

Internal Assessment Test II – November 2016

SCHEME

<b>Sub:</b>	<b>Data Base Management Systems</b>						<b>Code:</b>	10CS54	
<b>Date:</b>	03/11/2016	<b>Duration:</b>	90 mins	<b>Max Marks:</b>	50	<b>Sem:</b>	V	<b>Branch:</b>	CSE

<b>Note: Answer any five full questions</b>		<b>Marks</b>	<b>OBE</b>	
			<b>CO</b>	<b>RBT</b>
1	a) How is a view created and dropped? What problems are associated with updating of views?	2M+2M+1M	CO5	L1
	b) With program segments, explain retrieving of tuples (single and multiple) using embedded SQL.	2.5M+2.5M	CO5	L3
2	a) Describe how triggers and assertions are defined in SQL? Give examples.	2.5M+2.5M	CO5	L2
	b) Explain the concept of stored procedures using example.	5M	CO5	L2
3	a) List the inference rules for functional dependencies. Write the algorithm to determine the closure of X (set of attributes) under F (set of functional dependencies) with an example.	2.5M+2.5M	CO6	L1
	b) Explain the algorithm for ER-to- Relational Mapping	5M	CO3	L1
4	a) Describe the syntax of SELECT statement in SQL.	4M	CO5	L1
	b) What are insertion, deletion and modification anomalies. Describe with examples.	2M+2M+2M	CO6	L2
5	a) Consider $R = \{A B C D E F\}$ ; FDs $\{A \rightarrow BC, B \rightarrow E, CD \rightarrow EF\}$ Show that $AD \rightarrow F$ .	3M	CO6	L3
	b) What is the difference between the WHERE and HAVING clause?	2M+2M	CO6	L2
	c) Given below are 2 sets of Functional Dependencies. Are they equivalent? i) $A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E$ ii) $A \rightarrow BC, D \rightarrow AE$	3M	CO6	L3
6	Consider the following relations for a database. Supplier (Sno, Sname, Status, City) Product (Pno, Pname, Color, Weight, City) Shipments (Sno, Pno, Qty) <b>Solve the following questions using SQL.</b> i. Retrieve names of supplier who supply product # P2. ii. Retrieve the names of suppliers who do not supply any product supplied by Supplier # S2. iii. Retrieve product number for all products supplied by more than one supplier. iv. For each product supplied, get the product number, maximum quantity, minimum quantity supplied for that product. v. Retrieve supplier numbers for suppliers with status less than the current maximum in the supplier table.	2M * 5	CO6	L3
7	State the informal guidelines for relational schema design. Illustrate how violation of these guidelines may be harmful.	2.5M+2.5M+2.5M+2.5M	CO6	L2
8	a) Write note on Aggregate function in SQL with example.	3M+2M	CO6	L2
	b) Explain with an example, the basic constraints that can be specified when you create a table in SQL.	3M + 2M	CO6	L2

Internal Assessment Test II – November 2016

**SOLUTION**

<b>Sub:</b>	Data Base Management Systems					<b>Code:</b>	10CS54		
<b>Date:</b>	03/11/2016	<b>Duration:</b>	90 mins	<b>Max Marks:</b>	50	<b>Sem:</b>	V	<b>Branch:</b>	CSE

