

TED (10)-4070

Reg. No. ....

(REVISION—2010)

Signature.....

FOURTH SEMESTER DIPLOMA EXAMINATION IN ENGINEERING/  
TECHNOLOGY—OCT,2013

**DATABASE MANAGEMENT SYSTEMS**

(Common for CT and IF)

[Time : 3 hours

(Maximum marks : 100)

**PART—A**

I. Answer the following questions in one or two sentences. Each question carries 2 marks.

1. Given an example for a database schema.

**Example of database schema**

STUDENT

|      |               |       |       |
|------|---------------|-------|-------|
| Name | StudentNumber | Class | Major |
|------|---------------|-------|-------|

COURSE

|            |              |             |            |
|------------|--------------|-------------|------------|
| CourseName | CourseNumber | CreditHours | Department |
|------------|--------------|-------------|------------|

PREREQUISITE

|              |                    |
|--------------|--------------------|
| CourseNumber | PrerequisiteNumber |
|--------------|--------------------|

2. List the commands under the category of DDL

DDL (data definition language)

- ❖ Create
- ❖ Alter
- ❖ Drop
- ❖ Truncate
- ❖ Comment

❖ Rename

### 3. Define candidate key.

If a relation has *several* candidate keys, one is chosen arbitrarily to be the primary key. The primary key attributes are *underlined*.

### 4. Mention the use of referential integrity constraints.

A constraint involving *two* relations (the previous constraints involve a *single* relation).

Used to specify a *relationship* among tuples in two relations: the referencing relation and the referenced relation.

Tuples in the *referencing relation*  $R_1$  have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the *referenced relation*  $R_2$ . A tuple  $t_1$  in  $R_1$  is said to reference a tuple  $t_2$  in  $R_2$  if  $t_1[\text{FK}] = t_2[\text{PK}]$ .

A referential integrity constraint can be displayed in a relational database schema as a directed arc from  $R_1.\text{FK}$  to  $R_2$ .

### 5. Suggest a method for connecting a database server through application software.

- Connecting to a database, as already mentioned is a long process of Opening the connection, Closing the connection and so on. To repeat this process for every single user in the application is not a good approach and will slow down the processes of code execution. So, in program executions many such connections would be opened and closed and again opened that are identical. These processes are time consuming and not are opposite of good UX.
- In .NET Framework ADO.NET plays a part in this and minimizes the opening and closing process to make the program execution a bit faster by creating, what we call, *Connection Pool*. This technique reduces the number of times the connection is opened, by saving the instance of the connection, for every new connection it just looks for already opened connection and then if the connection exists, doesn't attempt to create new connection otherwise opens a new connection based on the connection String.
- It must be remembered, that only the connections with same configuration can be pooled. Any connection with even single dissimilarity would require a new pool for itself. Generally, it is based on the ConnectionString of the connection. You can learn how would that differ by changing the values in the connection string.

## PART B

II. Answer *any five* of the following. Each question carries 6 marks.

1. **Distinguish between physical and logical data independence.**

**Physical Data Independence** – the ability to modify the physical schema without changing the application programs

- Applications depend on the logical schema
- DBA may change physical level (tuning) without affecting applications
- The DBMS automatically make the required adjustments, and application programs are not changed (queries may need to be recompiled and optimized...)

**Logical Data Independence** – the ability to modify the logical schema without changing the application programs

- Applications depend on the logical schema via the Views
- Can be supported on a limited basis only (if view is not affected)

2. **Explain centralized and distributed database architecture.**

- **A distributed database** is a database in which portions of the database are stored on multiple computers within a network. Users have access to the portion of the database at their location so that they can access the data relevant to their tasks without interfering with the work of others. A centralized distributed database management system (DDBMS) manages the database as if it were all stored on the same computer. The DDBMS synchronizes all the data periodically and, in cases where multiple users must access the same data, ensures that updates and deletes performed on the data at one location will be automatically reflected in the data stored elsewhere.

**Centralized Systems:**

- Run on a single computer system and do not interact with other computer systems.

