, and whenever 2PL (or Strict 2PL, S2PL) practical utilization has been mentioned in the professional literature, the intention has been SS2PL.

5. With suitable examples, show how recovery in a database system can be done using LOG files with – [MODEL QUESTION]

a) immediate updation

b) deferred updation.

Answer:

a) Immediate Update

Immediate update, or UNDO/REDO, is another algorithm to support ABORT and machine failure scenarios.

1) While a transaction runs, changes made by that transaction can be written to the database at any time. However, the original and the new data being written must both be stored in the log BEFORE storing it on the database disk.
2) On a commit:
3) All the updates which has not yet been recorded on the disk is first stored in the log file and then flushed to disk.
4) The new data is then recorded in the database itself.

5) On an abort, REDO all the changes which that transaction has made to the database disk using the log entries.
6) On a system restart after a failure, REDO committed changes from log.

Example

Using immediate update, and the transaction TRAN1 again, the process is:

Time	Action	LOG
t1	START	-
t2	read(A)	-
t3	write(10,B)	Was B == 6, now 10
t4	write(20,C)	Was C == 2, now 20
t5	COMMIT	COMMIT

b) *Deferred Update*

Deferred update, or NO-UNDO/REDO, is an algorithm to support ABORT and machine failure scenarios.

1) While a transaction runs, no changes made by that transaction are recorded in the database.
2) On a commit:
3) The new data is recorded in a log file and flushed to disk.
4) The new data is then recorded in the database itself.
5) On an abort, do nothing (the database has not been changed).
6) On a system restart after a failure, REDO the log.

Example

Consider the following transaction TRAN1

Transaction TRAN1
read(A)
write(10,B)
write(20,C)
Commit

Using deferred update, the process is:

Time	Action	Log
t1	START	-
t2	read(A)	-
t3	write(10,B)	B = 10
t4	write(20,C)	C = 20
t5	COMMIT	COMMIT

DISK

After		
		B = 10
A = 5	C = 20	

Before		
		B = 6
A = 5	C = 2	

If the DMBS fails and is restarted:

- The disks are physically or logically damaged then recovery from the log is impossible and instead a restore from a dump is needed.
- If the disks are OK then the database consistency must be maintained. Writes to the disk which was in progress at the time of the failure may have only been partially done.
- Parse the log file, and where a transaction has been ended with 'COMMIT' apply the data part of the log to the database.
- If a log entry for a transaction ends with anything other than COMMIT, do nothing for that transaction.
- flush the data to the disk, and then truncate the log to zero.
- the process or reapplying transaction from the log is sometimes referred to as 'roll forward'.

Long Answer Type Questions

1. What are the roles of the Analysis, Redo and Undo phases in the recovery algorithm 'ARIES'? **[MODEL QUESTION]**

Answer:

The ARIES recovery procedure consists of three main steps:

- **Analysis:** It identifies the dirty (updated) pages in the buffer and the set of transactions active at the time of crash. The appropriate point in the log where REDO operation should start is also determined.
- **REDO phase:** It actually reapplies updates from the log to the database. Generally the REDO operation is applied to only committed transactions. However, in ARIES, this is not the case. Certain information in the ARIES log will provide the start point for REDO, from which REDO operations are applied until the end of the log is reached. Thus only the necessary REDO operations are applied during recovery.
- **UNDO phase:** The log is scanned backwards and the operations of transactions that were active at the time of the crash are undone in reverse order. The information needed for ARIES to accomplish its recovery procedure includes the log, the transaction table, and the dirty page table. In addition, check pointing is used.

2. i) How can you differentiate between the process recovery and restoration? Give example.
ii) What is integral backup? Explain redo and undo log.
iii) Why do we use index in a database table, although we have candidate key to identify a row uniquely? What is sparse index?
iv) Explain the process of checkpoint based recovery with suitable example.
[MODEL QUESTION]

Answer:

i) Restoring a database is copying the physical files from a backup medium (disk or tape) to the appropriate file locations for database operation. Recovery is process of updating database files restored from backup w/changes made to the database since backup, typically using redo log files.

Recovering files typically refers to saving one or more files. Recovery is the process of applying redo logs to the database to roll it forward. One can roll-forward until a specific point-in-time (before the disaster occurred), or roll-forward until the last transaction recorded in the log files.

ii) There is no term as integral backup in database dictionary, however, incremental backup means

Force writing a log: *Before T reaches its commit point, all log records must be written to disk.*

When recovering after a (system) failure:

- ***Rollback of transactions*:** Needed for transactions that have a [begin_transaction, T] in the Log, but no [commit, T].

Redoing transactions: Needed for transactions that have a [commit, T] in the Log after the last database checkpoint (writing all dirty database to disk).

Two types of algorithm:

– Deferred update

- DB is updated only when transaction commits. Here the DB is not modified with the proceeding transaction. Rather if [begin_transaction, T] in the Log together with [commit, T] then the DB is restored from the log records.

This is a type of NO-UNDO/REDO algorithm. Here Undo is not needed. Any changes made to the DB result from completed, committed transactions.

Redo operations perhaps necessary. Some transactions may have committed, but the changes not yet been saved to the DB.

– Immediate update

- DB is updated before transaction commits. Here the DB is not modified with the proceeding transaction. Rather if [begin_transaction, T] in the Log together with [commit, T] then the DB is restored from the log records.