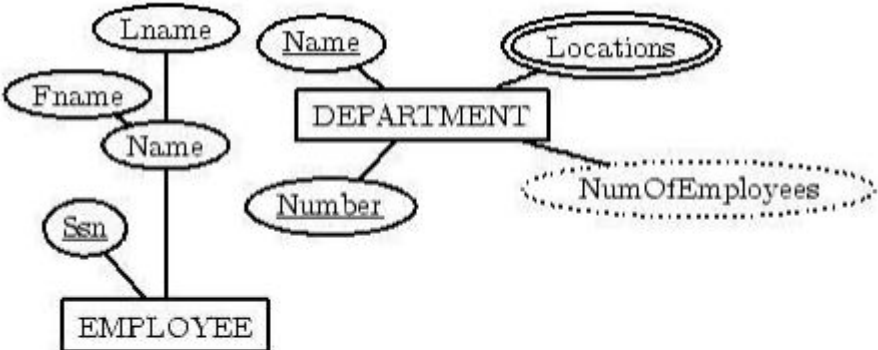
Questions and Answers.		Marks	OBE	
			CO	RBT
1	<p>c) How is a view created and dropped? What problems are associated with updating of views?</p> <p>A view refers to a single table that is derived from other tables</p> <p>Eg:</p> <pre>CREATE VIEW WORKS_ON1 AS SELECT FNAME, LNAME, PNAME, HOURS FROM EMPLOYEE, PROJECT, WORKS_ON WHERE SSN=ESSN AND PNO=PNUMBER</pre> <pre>CREATE VIEW DEPT_INFO(DEPT_NAME, NO_OF_EMPLS, TOTAL_SAL) AS SELECT DNAME, COUNT(*), SUM(SALARY) FROM DEPARTMENT, EMPLOYEE WHERE DNUMBER=DNO GROUP BY DNAME</pre>	2M+2M+1M	CO5	L1

<p>A View is always up to date; A view is realized at the time we specify(or execute) a query on the view</p> <p><b>DROP VIEW WORKS_ON1: TO DROP A VIEW.</b></p> <p>Updating of Views Updating the views can be complicated and ambiguous In general, an update on a view on defined on a single table w/o any aggregate functions can be mapped to an update on the base table</p> <ul style="list-style-type: none"> <li>• A view with a single defining table is updatable if we view contain PK or CK of the base table</li> <li>• View on multiple tables using joins are not updatable</li> <li>• View defined using grouping/aggregate are not updatable</li> </ul>			
<p>b) With program segments, explain retrieving of tuples (single and multiple) using embedded SQL.</p> <p>SQL can also be used in conjunction with a general purpose programming language, such as PASCAL, COBOL, or PL/I. The programming language is called the host language. The embedded SQL statement is distinguished from programming language statements by prefixing it with a special character or command so that a preprocessor can extract the SQL statements. In PL/I the keywords EXEC SQL precede any SQL statement. In some implementations, SQL statements are passed as parameters in procedure calls. We will use PASCAL as the host programming language, and a "\$" sign to identify SQL statements in the program. Within an embedded SQL command, we may refer to program variables, which are prefixed by a "%" sign. The programmer should declare program variables to match the data types of the database attributes that the program will process. These program variables may or may not have names that are identical to their corresponding attributes. Example: Write a program segment (loop) that reads a social security number and prints out some information from the corresponding EMPLOYEE tuple E1: LOOP:= 'Y'; while LOOP = 'Y' do begin writeln('input social security number:');</p>	2.5M+2.5M	CO5	L3