- General-purpose computer system: one to a few CPUs and a number of device controllers that are connected through a common bus that provide access to shared memory.
- Single-user system (e.g., personal computer or workstation): desk-top unit, single user, usually has only one CPU and one or two hard disks; the OS may support only one user.
- Multi-user system: more disks, more memory, multiple CPUs, and a multi-user OS. Serve a large number of users who are connected to the system via terminals. Often called *server* systems.

**3. Distinguish between Unique Key and Primary key with example.**

| Primary key   | Unique Key  |
|---|---|
| None of the fields that are part of the primary key can contain a null value. | Some of the fields that are part of the unique constraint can contain null values as long as the combination of values is unique. |

**4. Explain selection and rename operation with example.**

**Select:** Command description: select command is used to display the table & table values.

Sql> select \* from Table Name;

**Rename:** rename command is used to rename the objects

Sql>alter table old table name rename to new name;

**5. List the advantages of concurrent execution of transactions.**

DBMS ensures that execution of  $\{T_1, \dots, T_n\}$  is equivalent to some serial execution  $T_1' \dots T_n'$ .

Before reading/writing an object, a transaction requests a lock on the object, and waits till the DBMS gives it the lock. All locks are released at the end of the transaction. (Strict 2PL locking protocol.)

Idea: If an action of  $T_i$  (say, writing X) affects  $T_j$  (which perhaps reads X), one of them, say  $T_i$ , will obtain the lock on X first and  $T_j$  is forced to wait until  $T_i$  completes; this effectively orders the transactions.

What if  $T_j$  already has a lock on Y and  $T_i$  later requests a lock on Y?  
(Deadlock!)  $T_i$  or  $T_j$  is aborted and restarted!

**6. A student table contain the fields register\_number, mark 1, mark 2, mark 3. Write suitable SQL statements for the following:**

**(i) To get the account of total number of students in the table.**

SQL>select \* from student count(student);

**(ii) To delete all records in the table permanently.**

SQL>Truncate table student;

**(iii) List the register number of passed students (assume pass mark is 40 in each subject).**

SQL>select reg.no from student where SUM(mark1,mark2,mark3)>=120;

**7. Explain views and list their advantages.**

A **view** is a logical representation of one or more tables. In , a view is a stored query. A view derives its data from the tables on which it is based,called base tables. Base tables can be tables or other views. All operations performed on a view actually affect the base tables. You can use views in most places where tables are used. You can query, insert, update and delete from views. Views can be handled as any other table but they do not occupy any space.

Views enable you to tailor the presentation of data to different types of users. Views are often used to:

- Provide an additional level of table security by restricting access to predetermined set of rows or columns of a table.
- Hide data complexity
- Present the data in a different perspective from that of the base table
- Isolate applications from changes in definitions of base tables.

**PART—C**

(Answer one full question from each unit. Each question carries 15 marks.)

**III. (a). define schema, sub schema a explain three schema architecture with diagram.**

Schema – the logical structure of the database

e.g., the database consists of information about a set of customers and accounts and the relationship between them)

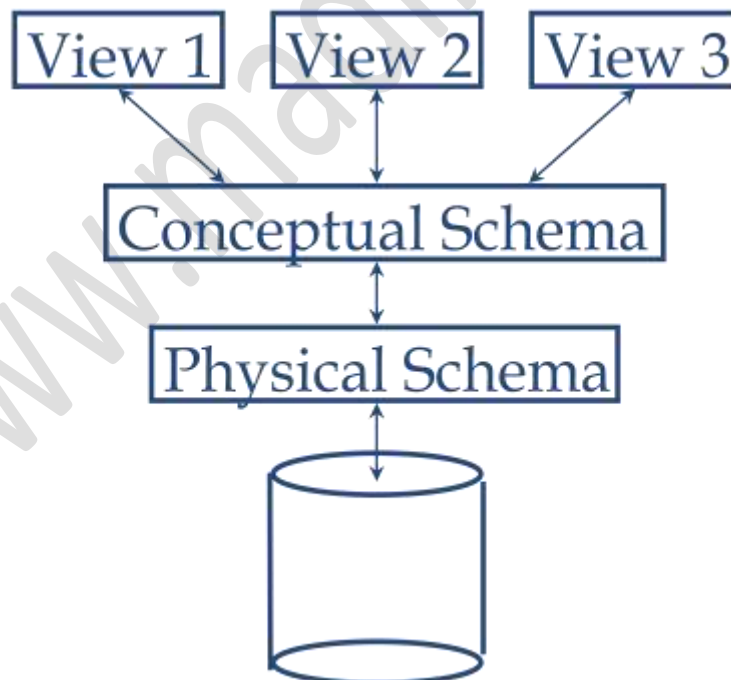
Analogous to type information of a variable in a program