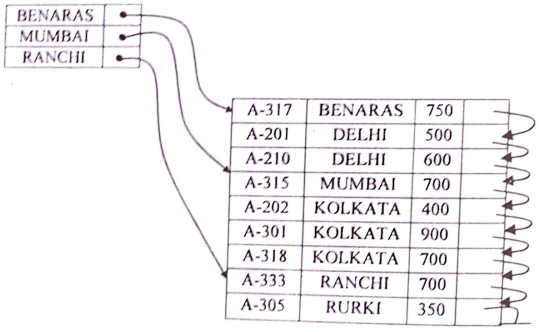
Here we assume that schedules are strict. Update operations recorded in disk log at intervals by force-writing. Transaction may commit before all changes saved to DB. This is a type of UNDO/REDO algorithm.

iii) In order to access the data directly index is used.
The process of Indexing is organized by using the candidate key of a relation through a separate data table with the help of pointers. If index is not there then data searching will be sequential. However, if there is an index over the candidate keys then it will be direct and fast.

Sparse Index files:
Dense indices are faster in general, but sparse indices requires less space and impose less maintenance for insertion and deletion:
A good compromise is to have a Sparse index with one entry per block. Because:

- Costliest is in bringing a block into main memory.
- It is guaranteed to have the correct block with this method, unless record is on a overflow block.
- Index size still small.

Sparse index Contains index records for only some search-key values. It is applicable when records are sequentially ordered on search key. To locate a record with search-key value *K* we find index record with largest search-key value less than or equal to *K*. Then search file sequentially starting at the record to which the index record points. It involves less space and less maintenance overhead for insertions and deletions. But it is generally slower than dense index for locating records.



3
Example of Sparse Index Files

Sparse Secondary index:
We can use an extra-level of indirection to implement secondary indices on search key that are not candidate key. A pointer does not point directly to the file but to a bucket that contains pointers to the file. Secondary indices must be dense, with an index entry for every search key value, and a pointer to every record in the file. It improves the

performance of queries on non-primary keys. They also impose serious overhead on database modification: whenever a file is updated, every index must be updated.

iv) The recovery process refers to the log to find a point at which it can restore the state and to decide the undo and redo operation. When the schedule operates in an interleaved fashion, the point of restoration may involve retracing a very large number of operations involving many transactions. There may not be any limit the system must look back in the log. It is better when the system keeps writing all updates intermittently till the point the effects of all these transactions are permanently recorded. The method by which this is done is called checkpointing. While the transaction manager maintains a log, it also periodically performs checkpoints. This involves the following tasks.

- Temporarily halting the starting of new transactions until all active transactions are either committed or aborted.
- Making a backup copy of the database.
- Making a copy of log file.
- Appending a checkpoint type log record to the log.

3. a) Distinguish between locking and timestamp protocols for concurrency controls. Explain multi-version two-phase locking.
b) Define three concurrency problems, dirty read, non-repeatable read, phantoms.
c) Consider the following *two* transactions:

T_1 : read (*A*) ;
read (*B*) ;
if *A* = 0, then *B* : *B* + 1 ;
write (*B*)

T_2 : read (*B*) ;
read (*A*) ;
If *B* = 0, then *A* : *A* + 1 ;
Write (*A*)

Add lock and unlock instructions to transactions T_1 and T_2, so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock? [MODEL QUESTION]

Answer:

a) Locking data items to prevent multiple transactions from accessing the items concurrently is the most common method used to ensure serializability. In this section, we examine the basic idea behind the locking concepts and introduce two-phase locking protocol.
A lock is a variable associated with data items that describe the status of the item with respect to possible operations that can be applied to it.
There are various types of locks are used in concurrency control like **Binary Locks and Shared/Exclusive Locks**. In the locking protocols, the order between every pair of conflicting transactions is determined at execution time by the first lock that they both request that involves incompatible modes.

Another method for determining the serializability order is select an ordering among transactions in advance. Timestamp – ordering scheme is the typical example of this method.

A timestamps is a unique variable associated with a transaction. Timestamp of transaction T is denoted by TS(T). This timestamp is assigned by the database system in the order in which the transaction submitted to the system. If Ti has been assigned timestamp TS(Ti) and a new transaction Tj enters the system then TS(Ti) < TS(Tj)

There are two simple ways for generating timestamp:

- Use the current value of system clock as the timestamps
- Use a logical counter that is incremented each time a value is assigned to a transaction as timestamp.

In order to use timestamp techniques, each data item Q is associtated with two timestamp values:

- read_TS(Q). This is the largest timestamp of any transaction that have successfully read data item Q.

read_TS(Q) = TS(T) where T is the youngest transaction that has read Q successfully

- write_TS(Q). This is the largest timestamp of any transaction that sucessfully written data item Q.

write_TS(Q) = TS(T) where T is the youngest transaction that has written Q successfully

Multi-version two-phase locking

The aim of Multi-Version Concurrency is to avoid the problem of Writers blocking Readers and vice-versa, by making use of multiple versions of data.

The problem of Writers blocking Readers can be avoided if Readers can obtain access to a previous version of the data that is locked by Writers for modification.

The problem of Readers blocking Writers can be avoided by ensuring that Readers do not obtain locks on data. Multi-Version Concurrency allows Readers to operate without acquiring any locks, by taking advantage of the fact that if a Writer has updated a particular record, its prior version can be used by the Reader without waiting for the Writer to Commit or Abort. In a Multi-version Concurrency solution, Readers do not block Writers, and vice versa.

***Requirements of Multi-Version Concurrency* systems**

As its name implies, multi-version concurrency relies upon multiple versions of data to achieve higher levels of concurrency. Typically, a DBMS ordering multi-version concurrency (MVDB), needs to provide the following features:

1. The DBMS must be able to retrieve older versions of a row.
2. The DBMS must have a mechanism to determine which version of a row is valid in the context of a transaction. Usually, the DBMS will only consider a version that was committed prior to the start of the transaction that is running the query. In order to determine this, the DBMS must know which transaction created a particular version of a row, and whether this transaction committed prior to the starting of the current transaction.

b) Dirty read—Dirty reads occur when one transaction reads data that has been written but not yet committed by another transaction. If the changes are later rolled back, the data obtained by the first transaction will be invalid.