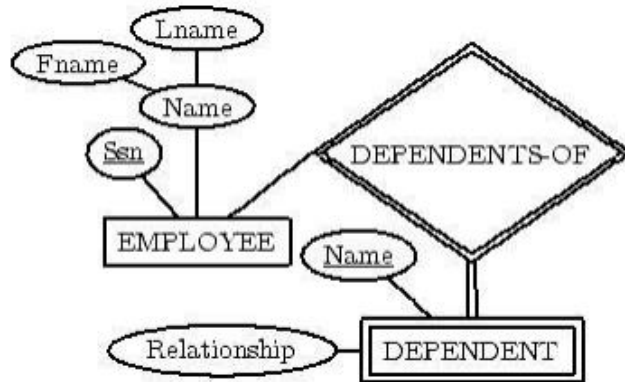
	<pre> readln(SOC_SEC_NUM); \$SELECT FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SALARY INTO %E.FNAME, %E.MINIT, %E.LNAME, %E.SSN, %E.BDATE, %E.ADDRESS, %E.SALARY FROM EMPLOYEE WHERE SSN=%SOC_SEC_NUM ; writeln( E.FNAME, E.MINIT, E.LNAME, E.SSN, E.BDATE, E.ADDRESS, E.SALARY); writeln('more social security numbers (Y or N)? '); readln(LOOP) end; In E1, a single tuple is selected by the embedded SQL query; that is why we are able to assign its attribute values directly to program variables. In general, an SQL query can retrieve many tuples. The concept of a cursor is used to allow tuple-at-a-time processing by the PASCAL program CURSORS: We can think of a cursor as a pointer that points to a single tuple (row) from the result of a query. The cursor is declared when the SQL query command is specified. A subsequent OPEN cursor command fetches the query result and sets the cursor to a position before the first row in the result of the query; this becomes the current row for the cursor. Subsequent FETCH commands in the program advance the cursor to the next row and copy its attribute values into PASCAL program variables specified in the FETCH command. An implicit variable SQLCODE communicates to the program the status of SQL embedded commands. An SQLCODE of 0 (zero) indicates successful execution. Different codes are returned to indicate exceptions and errors. A special END_OF_CURSOR code is used to terminate a loop over the tuples in a query result. A CLOSE cursor command is issued to indicate that we are done with the result of the query. When a cursor is defined for rows that are to be updated the clause FOR UPDATE OF must be in the cursor declaration, and a list of the names of any attributes that will be updated follows. The condition WHERE CURRENT OF cursor specifies that the current tuple is the one to be updated (or deleted) Example: Write a program segment that reads (inputs) a department name, then lists the names of employees who work in that department, one at a time. The program reads a raise amount for each employee and updates the employee's salary by that amount. E2: writeln('enter the department name:'); readln(DNAME); \$SELECT DNUMBER INTO %DNUMBER FROM DEPARTMENT WHERE DNAME=%DNAME; \$DECLARE EMP CURSOR FOR SELECT SSN, FNAME, MINIT, LNAME, SALARY FROM EMPLOYEE WHERE DNO=%DNUMBER FOR UPDATE OF SALARY; \$OPEN EMP; \$FETCH EMP INTO %E.SSN, %E.FNAME, %E.MINIT, %E.LNAME, %E.SAL </pre>			
2	c) Describe how triggers and assertions are defined in SQL? Give examples.	2.5M+2.5M	CO5	L2

<p>Specifying General Constraints Users can specify certain constraints such as semantics constraints  CREATE ASSERTION SALARY_CONSTRAINT CHECK ( NOT EXISTS ( SELECT * FROM EMPLOYEE E, EMPLOYEE M, DEPARTMENT D WHERE E.SALARY &gt; M. SALARY AND E.DNO=D.NUMBER AND D.MGRSSN=M.SSN</p>			
<p>d) Explain the concept of stored procedures using example.</p> <p><u>Database stored procedures and SQL/PSM</u></p> <ul style="list-style-type: none"> <li>• Persistent procedures/functions (modules) are stored locally and executed by the database server. As opposed to execution by clients .</li> <li>• Advantages: If the procedure is needed by many applications, it can be invoked by any of them (thus reduce duplications)</li> <li>• Execution by the server reduces communication costs</li> <li>• Enhance the modeling power of views</li> <li>• Disadvantages: Every DBMS has its own syntax and this can make the system less portable</li> <li>• A stored procedure EXAMPLE</li> <li>• CREATE PROCEDURE procedure-name (params)  local-declarations  procedure-body;</li> <li>• A stored function  CREATE FUNCTION fun-name (params)  RETRUNS return-type  local-declarations function-body;</li> </ul> <p>Calling a procedure or function  CALL procedure-name/fun-name (arguments);  E.g.,  CREATE FUNCTION DEPT_SIZE (IN deptno INTEGER)  RETURNS VARCHAR[7]  DECLARE TOT_EMPS INTEGER;</p>	5M	CO5	L2

