**TED (10)-4070** **Reg. No. ……………………**

**(REVISION—2010)** **Signature…………………...**

FOURTH SEMESTER DIPLOMA EXAMINATION IN ENGINEERING/ TECHNOLOGY—MARCH, 2012

**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. **Name any two DBMS packages.**

   4th Dimension

   Adabas D

2. **Define a tuple in RDBMS.**

   A tuple is one record (one row).

3. **Define a weak entity set.**

   A **weak entity** is an entity that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key. The foreign key is typically a primary key of an entity it is related to.

4. **List the use of 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 in a different table

5. **Write an example of select query with ORDERBY clause.**

```
SELECT column-list
FROM table_name
[WHERE condition]
[ORDER BY column1, column2, .. columnN] [ASC | DESC];
```

## PART B

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

**1. Explain DBMS. List any two advantages of DBMS over file processing**

DBMS A database management system is the software system that allows users to define, create and maintain a database and provides controlled access to the data. A Database Management System (DBMS) is basically a collection of programs that enables users to store, modify, and extract information from a database as per the requirements. DBMS is an intermediate layer between programs and the data. Programs access the DBMS, which then accesses the data

**Controlling Redundancy:** In file system, each application has its own private files, which cannot be shared between multiple applications. 1:his can often lead to considerable redundancy in the stored data, which results in wastage of storage space. By having centralized database most of this can be avoided. It is not possible that all redundancy should be eliminated.

**Integrity can be enforced:** Integrity of data means that data in database is always accurate, such that incorrect information cannot be stored in database. In order to maintain the integrity of data, some integrity constraints are enforced on the database.

**2. List different database users. Explain the duties of database administrators.**

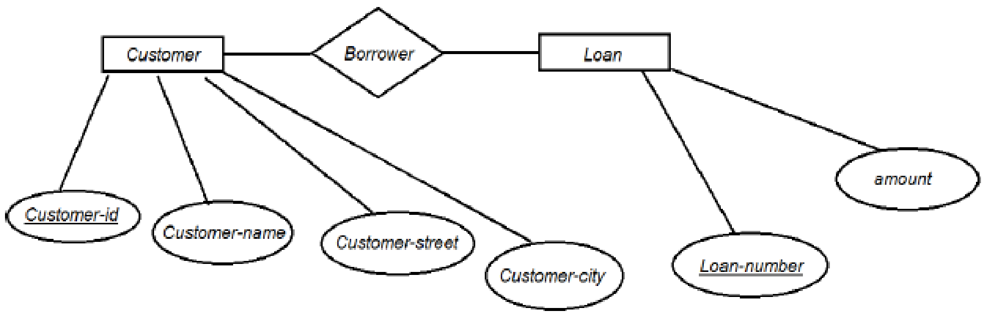
- Naive Users
- Online Users
- Application Programmers
- Sophisticated Users

- Data Base Administrator (DBA)

**Database Administrator:** The database administrator (DBA) is the person or group in charge for implementing the database system ,within an organization. The "DBA has all the system privileges allowed by the DBMS and can assign (grant) and remove (revoke) levels of access (privileges) to and from other users. DBA is also responsible for the evaluation, selection and implementation of DBMS package.

**3. Explain ER diagram. Show an example with entities customer and loan.**

- ER diagram is widely used in database design
- Represent conceptual level of a database system
- Describe things and their relationships in high level



**4. Define transaction and what are different transaction states?**

- A logical unit of work on a database
- An entire program
- A portion of a program
- A single command
- The entire series of steps necessary to accomplish a logical unit of work

- Successful transactions change the database from one CONSISTENT STATE to another (One where all data integrity constraints are satisfied)

**Different transaction states**

- Active Committed
- Partially committed
- Failed
- Aborted.

