USN | | | | | | | | | | |

Internal Assessment Test 2 – Nov. 2017

Sub:	DATABASE MANAGEMENT SYSTEM					Sub Code:	15CS53	Branch:	CSE
Date:	08/11/2017	Duration:	90 min's	Max Marks:	50	Sem / Sec:	5 / CSE- A,B,C		

Answer any 5 full questions.

		MARKS	OBE CO	OBE RBT
1	Explain informal design guidelines for relation schemas.	[10]	CO4	L2
2	Explain with an example, the basic constraints that can be specified, when you create a table in SQL.	[10]	CO3	L2
3	Consider the following relation for published books: BOOK(BookTitle, AuthorName, BookType, ListPrice, AuthorAffiliation, Publisher) Suppose the following dependencies exist: BookTitle → BookType, Publisher BookType → ListPrice AuthorName → AuthorAffiliation Find the key in this relation and normalize it upto 3NF	[10]	CO4	L3
4	a) What is a view? Explain how to create the view and how view can be dropped? Discuss view update. b) What is a trigger? How is it defined in SQL?Explain with examples	[5] [5]	CO3	L2
5	EMP(Fname,Lname,SSN,Bdate,Address,Sex,Salary,SuperSSN,DNo) DEPT(Dname,Dnumber,MgrSSN,Mgrstartdate) DEPTLOC(Dnumber,Dloc) PROJECT(Pname,Pnumber,Ploc,Dnum) WORKS-ON(ESSN,PNO,Hours) DEPENDENT(ESSN,DepnName,Sex,Relationship) For the above schema, Write SQL Queries for the following questions: a) Retrieve number of dependents for the employee 'Ram'. b) Retrieve name and address of all employees who work for the 'CSE' department. c) List female employees who are from 'Bangalore'. d) Retrieve the names of all employees who do not have supervisors. e) Show the resulting salaries if every employee working on the 'ProductX' project is given a 10% raise.	[10]	CO3	L3
6	a) What is a Functional dependency? Explain with example. List the Inference rules for functional dependencies. b) Consider R = (A B C D E F) and Functional dependencies F = {A→BC, C→E, CD→EF} Show that AD→F Explain the following	[10]	CO2	L3
7.	a) Explain stored procedures with example. c) Explain JDBC classes and interfaces	[10]	CO3	L2

SOLUTION

1	Explain informal design guidelines for relation schemas. GUIDELINE 1: 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. GUIDELINE 2: 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. GUIDELINE 3: 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. 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. 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. 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.
2	Explain with an example, the basic constraints that can be specified, when you create a table in SQL. The basic constraints that can be specified when you create a table in SQL are: • Not Null • Unique • Default • Primary Key • Foreign Key • Check Not Null: By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such constraint on this column specifying that NULL is now not allowed for that column. Unique: The UNIQUE Constraint prevents two records from having identical values in a particular column. Default: The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value. 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. 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. Example: CREATE TABLE EMPLOYEE (SSN NUMBER(10) PRIMARY KEY, NAME VARCHAR(10) NOT NULL, PHONE NUMBER(10) UNIQUE, AGE NUMBER(2) CHECK AGE>16, SALARY NUMBER(5) DEFAULT 10000, DEPTNO NUMBER(2) REFERENCES DEPARTMENT(DNO))
3	Consider the following relation for published books: BOOK(BookTitle, AuthorName, BookType, ListPrice, AuthorAffiliation, Publisher) Suppose the following dependencies exist: BookTitle → BookType, Publisher BookType → ListPrice AuthorName → AuthorAffiliation Find the key in this relation and normalize it upto 3NF Key: (BookTitle and AuthorName)

The relation is in 1NF and not in 2NF as no attributes are fully functionally dependent on the key(BookTitle and AuthorName). It is also not in 3NF.

2NF decomposition:

• Book1(BookTitle, AuthorName)

• Book2(BookTitle, BookType, ListPrice, Publisher)

• Book3(AuthorName, AuthorAffiliation)

The relations are not in 3NF because:

• BookTitle —-> BookType—-> ListPrice

Thus, BookType is neither a key itself nor a subset of a key and ListPrice is not a prime attribute. The 3NF decomposition will eliminate the transitive dependency of Listprice.

3NF decomposition:

• Book1(BookTitle, AuthorName)
• Book2A(BookTitle, BookType, Publisher)
• Book2B(BookType, ListPrice)
• Book3(AuthorName, AuthorAffiliation)

4 a) What is a view? Explain how to create the view and how view can be dropped? Discuss view update.