	<pre>SELECT COUNT (*) INTO TOT_EMPS FROM SELECT EMPLOYEE WHERE DNO = deptno; IF TOT_EMPS &gt; 100 THEN RETURN —HUGE   ELSEIF TOT_EMPS &gt; 50 THEN RETURN —LARGE   ELSEIF TOT_EMPS &gt; 30 THEN RETURN —MEDIUM   ELSE RETURN —SMALL   ENDIF;</pre>			
3	<p>c) List the inference rules for functional dependencies. Write the algorithm to determine the closure of X(set of attributes) under F(set of functional dependencies) with an example.</p> <p><u>Inference Rules</u></p> <p>IR1 (reflexive rule)<sup>1</sup>: If <math>X \supseteq Y</math>, then <math>X \rightarrow Y</math>.</p> <p>IR2 (augmentation rule)<sup>2</sup>: <math>\{X \rightarrow Y\} \models XZ \rightarrow YZ</math>.</p> <p>IR3 (transitive rule): <math>\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z</math>.</p> <p>IR4 (decomposition, or projective, rule): <math>\{X \rightarrow YZ\} \models X \rightarrow Y</math>.</p> <p>IR5 (union, or additive, rule): <math>\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ</math>.</p> <p>IR6 (pseudotransitive rule): <math>\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z</math>.</p> <p><u>Closure of a Set of Functional Dependencies</u></p> <p>Given a set X of FDs in relation R, the set of all FDs that are implied by X is called the closure of X, and is denoted <math>X^+</math>. Algorithms for determining <math>X^+</math> <math>X^+ := X</math>; repeat old<math>X^+ := X^+</math> for each FD <math>Y \rightarrow Z</math> in F do if <math>Y \subseteq X^+</math> then <math>X^+ := X^+ \cup Z</math>; until old<math>X^+ = X^+</math> ;</p> <p>Example: <math>A \rightarrow BC</math>  <math>E \rightarrow CF</math>  <math>B \rightarrow E</math>  <math>CD \rightarrow EF</math></p> <p>Compute <math>\{A, B\}^+</math> of the set of attributes under this set of FDs. Solution: Step1: <math>\{A, B\}^+ := \{A, B\}</math>. Go round the inner loop 4 time, once for each of the given FDs. On the first iteration, for A BC A <math>\{A, B\}^+ \{A, B\}^+ := \{A, B, C\}</math></p>	2.5M+2.5M	CO6	L1

<p>Step2: On the second iteration, for <math>ECF, \notin \{A, B, C\}</math>  Step3 :On the third iteration, for <math>B \in B \{A, B,C\} + \{A, B\} + := \{A, B, C, E\}</math>.  Step4: On the fourth iteration, for <math>CD \rightarrow EF</math> remains unchanged. Go round the inner loop 4 times again. On the first iteration result does not change; on the second it expands to <math>\{A,B,C,E,F\}</math>; On the third and forth it does not change. Now go round the inner loop 4 times. Closure does not change and so the whole process terminates, with <math>\{A,B\} + = \{A,B,C,E,F\}</math></p>								
<p>d) Explain the algorithm for ER-to- Relational Mapping</p> <p><u>Relational Database Design Using ER-to-Relational Mapping</u>  Step 1: For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E</p>  <pre> graph TD     EMPLOYEE[EMPLOYEE] --- Lname((Lname))     EMPLOYEE --- Fname((Fname))     EMPLOYEE --- Name1((Name))     EMPLOYEE --- Ssn((Ssn))     DEPARTMENT[DEPARTMENT] --- Name2((Name))     DEPARTMENT --- Locations((Locations))     DEPARTMENT --- Number((Number))     DEPARTMENT --- NumOfEmployees((NumOfEmployees))     EMPLOYEE --- NumOfEmployees   </pre> <p>EMPLOYEE</p> <table border="1" data-bbox="324 1109 571 1197"> <tr> <td>SSN</td> <td>Lname</td> <td>Fname</td> </tr> </table> <p>DEPARTMENT</p> <table border="1" data-bbox="324 1236 560 1324"> <tr> <td>NUMBER</td> <td>NAME</td> </tr> </table>	SSN	Lname	Fname	NUMBER	NAME	5M	CO3	L1
SSN	Lname	Fname						
NUMBER	NAME							

Step 2: For each weak entity type W in the ER schema with owner entity type E, create a relation R, and include all simple attributes (or simple components of composite attributes) of W as attributes. In addition, include as foreign key attributes of R the primary attribute(s) of the relation(s) that correspond to the owner entity type(s)