Nonrepeatable read—Nonrepeatable reads happen when a transaction performs the same query two or more times and each time the data is different. This is usually due to another concurrent transaction updating the data between the queries.

Phantom reads—Phantom reads are similar to nonrepeatable reads. These occur when a transaction (T1) reads several rows, and then a concurrent transaction (T2) inserts rows. Upon subsequent queries, the first transaction (T1) finds additional rows that were not there before.

T31	T32
lock-S(A)	**lock-S**(B)
read(A)	**read**(B)
lock-X(B)	**lock-X**(A)

The transactions are now deadlocked.

4. a) Consider the following transactions and find how many different schedules of the two transactions are possible? How many of these are serializable?

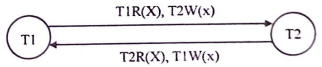
Transaction 1	Traction 2
Read (X)	Read (X)
X:=X–N	X:=X+M
Write (X)	Write (X)
Read (Y)	------
Y:=Y+N	------
Write (Y)	------

[MODEL QUESTION]

Answer:

A **serial schedule** is when all the operations of one transactions appear together (not mixed with the operations of any other transactions on the schedule).

A **serializable schedule** is a weaker term -- it is a schedule where the operations of different transactions may be mixed together on the schedule, so long as they are conflict-equivalent to some serial schedule.



There is a loop between T1 and T2 with the given schedule. Conflict will occur. Schedule can execute in exclusive mode to become serializable.

However,

Read (Y)
Y:=Y+N
Write (Y)

Has no conflict. It can be a serial schedule.

b) Transactions cannot be nested inside one another. Why not? **[MODEL QUESTION]**

Answer:

Nested Transactions

Problem: Lack of mechanisms that allow:

- A top-down, functional decomposition of a transaction into sub transactions.
- Individual sub transactions to abort without aborting the entire transaction.
- Although a nested transaction looks similar to a distributed transaction, it is not conceived of as a tool for accessing a multi database.

Characteristics of Nested Transactions Parent:

- creates children to perform subtasks
- either sequentially or concurrently waits until all children complete

Characteristics of Nested Transactions Consistency:

- Individual subtransactions are not necessarily consistent,
- But nested transaction as a whole is consistent
- No comm between parent and children.

Each subtransaction (together with its descendants):

- is isolated w.r.t. each sibling (and its descendants).
- hence, siblings are serializable,
- but order is not determined i.e. NT is *non-deterministic*.

Concurrent nested transactions are serializable.

A sub transaction is atomic:

- It can abort or commit independently of other sub transactions.
- Commit is conditional on commit of parent (since child task is a subtask of parent task).
- Abort causes abort of all sub transaction's children.

Nested transaction commits when root commits:

- At that point updates of committed sub transactions are made durable.

c) Systems do not allow transaction to commit changes to database on an individual basis. i.e., without simultaneously committing changes to all other database. Why not? **[MODEL QUESTION]**

Answer:

It doesn't allow a given transaction to commit changes to files on an individual basis because it must make sure that the system remains in a consistent state before and after a transaction. So it has to make sure that the system as a whole is reliable before making changes on an individual basis.

5. a) Explain the purpose of the checkpoint mechanism. How often should checkpoints be performed?

b) How does 'strict two-phase' and 'consecutive two-phase' locking protocol differ. **[MODEL QUESTION]**

Answer:

a) Checkpointing is done with log-based recovery schemes to reduce the amount of searching that needs to be done after a crash. If there is no checkpointing, then the entire log must be searched after a crash, and all transactions "redone" from the log. If checkpointing is used, then most of the log can be discarded. Since checkpoints are very expensive, how often they should be taken depends upon how reliable the system is. The more reliable the system, the less often a checkpoint should be taken.

b) According to the two-phase locking protocol a transaction handles its locks in two distinct, consecutive phases during the transaction's execution:

- Expanding phase (aka Growing phase): locks are acquired and no locks are released (the number of locks can only increase).
- Shrinking phase: locks are released and no locks are acquired.

The two phase locking (2PL) rule can be summarized as: never acquire a lock after a lock has been released. The serializability property is guaranteed for a schedule with transactions that obey this rule.

Typically, without explicit knowledge in a transaction on end of phase-1, it is safely determined only when a transaction has completed processing and requested commit. In this case all the locks can be released at once (phase-2).

To comply with the Strict two phase locking protocol a transaction needs to comply with 2PL, and release its write (exclusive) locks only after it has ended, i.e., being either committed or aborted. On the other hand, read (shared) locks are released regularly during phase 2. This protocol is not appropriate in B-trees because it causes Bottleneck (while B-trees always starts searching from the parent root).

6. Write short note on Recovery. **[MODEL QUESTION]**

Answer:

A transaction T reaches its commit point when all its operations that access the database have been executed successfully and recorded to the log.

Beyond the commit point, the transaction is said to be committed and its effects assumed to be permanently recorded in the database. The transaction then writes an entry [commit, T] into the log.

Force writing a log: *Before T reaches its commit point, all log records must be written to disk.*

When recovering after a (system) failure:

- ***Rollback of transactions*:** Needed for transactions that has a [begin_transaction, T] in the Log, but no [commit, T].

Redoing transactions: Needed for transactions that have a [commit, T] in the Log after the last database checkpoint (writing all dirty database to disk).

Two types of algorithm:

– Deferred update
- DB is updated only when transaction commits. Here the DB is not modified with the proceeding transaction. Rather if [begin_transaction, T] in the Log together with [commit, T] then the DB is restored from the log records.