Physical schema: database design at the physical level

Logical schema: database design at the logical level

Instance – the actual content of the database at a particular point in time

Analogous to the value of a variable



- Physical (Storage) schema decisions
- Mapping of entities to files (OS files)

- Data representation and encoding (compression)
- Access methods (Direct, Hashing, Indexed)
- Which indexes to maintain
- Clustering of records
- OS/DBMS issues (buffer management)

#### **External (View) schema decisions**

- Which entities to present/filter
- Data representation and encoding (compression)
- Programming language dependent issues
- Changes to names, order of attributes
- Derived (computed) fields and joined tables

#### **(b) List various database users.**

- Database administrators: responsible for authorizing access to the database, for co-ordinating and monitoring its use, acquiring software, and hardware resources, controlling its use and monitoring efficiency of operations.
- Database Designers: responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

End-users: they use the data for queries, reports and some of them actually update the database content.

- **Casual** : access database occasionally when needed

**Naïve or Parametric** : they make up a large section of the end-user population. They use previously well-defined functions in the form of “canned transactions” against the database.

Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations

- **Sophisticated** : these include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities. Many use tools in the form of software packages that work closely with the stored database.
- **Stand-alone** : mostly maintain personal databases using ready-to-use packaged applications. An example is a tax program user that creates his or her own internal database.

**OR**

**IV.**

**a) Explain various database languages and give 2 examples for each.**

**DML:** DML is abbreviation of **Data Manipulation Language**. It is used to retrieve, store, modify, delete, insert and update data in database.

|        |   |           |          |      |      |       |       |
|--------|---|-----------|----------|------|------|-------|-------|
| SELECT | – | Retrieves | data     | from | a    | table |       |
| INSERT | – | Inserts   | data     | into | a    | table |       |
| UPDATE | – | Updates   | existing | data | into | a     | table |

DELETE – Deletes all records from a table

**DDL**

DDL is abbreviation of **Data Definition Language**. It is used to create and modify the structure of database objects in database.

|        |   |         |         |    |     |          |
|--------|---|---------|---------|----|-----|----------|
| CREATE | – | Creates | objects | in | the | database |
| ALTER  | – | Alters  | objects | of | the | database |
| DROP   | – | Deletes | objects | of | the | database |

TRUNCATE – Deletes all records from a table and resets table identity to initial value.

**DCL**



DCL is abbreviation of **Data Control Language**. It is used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it.

GRANT – Gives user's access privileges to database  
REVOKE – Withdraws user's access privileges to database given with the GRANT command

**b) Write notes on data models. Explain physical data model & steps for its design.**

**A collection of modeling tools for describing**

- data
- data relationships
- data semantics
- data constraints
- Entity-Relationship model
- Relational model
- Other models:
  - object-oriented model
  - semi-structured data models (XML)
  - Older models: network model and hierarchical model
- **Low level or physical data models** provide concepts that describe the
- details of how data is stored in the computer. Concepts provided
- by low-level data models are generally meant for computer
- specialists, not for typical end users.

## UNIT—II

V.