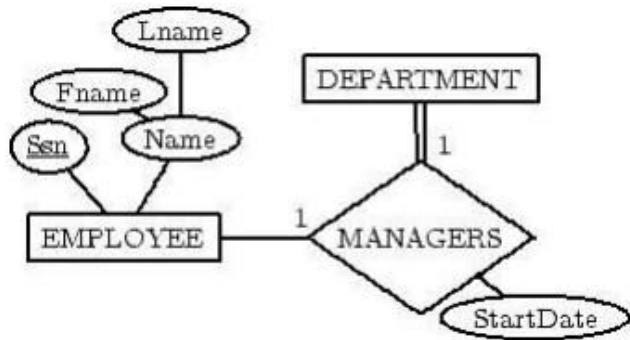


DEPENDENT

EMPL-SSN	NAME	Relationship
----------	------	--------------

Step 3: For each **binary 1:1 relationship type** R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. Choose one of the relations, say S, and include the primary key of T as a foreign key in S. Include all the simple attributes of R as attributes of S.

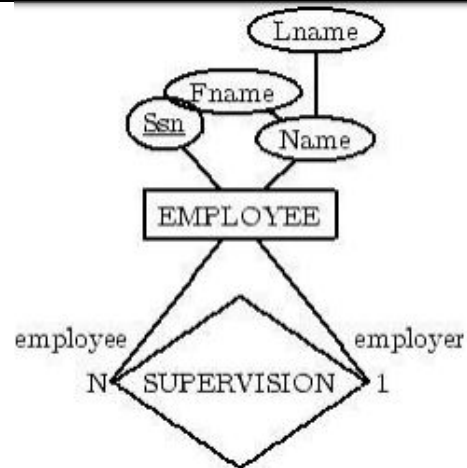




DEPARTMENT

MANAGER-SSN	StartDate
-------------	-----------

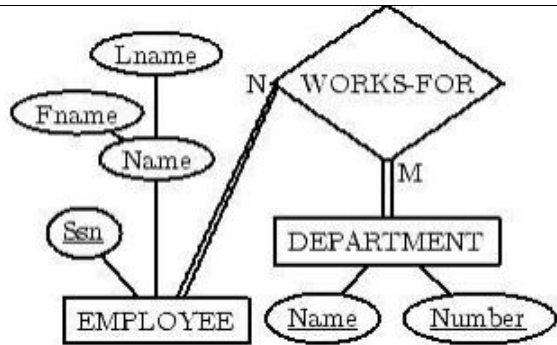
Step 4: For each regular **binary 1:N relationship type** R identify the relation (N) relation S. Include the primary key of T as a foreign key of S. Simple attributes of R map to attributes of S.



EMPLOYEE

SupervisorSSN
---------------

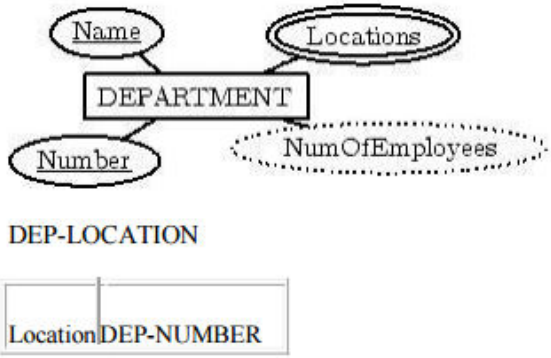
Step 5: For each **binary M:N relationship type R**, create a relation S. Include the primary keys of participant relations as foreign keys in S. Their combination will be the primary key for S. Simple attributes of R become attributes of S.



WORKS-FOR

<u>EmployeeSSN</u>	<u>DeptNumber</u>
--------------------	-------------------

Step 6: For each **multi-valued attribute A**, create a new relation R. This relation will include an attribute corresponding to A, plus the primary key K of the parent relation (entity type or relationship type) as a foreign key in R. The primary key of R is the combination of A and K.