5. **Explain index file in SQL. Construct SQL command for creating an index on a table.**

   **Index** is an object which can be defined as the ordered list of values of a column or combination of columns used for fastersearching and sorting of data. An index speeds up joins and searches by providing a way for a database management system to go directly to a row rather than having to search through all the rows

```
CREATE INDEX index_name
ON table_name (column_name1 [ASC | DESC] [, column_name2[ASC |
```

**ON table_name (column_name1 [ASC | DESC] [, column_name2[ASC | DESC]]......)**

6. **Explain embedded SQL.**

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 embe the SQL statements:

• SQL statements are intermixed with 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.

• 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.

Embedded SQL define how SQL statements can be embedded within general purpose programming languages, such as C, C++, Java, PL/I, Cobol, Pascal, and Fortran.
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

7. **Explain domain constraints in DBMS. Give an example with check.**

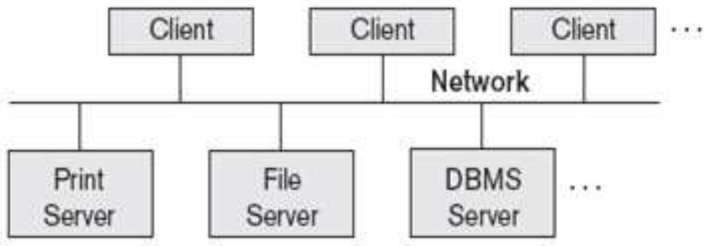
```
CREATE TABLE student
(
Roll no char (4) UNIQUE,
Name varchar2 (10) NOT NULL,
date_of_birth date NOT NULL,
marks number(3)CHECK (marks >= 0 AND marks <= 100)
);
```

**PART—C**

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

**UNIT-- I**

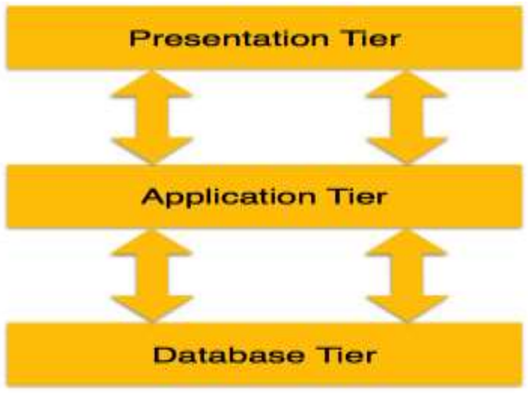
**III. (a). Explain two tier and three tier architecture.**



- Server handles
    - Query and transaction functionality related to SQL processing
- Client handles
    - User interface programs and application programs

**3-tier architecture**

Most widely used architecture is 3-tier architecture. 3-tier architecture separates it tier from each other on basis of users. It is described as follows:



- **Database (Data) Tier:** At this tier, only database resides. Database along with its query processing languages sits in layer-3 of 3-tier architecture. It also contains all relations and their constraints.

- **Application (Middle) Tier:** At this tier the application server and program, which access database, resides. For a user this application tier works as abstracted view of database. Users are unaware of any existence of database beyond application. For database-tier, application tier is the user of it. Database tier is not aware of any other user beyond application tier. This tier works as mediator between the two.

- **User (Presentation) Tier:** An end user sits on this tier. From a users aspect this tier is everything. He/she doesn't know about any existence or form of database beyond this layer. At this layer multiple views of database can be provided by the application. All views are generated by applications, which resides in application tier.

- The two-tier application programming model was developed to enhance the file server application programming model. As compared with the file server application programming model, the two-tier application programming model provides you with improved usability, scalability, and flexibility of applications.

**(b). Explain DDL, DML and DCL with two 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

**OR**

**IV. (a) Explain distributed database. Explain replication and duplication.**

A DDBMS (distributed database management system) is a centralized application that manages a distributed 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.

1. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be complex and time-consuming depending on the size and number of the distributed databases. This process can also require a lot of time and computer resources.
2. Duplication, on the other hand, has less complexity. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, users may change only the master database. This ensures that local data will not be overwritten.