This is a type of NO-UNDO/REDO algorithm. Here Undo is not needed. Any changes made to the DB result from completed, committed transactions.
Redo operations perhaps necessary. Some transactions may have committed, but the changes not yet been saved to the DB.

– Immediate update
- DB is updated before transaction commits. Here the DB is not modified with the proceeding transaction. Rather if [begin_transaction, T] in the Log together with [commit, T] then the DB is restored from the log records.

Here we assume that schedules are strict. Update operations recorded in disk log at intervals by force-writing. Transaction may commit before all changes saved to DB. This is a type of UNDO/REDO algorithm.

QUESTION 2014

GROUP – A
(Multiple Choice Type Questions)

1. Choose the correct alternatives for any *ten* of the following:

i) The set of permitted values for each attribute is called its
a) attribute set b) attribute range ✓c) domain d) group

ii) The operation on certain relation X, produces Y such that Y contains only selected attribute of X, such operation is
✓a) projection b) selection c) union d) difference

iii) What is the cardinality of a table with 100 rows and 10 columns?
a) 1000 ✓b) 100 c) 10 d) 10000

iv) An attribute of one table matching with the primary key of another table is called
✓a) foreign key b) secondary key c) candidate key d) surrogate key

v) If two relations have 5 and 10 rows respectively, then no. of tuples in Cartesian product will be
✓a) 50 b) 5 c) 10 d) 15

vi) The primary key indexing techniques do not allow
a) Sets of relations b) Multiple attributes
✓c) duplicate data d) Many to many relation

vii) A candidate key which is not a primary key is known as
a) super key ✓b) alternate key c) foreign key d) non prime attribute

viii) Which one is not a traditional set operator defined on relational algebra?
a) Union b) Intersection c) Set Difference ✓d) Join

ix) Which operator performs pattern matching in SQL?
a) Except b) Intersect c) Join ✓d) Like

x) Association among several entities is known as
a) attributes ✓b) relationship c) field d) none of these

xi) 2NF is based on _____ dependency
a) transitive ✓b) total c) partial d) functional

xii) If $X \supseteq Y$ then $X \rightarrow Y$ is an example of _____ dependency
a) partial b) join c) non trivial ✓d) trivial

GROUP – B
(Short Answer Type Questions)

2. Explain Relational Algebra using the operators $\{\partial, \Pi, \cup, -, X$ and show that: $A \cap B = A \cup B - ((A - B) \cup (B - A))$

See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Short Answer Type Question No. 5.

3. Describe the three-level architecture of DBMS.
See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Short Answer Type Question No. 1(a).

4. a) Explain the difference between external, internal and conceptual schemas.
b) What is the highest NF of each of the following relations?
i. $R1\ (J, K, L)$ with FDs are $J \rightarrow K,\ J \rightarrow L,\ K \rightarrow L$
ii. $R2(J, K, L, M)$ with FDs are $J \rightarrow KL,\ LM \rightarrow K$
a) See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Short Answer Type Question No. 3.
b) See Topic: NORMALIZATION, Long Answer Type Question No. 3.

5. Explain ACID properties of transactions.
See Topic: TRANSACTION MANAGEMENT, Short Answer Type Question No. 1.

6. "All primary keys are the super key but the converse is not true." Clarify.
See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Short Answer Type Question No. 3(a).

GROUP – C
(Long Answer Type Questions)

7. i) Describe dense and sparse indices with diagram.
ii) Define concept of aggregation. Give two examples where this concept is useful.
i) See Topic: INDEXING, Long Answer Type Question No. 2.
ii) See Topic: ER MODEL, Long Answer Type Question No. 4.

8. i) Describe the three tier architecture of the general DBMS.
ii) Let $R = (A, B)$ and $S = (A, C)$ and let r(R) and r(S) be relations. Write relational algebra expression equivalent to the domain relational calculus expressions:
a) $\{< a > \mid$ there exist $b (< a, b >$ belongs to $r \wedge b = 17\}$
b) {<a, b, c> | <a, b> belongs to $r \wedge < a, c >$ belongs to $s\}$
i) See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Short Answer Type Question No. 1(a).
ii) See Topic: ER MODEL, Long Answer Type Question No. 5.

9. i) Why certain functional dependencies are called trivial functional dependencies?
ii) Use Armstrong's axioms to prove the soundness of the union rule.
iii) Compute the closure of the following set F of FDs for each relation schema
R = (A, B, C, D, E).
A-> BC
CD->E
B->D
E->A.
List the candidate key for R.
See Topic: NORMALIZATION, Long Answer Type Question No. 9.

10. i) Construct a B+ tree for the following set of 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. Contract B+ tree the cases where the number of pointers that will fit in one node is as follows
a. Four
b. Six
c. Eight
ii) Consider the following tables
employee (emp_name, street, city)
Works (emp_name, company _ name, salary)
company (company_name, city)
managers (emp_name, manager _name)
Give SQL, expression for the following queries
a. Find the names and cities of residence of all employees who work for first Bank Corporation.
b. Find the name, street address and cities of residencies of all employees who work for first Bank Corporation and earn more than Rs. 100000.
c. Find all employees in the database who earn more than each employee of Small Bank Corporation.
i) See Topic: INDEXING, Long Answer Type Question No. 3.
ii) See Topic: SQL, Long Answer Type Question No. 2.