	 <p>DEP-LOCATION</p> <p>Location DEP-NUMBER</p> <p>Step 7: For each <b>n-ary relationship type R</b>, where <math>n &gt; 2</math>, create a new relation S to represent R. Include the primary keys of the relations participating in R as foreign keys in S. Simple attributes of R map to attributes of S. The primary key of S is a combination of all the foreign keys that reference the participants that have cardinality constraint <math>&gt; 1</math>.</p> <p>For a recursive relationship, we will need a new relation.</p>			
4	<p>c) Describe the syntax of SELECT statement in SQL.</p> <pre> <b>SELECT</b> &lt;attribute and function list&gt; <b>FROM</b> &lt;table list&gt; [ <b>WHERE</b> &lt;condition&gt; ] [ <b>GROUP BY</b> &lt;grouping attribute(s)&gt; ] [ <b>HAVING</b> &lt;group condition&gt; ] [ <b>ORDER BY</b> &lt;attribute list&gt; ]; </pre>	4M	CO5	L1
	<p>d) What are insertion, deletion and modification anomalies. Describe with examples.</p> <p><u>Insertion Anomalies</u> To insert a new employee tuple into EMP_DEPT, we must include either the attribute</p>	2M+2M+2M	CO6	L2

	<p>values for that department that the employee works for, or nulls. It's difficult to insert a new department that has no employee as yet in the EMP_DEPT relation. The only way to do this is to place null values in the attributes for employee. This causes a problem because SSN is the primary key of EMP_DEPT, and each tuple is supposed to represent an employee entity - not a department entity.</p> <p><u>Deletion Anomalies</u></p> <p>If we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database.</p> <p><u>Modification Anomalies</u></p> <p>In EMP_DEPT, if we change the value of one of the attributes of a particular department- say the manager of department 5- we must update the tuples of all employees who work in that department.</p>			
5	d) Consider $R = \{A B C D E F\}$ ; $FDs \{A \rightarrow BC, B \rightarrow E, CD \rightarrow EF\}$ Show that $AD \rightarrow F$ .	3M	CO6	L3

<p>5a) <u>Sol<sup>n</sup></u>: Given <math>A \rightarrow BC \rightarrow \textcircled{1}</math>  <math>B \rightarrow E \rightarrow \textcircled{2}</math>  <math>CD \rightarrow EF \rightarrow \textcircled{3}</math></p> <p>Applying Decomposition rule on <math>\textcircled{1}</math>, we get  <math>A \rightarrow B</math>  <math>A \rightarrow C \rightarrow \textcircled{4}</math></p> <p>Applying pseudo transitive rule on <math>\textcircled{3}</math> using <math>\textcircled{1}</math>, we get  <math>AD \rightarrow EF \rightarrow \textcircled{5}</math></p> <p>Applying Decomposition rule on <math>\textcircled{5}</math>, we get  <del><math>AD \rightarrow E</math></del>  <div style="border: 1px solid black; padding: 2px; display: inline-block;"><math>AD \rightarrow F</math></div></p>			
<p>e) What is the difference between the WHERE and HAVING clause?</p> <p>WHERE is used in select statement to identifies the tuples to be retrieved by the query. This clause takes conditions for selection. A missing WHERE-clause indicates no condition on tuple selection; hence, all tuples of the relation specified in the FROM-clause qualify and are selected for the query result. WHERE clause can also be used for nested queries.</p> <p>HAVING provides a condition on the group of tuples associated with each value of the grouping attributes; and only the groups that satisfy the condition are retrieved in the result of the query. The rule is that the WHERE-clause is executed first, to select individual tuples; the HAVING-clause is applied later, to select individual groups of tuples.</p>	2M + 2M	CO6	L2

f) Given below are 2 sets of Functional Dependencies. Are they equivalent?

i)  $A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E$  ii)  $A \rightarrow BC, D \rightarrow AE$

Sol<sup>n</sup> 1: Let  $F: \{ A \rightarrow B, AB \rightarrow C \rightarrow \textcircled{1}$   
 $AB \rightarrow C, \rightarrow \textcircled{2}$   
 $D \rightarrow AC, \rightarrow \textcircled{3}$   
 $D \rightarrow E \} \rightarrow \textcircled{4}$

and  $E: \{ A \rightarrow BC, \rightarrow \textcircled{5}$   
 $D \rightarrow AE \} \rightarrow \textcircled{6}$