**(b). Discuss 3 phases of query processor.**

1. Parsing and translation
2. Optimization
3. Evaluation

**Optimization**

- A relational algebra expression may have many equivalent expressions

E.g., *balance* 2500(*balance*(*account*)) is equivalent to balance(balance2500(*account*))

- Each relational algebra operation can be evaluated using one of several different algorithms Correspondingly, a relational-algebra expression can be evaluated in many ways.
- Annotated expression specifying detailed evaluation strategy is called an **evaluation-plan**.

E.g., can use an index on *balance* to find accounts with balance < 2500,or can perform complete relation scan and discard accounts with balance ≥2500

**Query Optimization**: Amongst all equivalent evaluation plans choose the one with lowest cost. Cost is estimated using statistical information from the database catalog

e.g. number of tuples in each relation, size of tuples, etc.

How to measure query costs:

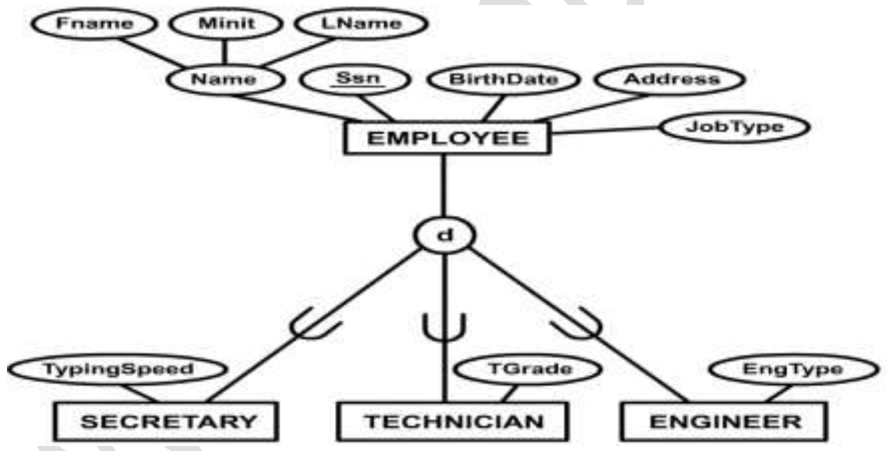
Algorithms for evaluating relational algebra operations How to combine algorithms for individual operations in order to evaluate a complete expression

We study how to optimize queries, that is, how to find an evaluation plan with lowest estimated cost

## UNIT II

**V. (a) Explain enhanced ER diagram. Draw an example ER diagram of an employee showing sub class and super class.**

- Includes all modeling concepts of basic ER
- Additional concepts: subclasses/superclasses, specialization/generalization, categories, attribute inheritance
- The resulting model is called the enhanced-ER or Extended ER (E2R or EER) model
- It is used to model applications more completely and accurately if needed
- It includes some object-oriented concepts, such as inheritance



**(b). Consider tables Car and Boat which list models of cars and boats and their respective prices. Suppose a customer wants to buy a car and a boat, but she does not want to spent more money for the boat than for the car:**

**Car**

| **Car Model** | **Car price** |
|---|---|
| **Car A** | **20,000** |
| **Car B** | **30,000** |
| **Car C** | **50,000** |

**Boat**

| **Boat Model** | **Boat price** |
|---|---|
| **Boat 1** | **10,000** |
| **Boat 2** | **40,000** |
| **Boat 3** | **60,000** |

**Discuss the relational algebra to join the above table to form the tuples which meet her requirement. What is equijoin ?**

**THETA Join**

- RESULT=Car $_{\{CarPrice>BoatPrice\}}$ Boat;
- Result=R1 $_{\{Condition\}}$ R2; Condition: $\{<, >, =, \leq, \geq, \neq\}$;
- EquiJoin when “=“.
- SELECT * FROM Car, Boat

  WHERE CarPrice>BoatPrice;

A common case of the join operation R1.R2 is one in which the join condition consists only of equality condition This type of join where the only comparison operator used is = is known as Equijoin

Ex: Two relations BOOK and PUBLISHER can joined as BOOK.P_ID and PUBLISHER.P_ID

**OR**

**VI. (a) Explain natural join using relational algebra. What will be the natural join result for the following relations :**