11. Write short notes on any three topics:
a) Functional Dependency
b) Indexing
c) Mapping cardinalities
d) Query processing and optimization
e) Hashing
a) See Topic: NORMALIZATION, Long Answer Type Question No. 12.a)
b) See Topic: INDEXING, Long Answer Type Question No. 6(a).
c) See Topic: ER MODEL, Long Answer Type Question No. 13(a).
d) See Topic: FILE ORGANIZATION & QUERY OPTIMIZATION, Long Answer Type Question No. 2(c).
e) OUT OF SYLLABUS

QUESTION 2015

GROUP – A
(Multiple Choice Type Questions)

1. Choose the correct alternatives for any *ten* of the following:
i) Referential integrity is directly related to
a) relational key ✓b) foreign key c) primary key d) candidate key

ii) Which level of Abstraction describes how data are stored in the database?
✓a) Physical level b) View level c) Abstraction level d) Logical level

iii) Which of the following is true?
✓a) A relation in BCNF is always in 3NF b) A relation in 3NF is always in BCNF
c) BCNF and 3NF are same d) A relation in BCNF is not in 3NF

iv) Consider a schema $R(A, B, C, D)$ and functional dependencies $A \rightarrow B$ and $C \rightarrow D$. Then the decomposing $R_1(A, B)$ and $R_2(C, D)$ is
- ✓a) dependency preserving but not lossless join
- b) dependency preserving and lossless join
- c) lossless join but not dependency preserving
- d) lossless join

v) To select a tuple from a relational database table, the symbol used in relational algebra is
a) ρ (Row) ✓b) σ (Sigma) c) Π (Project) d) none of these

vi) $R = (A, B, C, D)$, $F = (AB \rightarrow C, C \rightarrow D)$. Find candidate key
✓a) AB b) ABC c) $ABCD$ d) none of these

vii) Which is the SQL command to remove rows from a table?
a) REMOVE ✓b) DELETE c) TRUNCATE d) All of these

viii) The first phase of query processing is:
✓a) decomposition b) restructuring c) analysis d) none of these

ix) The distinguishable parts of a record are called:
a) Files b) Data ✓c) Fields d) Database

x) Normalization of database is needed to:
a) make data more intelligible to humans b) remove error in data entry
✓c) eliminate redundant data d) all of these

GROUP – B
(Short Answer Type Questions)

2. Compute the closure of the following set F of functional dependencies for relation schema:
$R = (A, B, C, D, E)$, $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$
List the candidate keys for R
See Topic: NORMALIZATION, Long Answer Type Question No. 5.

3. Explain the query optimization technique with relevant examples
See Topic: FILE ORGANIZATION & QUERY OPTIMIZATION, Long Answer Type Question No. 1.

4. What is lossless decomposition? Consider the relation $R_1(A, B, C)$ and $R_2(C, D)$. Show that this decomposition is dependency preserving or not.
1st part: See Topic: NORMALIZATION, Short Answer Type Question No. 9.
2nd part: Question is not clear.

5. How does BCNF differ from 3NF? Why is it considered stronger than 3NF?
See Topic: NORMALIZATION, Short Answer Type Question No. 14.

6. Discuss the different database anomalies.
See Topic: NORMALIZATION, Short Answer Type Question No. 15.

GROUP – C
(Long Answer Type Questions)

7. a) Consider the following schema:
Book(acc_no, yr_pub, title)
User(card_no, bname, baddress)
Borrow(acc_no, doi, card_no)
Where acc_no is account number, yr_pub is year of publication, bname is borrower name, baddress is borrower address, doi is date of issue.
Perform the following queries on the table.(In SQL)
(i) Find the account number whose year of publication is 1985.
(ii) Display the title of the book which has been borrowed by "Vijoy"
(iii) Find the borrower name who lives in same city as "Vijoy"
(iv) find the borrower name and address who should issue book on 14-05-1988
(v) Find the acc_no of Book whose year of publication is 1992 and title is "Compiler Design"
b) State Armstrong's Axioms.
a) See Topic: SQL, Long Answer Type Question No. 3.
b) See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Long Answer Type Question No. 3.

8. a) What is index? Define clustering indices, hash indices, dense indices and Primary-secondary index.
b) What is data abstraction?
c) Define the concept of aggregation, generalization, and specialization and attribute inheritance.
a) See Topic: INDEXING, Long Answer Type Question No. 4.
b) See Topic: INTRODUCTION, Short Answer Type Question No. 8.
c) See Topic: ER MODEL, Short Answer Type Question No. 14.

9. What do you mean by Super key, Candidate key and Primary key? Take a single example of a database and explain the relationship between primary key, candidate key, foreign key in the same example.
See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Long Answer Type Question No. 6.

10. a) Explain with two examples why the set $\{\sigma, \pi, \cup, -, X\}$ is called the complete set of relational algebra operation.
b) Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number or recorded accidents. State all your assumptions.
a) See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Long Answer Type Question No. 7.
b) See Topic: ER MODEL, Long Answer Type Question No. 6.

11: Write short notes on any three of the following:
- a) Role of DBA in database design.
- b) Three-level architecture of DBMS.
- c) Query language
- d) B+ tree
- e) Logical and physical data independence.

a) See Topic: INTRODUCTION, Long Answer Type Question No. 5(c).
b) See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Long Answer Type Question No. 4(a).
c) See Topic: SQL, Long Answer Type Question No. 7.

d) OUT OF SYLLABUS
e) See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Long Answer Type Question No. 4(b).