**a) Distinguish strong entity set & weak entity set with example.**

|                 |                   |
|-----------------|-------------------|
| Weak Entity set | Strong Entity set |
|-----------------|-------------------|

|  |  |
|--|--|
| <ul style="list-style-type: none"> <li>■ Some entity sets in real world naturally depend on some other entity set <ul style="list-style-type: none"> <li>■ They can be uniquely identified only if combined with another entity set</li> </ul> </li> <li>■ Example: <ul style="list-style-type: none"> <li>■ section1, section2, ... become unique only if you put them into a context, e.g. csce4350</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>■ Some entity sets in real world naturally independent from other entity set <ul style="list-style-type: none"> <li>■ They can be uniquely identified</li> </ul> </li> <li>■ Example: <ul style="list-style-type: none"> <li>■ Vehicles, student.etc</li> </ul> </li> </ul> |
|--|--|

b) Consider two relations supplier & products :

**Supplier (s#, sname, status, city)**

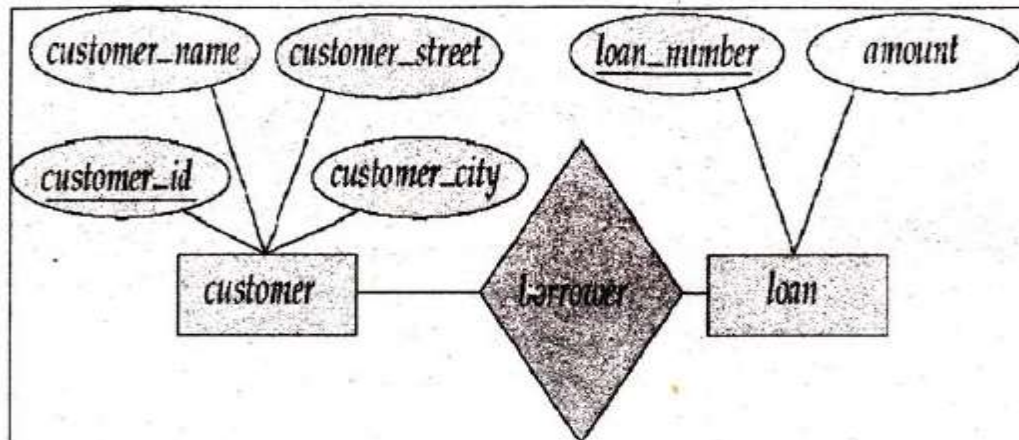
**Products (p#, pname, color, weight, city).**

**Suppose if we want to get all the combinations of supplier and product information such that the supplier and product are located in the same city, identify the suitable relational operations involved and also give the syntax.**

- ▶ **Boyce-Codd Normal Form.**
- ▶ Supplier (s#, sname, status, city)
- ▶ Products (p#, pname, color, weight, city).
- ▶ A relation schema  $R$  is in BCNF with respect to a set  $F$  of functional dependencies if for all functional dependencies in  $F^+$  of the form
- ▶  $\alpha \rightarrow \beta$ , where  $\alpha \subseteq R$  and  $\beta \subseteq R$ , at least one of the following holds:
- ▶  $\alpha \rightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$ )
- ▶  $\alpha$  is a superkey for  $R$

OR

VI. (a) Distinguish specialization figure identify entity set, relationships, attributes and primary key if any in the relations:



○ **Entity set:**

Customer, loan (strong entities).

○ **Relationships:**

One to one relation (1:1)

○ **Attributes:**

Customer\_name , Customer\_street, Customer\_id, Customer\_city, loan\_number, amount.

■ **Primary key:**

- Customer\_id
- Loan\_number

UNIT—III

VII. (a) We have two relations :

Item (itemno, description, price, stock)

Orders (orderno, date\_of\_order, itemno, qty)

Identify integrity constraint required to prevent the ordering of an item that have not placed in the relation 'item'. Give the syntax to implement the constraint.

The Orders relation schema:

Item(itemno, description,price,stock)

Orders(orderno,date\_of\_order,itemno,qty)

In Item relation:has 1 key Key1 = {itemno}, Key2 = {description}, which are also superkeys. {itemno,price} is a superkey but *not* a key.

In Orders relation:has 1 key Key1 = {orderno}, Key2 = {date\_of\_order}, which are also superkeys. {orderno,date\_of\_order} is a superkey but *not* a key.

- If a relation has *several* candidate keys, one is chosen arbitrarily to be the primary key. The primary key attributes are *underlined*.

**(b) Define normalization. Explain 1 NF, 2 NF with example.**

- This is the process which allows you to winnow out redundant data within your database.
- This involves restructuring the tables to successively meeting higher forms of Normalization.
- A properly normalized database should have the following characteristics
  - Scalar values in each fields
  - Absence of redundancy.
  - Minimal use of null values.
  - Minimal loss of information.
- **First Normal Form (1NF)**

A table is considered to be in 1NF if all the fields contain only scalar values (as opposed to list of values).