**Employee**

| **Name** | **Empid** | **Dept. Name** |
|---|---|---|
| **Harry** | **3415** | **Finance** |
| **Sally** | **2241** | **Sales** |
| **George** | **3401** | **Finance** |
| **Harriet** | **2202** | **Sales** |

**Department**

| **Department Name** | **Manager** |
|---|---|
| **Finance** | **George** |
| **Sales** | **Harriet** |
| **Production** | **Charles** |

**Natural join (⋈):**

Natural join (⋈) is a binary operator that is written as (*R* ⋈ *S* where *R* and *S* are relations. The result of the natural join is the set of all combinations of tuples in *R* and *S* that are equal on their common attribute names. For an example consider the tables *Employee* and *Dept* and their natural join:

***Employee***

| **Name** | **EmpId** | **DeptName** |
|---|---|---|
| Harry | 3415 | Finance |
| Sally | 2241 | Sales |
| George | 3401 | Finance |
| Harriet | 2202 | Sales |

***Dept***

| **DeptName** | **Manager** |
|---|---|
| Finance | George |
| Sales | Harriet |
| Production | Charles |

***Employee* ⋈ *Dept***

| **Name** | **EmpId** | **DeptName** | **Manager** |
|---|---|---|---|
| Harry | 3415 | Finance | George |
| Sally | 2241 | Sales | Harriet |
| George | 3401 | Finance | George |
| Harriet | 2202 | Sales | Harriet |

This can also be used to define composition of relations. For example, the composition of *Employee* and *Dept* is their join as shown above, projected on all but the common attribute *DeptName*. Incategory theory, the join is precisely the fiber product.

The natural join is arguably one of the most important operators since it is the relational counterpart of logical AND. Note carefully that if the same variable appears in each of two predicates that are connected by AND, then that variable stands for the same thing and both appearances must always be substituted by the same value. In particular, natural join allows the combination of relations that are associated by a foreign key. For example, in the above example a foreign key probably holds from *Employee*.*DeptName* to *Dept*.*DeptName* and then the natural join of *Employee* and *Dept* combines all employees with their departments. Note that this works because the foreign key holds between attributes with the same name. If this is not the case such as in the foreign key from *Dept*.*manager* to*Employee*.*Name* then we have to rename these columns before we take the natural join. Such a join is sometimes also referred to as an **equijoin** (see $\theta$-join).

More formally the semantics of the natural join are defined as follows:

$$R \bowtie S = \{ t \cup s \mid t \in R \ \wedge \ s \in S \ \wedge \ Fun(t \cup s) \}$$

where *Fun* is a predicate that is true for a relation *r* if and only if *r* is a function. It is usually required that *R* and *S* must have at least one common attribute, but if this constraint is omitted, and *R* and *S*have no common attributes, then the natural join becomes exactly the Cartesian product.

The natural join can be simulated with Codd's primitives as follows. Assume that $c_1,...,c_m$ are the attribute names common to *R* and *S*, $r_1,...,r_n$ are the attribute names unique to *R* and $s_1,...,s_k$ are the attribute unique to *S*. Furthermore assume that the attribute names $x_1,...,x_m$ are neither in *R* nor in *S*. In a first step we can now rename the common attribute names in *S*:

$$T = \rho_{x_1/c_1,\ldots,x_m/c_m}(S) = \rho_{x_1/c_1}(\rho_{x_2/c_2}(\ldots\rho_{x_m/c_m}(S)\ldots))$$

Then we take the Cartesian product and select the tuples that are to be joined:

$$P = \sigma_{c1=x1,\ldots,c_m=x_m}(R \times T) = \sigma_{c1=x1}(\sigma_{c2=x2}(\ldots\sigma_{c_m=x_m}(R \times T)\ldots))$$