QUESTION 2016

GROUP – A
(Multiple Choice Type Questions)

1. Choose the correct alternatives for any *ten* of the following:

i) If a set of attributes K, in a relation to schema R1 is a foreign key to R1 then
a) every tuple of R1 has a distinct value for K
✓b) K is key for some other relation
c) K cannot have a null value for tuples in R1
d) K is a primary key for R1

ii) Which of the following features is supported in relational database model?
a) complex data types
b) multi-valued attributes
✓c) associations with multiplicates
d) generalization relationships

iii) SQL is a
a) procedural language
✓b) non-procedural language
c) complex language
d) none of these

iv) The entity integrity constraint states
✓a) no primary key value can be null
b) a part of the key may be null
c) duplicate object values are allowed
d) none of these

v) What is the default format of date in Oracle?
a) dd-mm-yy
b) dd-m-yyyy
✓c) dd-mon-yy
d) none of these

vi) Which of the following aggregate functions works with characters?
a) max
b) avg
c) count
✓d) none of these

vii) If we do not specify the constraint name for a constraint, the default name is in the format
✓a) SYS_Cn
b) Cn_SYS
c) Cn
d) none of these

viii) The information about data in a database is called
a) tera data
✓b) meta data
c) hyper data
d) none of these

ix) Which data abstraction level specifies how data are stored in database?
✓a) physical
b) logical
c) view
d) none of these

x) In transaction, a WRITE operation will
✓a) read data from table and write on buffer
b) write on table
c) depend on application
d) lock a data for updating

GROUP – B
(Short Answer Type Questions)

2. Tables:

Student

Roll(PK)	Name	Dept id (PK)
1	ABC	1
2	DEF	1
3	GHI	2
4	JKL	3

Department

Dept id (PK)	Dept_Name
1	A
2	B
3	C

What will happen if we try to execute the following two SQL statements?
Give proper explanation for your answer.
a) Update Student set Dept id = Null where Roll no = 1;
b) Update Department set Dept id = Null where Dept id = 1;
See Topic: ER MODEL, Short Answer Type Question No. 11.

3. Let R (a, b, c) and S (d, e, f) be two relations in which d is the FK for S that refers to the primary key of R. Which of the following is true about the Referential integrity constraint? Give proper explanation for your answer for choosing or not choosing each of the option.
a) Insert into R b) Insert into S c) Delete from R d) Delete from S.
See Topic: ER MODEL, Short Answer Type Question No. 12.

4. Let E1 and E2 be two entities in an E-R diagram with simple valued attributes. R1 and R2 are two relations between E1 and E2, where R1 is none-to-many and R2 is many-to-many. R1 and R2 do not have any attributes of their own. What is the minimum number of tables required to represent the situation in the relational model? Give proper explanation for your answer.
See Topic: ER MODEL, Short Answer Type Question No. 10.

5. 'All primary keys are super key but the converse is not true'. Explain with example.
See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Short Answer Type Question No. 3(a).

GROUP – C
(Long Answer Type Questions)

6. a) Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.
b) What is Metadata? Explain with the help of a example.
c) Differentiate between Delete and Truncate operations.
a) See Topic: ER MODEL, Long Answer Type Question No. 7.
b) See Topic: INTRODUCTION, Long Answer Type Question No. 4.
c) See Topic: SQL, Short Answer Type Question No. 6(a).

7. Consider the following schemas:
Employee_master (EmpNo, Name, Job, Hiredate, Salary, manager_id, Dept_no, Age, E_sal)
Perform the following queries on table (write appropriate SQL statement) (any five)
a) List all employees' names and jobs whose job includes 'M' or 'P'.
b) List all employees' names and their salaries whose salary lies between 15000 and 35000. (using between clause)
c) List all employees' names, salaries and 25% raise in salary.
d) Find how much amount the company is spending towards salary head.
e) List all employees' names and their manager_id whose manager_id is 7902, 7566 or 7789.
f) List the difference between minimum and maximum salaries of employees.
See Topic: SQL, Long Answer Type Question No. 5.

8. What is functional dependency? Explain with example. Define 1NF, 2nd NF, 3rd NF and BCNF with example.
1st part: See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Long Answer Type Question No. 3.

2nd part: See Topic: NORMALIZATION, Short Answer Type Question No. 1(b) & 2.

9. a) Write and explain GROUP BY, LIKE, DISTINCT, INNER JOIN and UPDATE commands in SQL. Also give one example for each.
b) Explain ACID properties.
a) See Topic: SQL, Long Answer Type Question No. 4.
b) See Topic: TRANSACTION MANAGEMENT, Short Answer Type Question No. 1.

10. Write short notes on any three of the following:
a) Multi-level index
b) Logical data independence and physical data independence
c) Codd's rule
d) The three-level architecture of DBMS
e) Query optimization.
a) See Topic: INDEXING, Long Answer Type Question No. 6(b).
b) See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Long Answer Type Question No. 4(b).
c) See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Long Answer Type Question No. 9(b).
d) See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Long Answer Type Question No. 4(a).
e) See Topic: FILE ORGANIZATION & QUERY OPTIMIZATION, Long Answer Type Question No. 2(b).

QUESTION 2017

GROUP – A
(Multiple Choice Type Questions)

1. Choose the correct alternatives for any *ten* of the following:
i) Relational algebra is a ..... language.
a) non-procedural ✓b) procedural c) programming d) none of these

ii) Which of the following clauses is used to enforce a condition on a SQL statement containing "group by" clause?
a) Where ✓b) Having c) Order by d) none of these

iii) What is the cardinality of a table with 100 rows and 10 columns?
a) 1000 ✓b) 100 c) 10 d) 10000

iv) The main goal of indexing is to
✓a) search an item faster from a table b) insert an item faster into a table
c) delete an item faster into a table d) none of these

v) The collection of information stored in a database at a particular moment is called as
a) Schema ✓b) instance of the database
c) data domain d) independence

vi) Grant and revoke are ..... statements
a) DDL ✓b) TCL c) DCL d) DML

vii) Referential integrity is directly related to
a) relational key ✓b) Foreign key c) Primary key d) Candidate key

viii) Generalization is a ..... approach
✓a) bottom up b) top down c) both (a) & (b) d) none of these

ix) Any relation that is not part of the logical model but is made visible to a user as a virtual relation, is called as
a) relation ✓b) view c) tuple d) none of these

x) Normalization removes
a) dependency of data b) uniqueness of data
✓c) redundancy of data d) none of these

xi) Which is the SQL command to remove rows from a table?
a) REMOVE ✓b) DELETE c) TRUNCATE d) all of these

GROUP – B
(Short Answer Type Questions)

2. Explain the different levels of abstraction of the data base management system
See Topic: INTRODUCTION, Short Answer Type Question No. 8.