F covers E:

Applying pseudo transitive rule on  $\textcircled{1}$  &  $\textcircled{2}$ , we get

- $\textcircled{7} - A \rightarrow C$  - Applying reflexive rule on  $\textcircled{7}$
- $\textcircled{8} - A \rightarrow C$  - Applying reflexive rule on  $\textcircled{7}$
- $\textcircled{9} - A \rightarrow BC$  - Applying union rule on  $\textcircled{1}$  &  $\textcircled{8}$
- $\textcircled{10} - D \rightarrow A$  - Applying decomposition on  $\textcircled{3}$
- $\textcircled{11} - D \rightarrow AE$  - Applying union on  $\textcircled{10}$  &  $\textcircled{4}$

$\therefore F$  covers  $E$ .

E covers F:

- $\textcircled{12} - A \rightarrow B$  - Applying decomposition on  $\textcircled{5}$
- $\textcircled{13} - A \rightarrow C$  - " " " "
- $\textcircled{14} - AB \rightarrow CB$  - Augmenting  $\textcircled{13}$  with  $B$ .
- $\textcircled{15} - AA \rightarrow CB$  - Pseudo transitive rule on  $\textcircled{14}$  &  $\textcircled{13}$
- $\textcircled{16} - A \rightarrow CB$  - Reflexive rule on  $\textcircled{15}$
- $\textcircled{17} - D \rightarrow A$  - Applying decomposition on  $\textcircled{11}$
- $\textcircled{18} - D \rightarrow C$  - " " transitivity on  $\textcircled{17}$  &  $\textcircled{13}$
- $\textcircled{19} - D \rightarrow AC$  - Applying union on  $\textcircled{17}$  &  $\textcircled{18}$
- $\textcircled{20} - D \rightarrow AE$  - " " decomposition on  $\textcircled{6}$

3M

CO6

L3

6	<p>Consider the following relations for a database.</p> <p>Supplier (Sno, Sname, Status, City)  Product(Pno, Pname, Color, Weight, City)  Shipments (Sno, Pno, Qty) <b>Solve the following questions using SQL.</b></p> <p>vi. Retrieve names of supplier who supply product # P2.  vii. Retrieve the names of suppliers who do not supply any product supplied by Supplier # S2.  viii. Retrieve product number for all products supplied by more than one supplier.  ix. For each product supplied, get the product number, maximum quantity, minimum quantity supplied for that product.  x. Retrieve supplier numbers for suppliers with status less than the current maximum in the supplier table.</p> <p>Q. (i) <pre>SELECT Sname FROM Supplier SU, Shipment SH WHERE SU.Sno = SH.Sno AND SH.Pno = 'P2';</pre></p> <p>(ii) <pre>SELECT Sname FROM Supplier SUI, Shipment SH1 WHERE SUI.Sno = SH1.Sno AND SH1.PNO NOT IN (SELECT SH2.PNO FROM Shipment SH2 WHERE SH2.Sno = 'S2');</pre></p> <p>iii) <pre>SELECT Sno FROM Shipment GROUP BY PNO HAVING COUNT(Sno) &gt; 1;</pre></p>	2M * 5	CO6	L3
---	--	--------	-----	----



	<p>iv) <code>SELECT Pno, Max(Qty), Min(Qty)</code>  <code>FROM Shipment</code>  <del><code>WHERE</code></del>  <code>GROUP BY Pno;</code></p> <p>v) <code>SELECT Sno</code>  <code>FROM Supplier</code>  <code>WHERE status &lt; (SELECT MAX(status)</code>  <code>FROM Supplier);</code></p>			
7	<p>State the informal guidelines for relational schema design. Illustrate how violation of these guidelines may be harmful.</p> <p><b>GUIDELINE 1:</b> Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation. Intuitively, if a relation schema corresponds to one entity type or one relationship type, the meaning tends to be clear.</p> <p><b>GUIDELINE 2:</b> Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations. If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.</p> <p><b>GUIDELINE 3:</b> As far as possible, avoid placing attributes in a base relation whose values may frequently be null. If nulls are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation.</p> <p>Having too many Null values can be waste space at the storage level and may also lead to problems with understanding the meaning of the attributes and with specifying JOIN operations at the logical level.</p> <p>Another problem with nulls is how to account for them when aggregate operations such as COUNT or SUM are applied. Moreover, nulls can have multiple interpretations,</p>	2.5M+2.5M+ 2.5M+2.5M	CO6	L2