A view refers to a single virtual table that is derived from other tables

Eg:

```
CREATE VIEW WORKS_ON1
AS SELECT FNAME, LNAME, PNAME, HOURS
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE SSN=ESSN AND PNO=PNUMBER

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
```

A View is always up to date; A view is realized at the time we specify(or execute) a query on the view

DROP VIEW WORKS_ON1: TO DROP A VIEW.

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.

- A view with a single defining table is updatable if we view contain PK or CK of the base table
- View on multiple tables using joins are not updatable
- View defined using grouping/aggregate are not updatable

b) What is a trigger? How is it defined in SQL?Explain with examples

Trigger is used to specify automatic actions that the database system will perform when certain events and conditions occur.

A typical trigger which is regarded as an ECA (Event, Condition, Action) rule has three components:

i)The event(s): These are usually database update operations that are explicitly applied to the database. These events are specified after the keyword BEFORE in our example, which means that the trigger should be executed before the triggering operation is executed. An alternative is to use the keyword AFTER, which specifies that the trigger should be executed after the operation specified in the event is completed.

ii) The condition that determines whether the rule action should be executed: Once the triggering event has occurred, an optional condition may be evaluated.

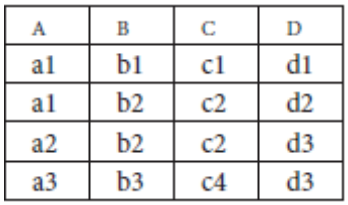
iii) The action to be taken: The action is usually a sequence of SQL statements, but it could also be a database transaction or an external program that will be automatically executed.

Eg:

```
CREATE TRIGGER SALARY_VIOLATION
BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN
   ON EMPLOYEE
FOR EACH ROW
      WHEN ( NEW.SALARY > ( SELECT SALARY FROM EMPLOYEE
WHERE SSN = NEW.SUPERVISOR_SSN ) )
```

	INFORM_SUPERVISOR(**NEW**.Supervisor_ssn, **NEW**.Ssn);

5	EMP(Fname,Lname,SSN,Bdate,Address,Sex,Salary,SuperSSN,DNo)
	DEPT(Dname,Dnumber,MgrSSN,Mgrstartdate)
	DEPTLOC(Dnumber,Dloc)
	PROJECT(Pname,Pnumber,Ploc,Dnum)
	WORKS-ON(ESSN,PNO,Hours)
	DEPENDENT(ESSN,DepnName,Sex,Relationship)
	For the above schema, Write SQL Queries for the following questions:
	a) Retrieve number of dependents for the employee 'Ram'.
	`SELECT E.SSN, COUNT(D.Dependent_name)FROM EMPLOYEE E, DEPENDENT D WHERE E.SSN=D.ESSN AND E.Fname='Ram' GROUP BY E.SSN;`
	b) Retrieve name and address of all employees who work for the 'CSE' department.
	`SELECT Fname, Lname, Address FROM EMPLOYEE E, DEPARTMENT D WHERE E.Dno = D.Dnumber AND D.Dname = 'CSE';`
	c) List female employees who are from 'Bangalore'.
	`SELECT Fname, Lname FROM EMPLOYEE WHERE Sex='F' AND Address LIKE '%Bangalore%';`
	d) Retrieve the names of all employees who do not have supervisors.
	`SELECT Fname, Lname FROM EMPLOYEE WHERE SUPER_SSN IS NULL;`
	e) Show the resulting salaries if every employee working on the 'ProductX' project is given a 10% raise.
	`SELECT E.SSN, Salary*1.1 FROM EMPLOYEE E, WORKS_ON W, PROJECT P WHERE E.SSN=W.ESSN AND W.Pno=P.Pnumber AND P.Pname = 'ProductX';`

6	a)What is a Functional dependency? Explain with example. List the Inference rules for functional dependencies. A functional dependency, denoted by X → Y, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t1 and t2 in r that have t1[X] = t2[X], they must also have t1[Y] = t2[Y].  In the above table, the following FDs may hold because the four tuples in the current extension have no violation of these constraints: B → C; C → B; {A, B} → C; {A, B} → D; and {C, D} → B. However, the following do not hold because we already have violations of them in the given extension: A → B (tuples 1 and 2 violate this constraint); B → A (tuples 2 and 3 violate this constraint); D → C (tuples 3 and 4 violate it). Inference Rules: IR1 (reflexive rule)[1]: If $X \supseteq Y$, then $X \rightarrow Y$. IR2 (augmentation rule)[2]: $\{X \rightarrow Y\} \models XZ \rightarrow YZ$. IR3 (transitive rule): $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$. IR4 (decomposition, or projective, rule): $\{X \rightarrow YZ\} \models X \rightarrow Y$. IR5 (union, or additive, rule): $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$. IR6 (pseudotransitive rule): $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$. b)Consider R = (A B C D E F) and Functional dependencies F = {A→BC, C→E, CD→EF} Show that AD→F Explain the following