Place all items that appear in the repeating group in a new table

Designate a primary key for each new table produced.

Duplicate in the new table the primary key of the table from which the repeating group was extracted or vice versa.

1. Place all items that appear in the repeating group in a new table

2. Designate a primary key for each new table produced.
  3. Duplicate in the new table the primary key of the table from which the repeating group was extracted or vice versa.
- Second Normal Form (2NF)

For a table to be in 2NF, there are two requirements

- The database is in first normal form
- All **nonkey** attributes in the table must be functionally dependent on the entire primary key

*Note: Remember that we are dealing with non-key attributes*

**OR**

#### **VIII. (a) Define functional dependency.**

- **Functional Dependencies**

If one set of attributes in a table determines another set of attributes in the table, then the second set of attributes is said to be functionally dependent on the first set of attributes.

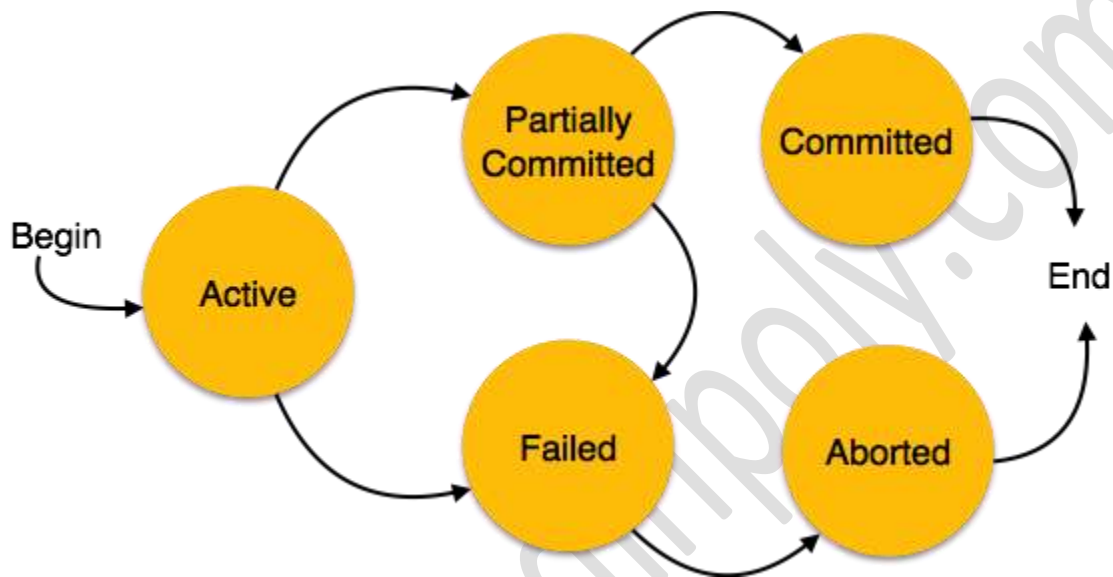
Database to track reviews of papers submitted to an academic conference. Prospective authors submit papers for review and possible acceptance in the published conference proceedings. Details of the entities

- Author information includes a unique author number, a name, a mailing address, and a unique (optional) email address.
- Paper information includes the primary author, the paper number, the title, the abstract, and review status (pending, accepted, rejected)
- Reviewer information includes the reviewer number, the name, the mailing address, and a unique (optional) email address

- A completed review includes the reviewer number, the date, the paper number, comments to the authors, comments to the program chairperson, and ratings (overall, originality, correctness, style, clarity)

**(b) With help of a diagram illustrate transaction states.**

A transaction in a database can be in one of the following state:



- **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 this state. After execution of all operations, the database system performs some checks e.g. the consistency state of database after applying output of transaction onto the database.
- **Failed:** If any checks made by database recovery system fails, the transaction is said to be in failed state, from where it can no longer proceed further.
- **Aborted:** If any of checks fails and transaction reached in Failed state, the recovery manager rolls back all its write operation on the database to make database in the state where it was prior to start of execution of transaction. Transactions in this state are called aborted. Database recovery module can select one of the two operations after a transaction aborts:
  - Re-start the transaction
  - Kill the transaction

- **Committed:** If transaction executes all its operations successfully it is said to be committed. All its effects are now permanently made on database system.