	<p>GUIDELINE 4: Design relation schemas so that they can be JOINed with equality conditions on attributes that are either primary keys or foreign keys in a way that guarantees that no spurious tuples are generated. Do not have relations that contain matching attributes other than foreign key-primary key combinations. If such relations are unavoidable, do not join them on such attributes, because the join may produce spurious tuples.</p>			
8	<p>c) Write note on Aggregate function in SQL with example.          Aggregate function in SQL are : COUNT, SUM, MAX, MIN, and AVG          The functions SUM, MAX, MIN, and AVG are applied to a set or multiset of numeric values.          SUM: used to find sum of the values for an attribute.          MAX: used to find maximum of value for an attribute.          MIN: used to find minimum of value for an attribute.          AVG: used to find average of the values for an attribute.          COUNT: used to count values in a column or number of tuples. COUNT(*) returns number of tuples in a relation.</p> <p>Ex:          SELECT SUM(SALARY), MAX(SALARY), MIN(SALARY), AVG(SALARY), COUNT(EID)          FROM EMPLOYEE;</p>	3M+2M	CO6	L2
	<p>d) Explain with an example, the basic constraints that can be specified when you create a table in SQL.</p> <p>The basic constraints that can be specified when you create a table in SQL are:</p> <ul style="list-style-type: none"> <li>• Not Null</li> <li>• Unique</li> <li>• Default</li> <li>• Primary Key</li> <li>• Foreign Key</li> <li>• Check</li> </ul> <p>Not Null: By default, a column can hold NULL values. If you do not want a column to</p>	3M + 2M	CO6	L2

	<p>have a NULL value, then you need to define such constraint on this column specifying that NULL is now not allowed for that column.</p> <p>Unique: The UNIQUE Constraint prevents two records from having identical values in a particular column.</p> <p>Default: The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value.</p> <p>Primary Key: A primary key is a single field or combination of fields that uniquely identify a record. The fields that are part of the primary key cannot contain a NULL value and must be Unique. Each table should have a primary key, and each table can have only ONE primary key.</p> <p>Foreign Key: A foreign key is a key used to link two tables together. Foreign Key is a column or a combination of columns whose values match a Primary Key of another table. The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.</p> <p>Example:  CREATE TABLE EMPLOYEE  (  SSN NUMBER(10) PRIMARY KEY,  NAME VARCHAR(10) NOT NULL,  PHONE NUMBER(10) UNIQUE,  AGE NUMBER(2) CHECK AGE&gt;16,  SALARY NUMBER(5) DEFAULT 10000,  DEPTNO NUMBER(2) REFERENCES DEPARTMENT(DNO)  )</p>			
--	---	--	--	--

Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1:	Describe features, classifications and characteristics of database systems.	1	2	1	1	1	1	1	-		-	-	1
CO2:	Design an information model for given requirements expressed in the form of an entity relation diagram.	1	3	3	1	1	2	-	-	1	-	-	1
CO3:	Design a relational data model for a given information model.	1	3	3	1	1	2	-	-	1	-	-	1
CO4:	Write relational algebra query for a given problem.	1	3	3	1	1	2	-	-	1	-	-	1
CO5:	Write SQL for CRUD to fulfill given requirement.	1	3	3	1	1	2	-	-	1	-	-	1
CO6:	Apply normalization techniques for a given relational model.	1	3	3	1	1	2	-	-	1	-	-	1

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 - *Engineering knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - *Conduct investigations of complex problems*; PO5 - *Modern tool usage*; PO6 - *The Engineer and society*; PO7- *Environment and sustainability*; PO8 – *Ethics*; PO9 - *Individual and team work*; PO10 - *Communication*; PO11 - *Project management and finance*; PO12 - *Life-long learning*