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

Sol^{n} :

Given $A \rightarrow BC \rightarrow ①$
$B \rightarrow E \rightarrow ②$
$CD \rightarrow EF \rightarrow ③$

Applying Decomposition rule on ①, we get

$A \rightarrow B$
$A \rightarrow C \rightarrow ④$

Applying pseudo transitive rule on ③ using ①, we get

$AD \rightarrow EF \rightarrow ⑤$

Applying Decomposition rule on ⑤, we get

$AD \rightarrow E$

$AD \rightarrow F$

7. a) Explain stored procedures with example.

Database stored procedures and SQL/PSM

- Persistent procedures/functions (modules) are stored locally and executed by the database server. As opposed to execution by clients.
- Advantages: If the procedure is needed by many applications, it can be invoked by
- any of them (thus reduce duplications)
- Execution by the server reduces communication costs
- Enhance the modeling power of views
- Disadvantages: Every DBMS has its own syntax and this can make the system less portable

A stored procedure EXAMPLE

CREATE PROCEDURE procedure-name (params)
local-declarations
procedure-body;
A stored function

CREATE FUNCTION fun-name (params)
RETRUNS return-type
local-declarations function-body;

Calling a procedure or function: CALL procedure-name/fun-name (arguments);

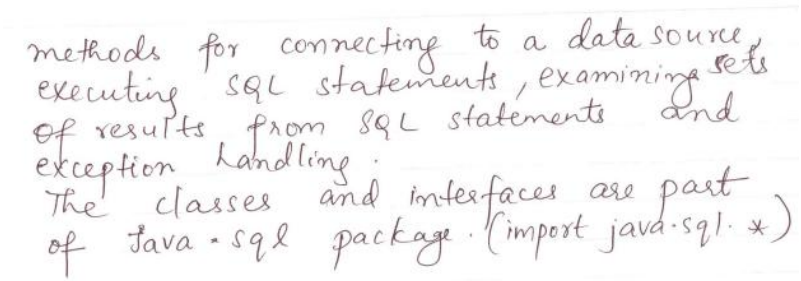
E.g.,

```
CREATE FUNCTION DEPT_SIZE (IN deptno INTEGER)
RETURNS VARCHAR[7]
DECLARE TOT_EMPS INTEGER;
SELECT COUNT (*) INTO TOT_EMPS
FROM SELECT EMPLOYEE
WHERE DNO = deptno;
IF TOT_EMPS > 100
  THEN RETURN —HUGE
ELSEIF TOT_EMPS > 50
  THEN RETURN —LARGE
ELSEIF TOT_EMPS > 30
  THEN RETURN —MEDIUM
ELSE RETURN SMALL
ENDIF
```

b) Explain JDBC classes and interfaces

JDBC classes and Interfaces

JDBC is a collection of Java classes and interfaces that enables database access from programs in Java. It contains



JDBC Driver Management

- All drivers are managed by the DriverManager class
- Loading a JDBC driver:
  - In the Java code:
    Class.forName("oracle/jdbc.driver.Oracledriver");
  - When starting the Java application:
    -Djdbc.drivers=oracle/jdbc.driver

Connections in JDBC

We interact with a data source through sessions. Each connection identifies a logical session.

- JDBC URL:
  jdbc:<subprotocol>:<otherParameters>

Example:

```
String url= "jdbc:oracle:www.bookstore.com:3083" ;
Connection con;
try{
        con = DriverManager.getConnection(url,usedId,password);
} catch SQLException excpt { ⋯}
```

Executing SQL Statements

- Three different ways of executing SQL statements:
  - Statement (both static and dynamic SQL statements)
  - PreparedStatement (semi-static SQL statements)
  - CallableStatment (stored procedures)
- PreparedStatement class:
  Precompiled, parametrized SQL statements:
  - Structure is fixed
  - Values of parameters are determined at run-time

Resultsets

- PreparedStatement.executeUpdate only returns the number of affected records
- PreparedStatement.executeQuery returns data, encapsulated in a ResultSet object (a cursor)

```
ResultSet rs=pstmt.executeQuery(sql);
// rs is now a cursor
  While (rs.next()) {
      // process the data
      }
```