#### UNIT—IV

**IX. (a). List the operations that can be done using ‘ALTER TABLE’ command.**

**Give the syntax of each operation.**

■ **Dropping primary**

alter table table name drop primary key;

■ **Creating primary key**

Sql> alter table Table name add primary key (attribute);

■ **Creating foreign key**

Sql> alter table Table Name2 add foreign key (attribute) references Table name1 (attribute);

■ **Adding & modifying constraints**

Sql> alter table Table name add ( constraint);

**(b) Distinguish Embedded SQL & Dynamic SQL**

| <b>Embedded SQL</b>   | <b>Dynamic SQL</b>  |
|---|---|
| <p>The central idea of embedded SQL is to blend SQL language statements directly into a program written in a "host" programming language, such as C, Pascal, COBOL, FORTRAN, PL/I, or Assembler. Embedded SQL uses the following techniques to embed the SQL statements:</p> <ul style="list-style-type: none"> <li>• SQL statements are intermixed with</li> </ul> | <p>The central concept of dynamic SQL is simple: don't hardcode an embedded SQL statement into the program's source code. Instead, let the program build the text of a SQL statement in one of its data areas at runtime. The program then passes the statement text to the DBMS for execution "on the fly." Although the details get</p> |

|   |  |
|---|--|
| <p>statements of the host language in the source program. This "embedded SQL source program" is submitted to a SQL pre-compiler, which processes the SQL statements.</p> <ul style="list-style-type: none"> <li>• Variables of the host programming language can be referenced in the embedded SQL statements, allowing values calculated by the program to be used by the SQL statements.</li> <li>• Program language variables also are used by the embedded SQL statements to receive the results of SQL queries, allowing the program to use and process the retrieved values.</li> </ul> | <p>quite complex, all of dynamic SQL is built on this simple concept, and it's a good idea to keep it in mind. As you might expect, dynamic SQL is less efficient than static SQL. For this reason, static SQL is used whenever possible, and many application programmers never need to learn about dynamic SQL. However, dynamic SQL has grown in importance as more and more database access has moved to a client/server, front-end/back-end architecture over the last ten years.</p> |
|---|--|

**OR**

**X.**

**(a) Consider the following set of commands:**

**SAVEPOINT S1 ;**

**DELETE FROM STUDENT WHERE NAME="ABRAHAM";**

**SAVEPOINT S2;**

**COMMIT;**

**Explain each command and give the use of the above code segment.**

**SAVE POINT**

It define a new save point within the current transaction

: save point command is used to identify a point in a



Transaction in which it can be restored using roll back command.

: rollback command is used to restore database to original

Since last commit.

The Name" ABRAHAM" was deleted from the Table STUDENT,

Here S1 and S2 are savepoint. So it cannot be deleted.

And the transaction was committed.

**(b). Explain cursor and their use. List steps for handling them.**

Cursors enable us to examine, in the host language program, a collection of JWS computed by an Embedded SQL statement:

1) We usually need to open a cursor if the embedded statement is a SELECT (i.e.) a query).

However, we can avoid opening a cursor if the answer contains a single row, as we see shortly.

2) NSERT, DELETE, and UPDATE statements typically require no cursor, although some variants of DELETE and UPDATE use a cursor.

A cursor can be thought of as 'pointing' to a row in the collection of answers to the query associated with it. When a cursor is opened, it is positioned just before the first row. We can use the FETCH command to read the first row of cursor sinfo into host language variables:

FETCH sinfo INTO: csname, cage; When the FETCH statement is executed, the cursor is

positioned to point at the next row (which is the first row in the tablewhen FETCH is executed for the first time after opening the cursor) and the column values in the row are copied into the

corresponding host variables. By repeatedly executing this FETCH statement (say,

in a while-loop in the C program), we can read all the rows computed by the query, one row at a time.