3. What is constraint? Explain domain constraint and Entity Integrity constraint.
See Topic: ER MODEL, Short Answer Type Question No. 15.

4. What is Relationship? Explain different degrees of relationship.
See Topic: ER MODEL, Short Answer Type Question No. 13.

5. "All primary keys are the super keys but converse is not true."– Clarify. Define multi-valued attribute and composite attribute with suitable example.
See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Short Answer Type Question No. 3(a).

6. Consider the following tables with their functional dependencies:
Professor (Professor_code) → (Head_of_dept, Percent_time)
(Department, Professor_code) → (Head_of_dept, Percent_time)
(Department) → (Head_of_dept)
(Head_of_dept, Professor_code) → (Department, Percent_time)
It is assumed that –
i) A professor can work in more than one department
ii) The percentage of the time he spends in each department is given
iii) Each department has one Head_of_dept
Normalize the table up to BCNF
See Topic: NORMALIZATION, Short Answer Type Question No. 10.

GROUP – C
(Long Answer Type Questions)

7. a) Explain the ACID properties for a transaction.
b) Explain all the states of a transaction with example for each state.
c) What is a schedule? Give an example of a serial schedule with two transactions.

a) See Topic: TRANSACTION MANAGEMENT, Short Answer Type Question No. 1.
b) See Topic: TRANSACTION MANAGEMENT, Short Answer Type Question No. 2.
c) See Topic: TRANSACTION MANAGEMENT, Short Answer Type Question No. 3.

8. Consider the following two schemas:
EMPLOYEE (EMP#, ENAME, JOB, HIREDATE, MANAGER#, SALARY, COMM, DEPT#)
DEPARTMENT (DEPT#, DNAME, LOCATION)
Perform the following queries on the tables (write appropriate SQL statement):
i) List the name, salary and PF amounts of all employees (PF is calculated at 10% of the basic)
ii) List the number of employees and average salary in DEPT#20
iii) List the department number and total salary payable in each department
iv) List the names of the employees who are more than 20 years old in the company
v) List the names of the employee whose name either starts or ends with 'S'.
See Topic: SQL, Long Answer Type Question No. 1.

9. a) Differentiate between hierarchical, network and relational model.
b) Draw an E-R Diagram for a library management system.
c) Explain the following terms with example: Aggregation, Specialization, Generalization, Derived Attribute, Unary Relationship
a) See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Long Answer Type Question No. 1(a).
b) See Topic: ER MODEL, Long Answer Type Question No.8.
c) Aggregation, Specialization, Generalization, Derived Attribute:
See Topic: ER Model, Long Answer Type Question No. 10.
Unary Relationship: See Topic: ER MODEL, Short Answer Type Question No. 13.

10. a) Proof with an example that a relation in BCNF is in 3NF, but the converse is not true.
b) Find out the candidate Keys for the following relation R:
R (A,B,C,D,E,H) , F = {A→B, BC→D, E→C, D→A}
c) For relation R(L, M, N, O, P) the following FD's hold:
M→O, NO→P, P→L, L→MN
R is decomposed into R1 = (L, M, N, P) and R2 = (M, O)
i) Is the above decomposition lossless-join decomposition? Explain
ii) Is the above decomposition dependency preserving? Explain.
a) See Topic: NORMALIZATION, Short Answer Type Question No. 14.
b) See Topic: ER MODEL, Long Answer Type Question No. 9.
c) See Topic: NORMALIZATION, Long Answer Type Question No. 10.

11. Write the short notes any three of the following:
a) Primary Indexing
b) Database approach and the file based approach
c) Natural join and Equi join
d) B-Tree
e) Strong entity and weak entity
a) See Topic: INDEXING, Long Answer Type Question No. 6(c).
b) See Topic: FILE ORGANIZATION & QUERY OPTIMIZATION, Long Answer Type Question No. 2(d).
c) See Topic: SQL, Long Answer Type Question No. 9 (c) & (d).
d) OUT OF SYLLABUS
e) See Topic: ER MODEL, Long Answer Type Question No. 13(b).

QUESTION 2018

GROUP – A
(Multiple Choice Type Questions)

1. Choose the correct alternatives from any ten of the following:
i) A relational database consists of a collection of
✓a) tables b) fields c) records d) keys

ii) Suppose R is a relation of n attributes $\{A_1, A_2, \ldots, A_n\}$ as a function of n. How many super keys R has if the only key is A_1 ?
a) $2^* n$ b) $2^*(n-1)$ ✓c) $2^{(n-1)}$ d) None of these

iii) Identify the correct statement:
✓a) Physical level Abstraction : Describes how a record is stored.
b) Physical level Abstraction : Describe how schema is stored in a data base.
c) Physical level Abstraction : Hides details of data types.
d) None of the above

iv) Consider the following SQL statements.
S1:
INSERT INTO employees (first_name,last_name,fname)
VALUES ('John', 'Capita', 'xcapit00');
S2:
SELECT instructor. ID, department.dept_name FROM instructor, department
WHERE instructor. dept_name = department. dept_name AND department. budget > 95000;
Identify the correct statement related to S1 and S2:
a) Both S1 and S2 are Data Definition (DDL) Queries.
b) S1 is a Data Control Query and S2 is a Data Manipulation (DML) Query
✓c) Both S1 and S2 are Data Manipulation (DML) Queries
d) S1 is a Data Definition (DDL) Query and S2 is a Data Control Query

v) To remove a relation from an SQL database, we use the _____ command.
a) delete b) purge c) remove ✓d) drop

vi) For relations r_1 (A, B, C), r_2 (C, D, E) and r_3 (E, F), assume r_1 has 1000 tuples and r_2 has 1500 tuples and r_3 has 750 tuples. Maximum size of $r_1 \bowtie r_2 \bowtie r_3$ is
✓a) 1000 tuple b) 750 tuple c) 1500 tuple d) 1500750 tuple

vii) The descriptive property possessed by each entity set is
a) entity b) relation c) model ✓d) attribute

viii) The entity set person is classified as student and employee. This process is called _____
a) Generalization ✓b) Specialization
c) Inheritance d) Constraint generalization

ix) Which form has a relation that possesses data about an individual entity?
a) 2NF b) 3NF ✓c) 4NF d) 5NF

x) Consider the following R1 relation and functional dependencies.
R1 (a, b, c, d, e, f)
FD1 : a → b, c, d, f
FD2 : d → f
a) relation R1 is in Boyce Codd normal form.
b) relation R1 is in Third normal form.
✓c) relation R1 is not in Third normal form due to transitive dependency
d) None of the above

xi) Which of these is not a feature of hierarchical model?
a) Organizes the data in tree-like structure.
b) Parent node can have any number of child nodes
c) Root node does not have any parent.
✓d) Child node can have any number of parent nodes.

Group – B
(Short Answer Type Questions)

2. Explain the difference between external, internal and conceptual schemas. Distinguish between logical and physical data independence.
1st Part: See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Short Answer Type Question No. 3.
2nd Part: See Topic: DATA MODELS & ARCHITECTURE OF DBMS, Short Answer Type Question No. 1(b).

3. "All super keys are not candidate keys but the vice-versa is true". — Justify the statement.
See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Short Answer Type Question No. 6.

4. Express division operation in terms of basic relational algebra operations.
See Topic: ER MODEL, Short Answer Type Question No. 16.

5. What are the different parameters to evaluate the merits and demerits of Indexing and Hashing Schemes for database applications?
See Topic: INDEXING, Short Answer Type Question No. 3.

6. What is meant by spurious tuple?
Consider the following relational schema:
r(A, B, C, D, E, F, G)
The following functional dependency is held in the relation
A – > B ; B – > C ; C – > D, E ; F – > G
Maintaining the functional dependency the relation r is broken as follows:
r1(A, B) ; r2(B, C) ; r3 (C, D, E) ; r4 (F, G)
Verify whether this decomposition is lossless or not.
See Topic: NORMALIZATION, Short Answer Type Question No. 16.

GROUP – C
(Long Answer Type Questions)

7. a) With proper diagram explain extended ER features (Generalization, Specialization and Aggregation).
See Topic: ER MODEL, Short Answer Type Question No. 14.

b) Draw a sample ER diagram for a college administration system. It should keep information like name and contact number of employees and students, attendants, salary statement of employees, department wise class and room allotment and examination result of students
See Topic: ER MODEL, Short Answer Type Question No. 11.

8. a) Discuss about possible situations where hashing is preferred over indexing to search values.
See Topic: INDEXING, Long Answer Type Question No. 5(a).

b) Discuss about possible situations where is indexing preferred over hashing to search values.
See Topic: INDEXING, Long Answer Type Question No. 5.b).

c) Why is secondary index defined? Discuss some of the possible applications.
See Topic: INDEXING, Long Answer Type Question No. 5.c).

d) Show the steps of Insertion in a B+ Tree of order 3 for following values:
80, 50, 20, 70, 30, 100, 120, 60
See Topic: INDEXING, Long Answer Type Question No. 5.d).

9. a) Explain different types of database anomalies and how normalization removes them.
See Topic: NORMALIZATION, Long Answer Type Question No. 11.

b) Write the algorithm to find out the candidate key from given a relation.
See Topic: RELATIONAL MODEL & RELATIONAL DATABASE DESIGN, Short Answer Type Question No. 7.

c) Consider the following relation:

$$r(R) = \{A, B, C, D, E, F, G, H, I\}$$

Functional dependency is given below:

$$F = \{A \rightarrow B,\ C \rightarrow DE,\ F \rightarrow G,\ B \rightarrow GH,\ AF \rightarrow C,\ E \rightarrow I\}$$

Determine the current normal form of the given relation. Decompose it up to BCNF.
See Topic: NORMALIZATION, Short Answer Type Question No. 17.

d) Explain with example why BCNF is stricter than 3NF?
See Topic: NORMALIZATION, Short Answer Type Question No. 14 (2nd Part).

10. Consider the following two schemas:
EMPLOYEE (EMP_ID, FNAME, ADDRESS, JOIN_DATE, SALARY, MANAGER#, DEPT_ID).
DEPT(DEPT_ID, DNAME, LOCATION).
Write appropriate SQL statements based on above tables.
a) List the details of employees whose salary is less than the average salary.
b) List the department id and the number of employees working in that department.
c) List the name of employees whose name have exactly five letters.
d) List the details of employees who are more than 10 years old in the company.
e) Display the minimum and maximum salary of the employee.
See Topic: SQL, Long Answer Type Question No. 6.

11. Write short notes on any three of the following:
a) Query Optimization Technique
b) Inner Join and Outer Join
c) Selection and Projection
d) Multi-level Index