Finally we take a projection to get rid of the renamed attributes:

$$U = \pi_{r_1,\ldots,r_n,c_1,\ldots,c_m,s_1,\ldots,s_k}(P)$$

**(b). Discuss the relational model concepts of database. List equivalent RDBMS notation for table, column and row.**

- A **relational database** is a digital database whose organization is based on the relational model of data, as proposed by E.F. Codd in 1970. This model organizes data into one or more tables (or "relations") of rows and columns, with a unique key for each row. Generally, each entity type described in a database has its own table, the rows representing instances of that entity and the columns representing the attribute values describing each instance. Because each row in a table has its own unique key, rows in other tables that are related to it can be linked to it by storing the original row's unique key as an attribute of the secondary row (where it is known as a "foreign key"). Codd showed that data relationships of arbitrary complexity can be represented using this simple set of concepts.
- **RDBMS notation for table, column and row.**

| | |
|---|---|
| Table: | Relation |
| Column: | Attribute |
| Row: | Tuple |

**UNIT III**

**VII. (a). Define Normalization. Explain 1 NF, 2 NF and 3 NF.**

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

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*

This form dictates that all non-key attributes of a table must be functionally dependent on a candidate key i.e. there can be no interdependencies among non-key attributes.

For a table to be in 3NF, there are two requirements

- ▫ The table should be second normal form
- ▫ No attribute is transitively dependent on the primary key

**VIII. List steps to establish referential integrity to the following situation for deleting an employee (to avoid deleting in both tables).**

**Suppose company X has 2 tables, an Employee table and Employee Salary table. In the Employee table we have 2 columns---the employee ID and the employee name. In the employee salary table, we have 2 columns—the employee ID and the salary for the given ID.**

- *Referential integrity* ensures that all references from one table to another are valid. This prevents problems from occurring when changes in one table are not reflected in another.
- To illustrate the concept of referential integrity, assume that you have a table called ENGINEERS where you store information about service engineer employees. You need to add an engineer to a new office in this table:

```
INSERT INTO EMPLOYEE (EMPL_NUM,NAME,REP_OFFICE,
TITLE,HIRE_DATE) VALUES (400,�Marv Epper�,50,
�Engineer�, 10/1/93,NULL);
```

There�s nothing inherently incorrect about this statement. However, if office 50 does not yet exist, this record could potentially corrupt the data integrity.

Every office value in the ENGINEERS table should be a valid office in the OFFICES table. This rule is called a referential integrity *constraint*.

Note that a valid reference is not the same as a correct reference. Referential integrity does not correct a mistake such as assigning an engineer to the *wrong* office; it only verifies that the office actually exists

**OR**

**VIII (a) Define 4 NF. Check whether the following relation is in 4NF. If not how to make it in 4NF.**

| ***Course ID*** | ***Instructor*** | ***Text book*** |
|---|---|---|
| **MGS 404** | **Clay** | **Hansen** |
| **MGS 404** | **Clay** | **Kroenke** |
| **MGS 404** | **Drake** | **Hansen** |
| **MGS 404** | **Drake** | **Kroenke** |

**Fourth Normal Form (4NF)**

- Fourth normal form eliminates independent many-to-one relationships between columns.
- To be in Fourth Normal Form,
    - a relation must first be in Boyce-Codd Normal Form.
    - a given relation may not contain more than one multi-valued attribute.
- **By placing the multi-valued attributes in tables by themselves, we can convert the above to 4NF**
- Change to:
  COURSE-INST (Course-Id, Instructor)
  COURSE-TEXT (Course-Id, Textbook)

**b).Explain referential integrity with an example.** **7**

*Referential integrity* ensures that all references from one table to another are valid. This prevents problems from occurring when changes in one table are not reflected in another.

To illustrate the concept of referential integrity, assume that you have a table called ENGINEERS where you store information about service engineer employees. You need to add an engineer to a new office in this table:

```
INSERT INTO ENGINEERS (EMPL_NUM,NAME,REP_OFFICE,
TITLE,HIRE_DATE) VALUES (400,�Marv Epper�,50,
�Engineer�, 10/1/93,NULL);
```

There�s nothing inherently incorrect about this statement. However, if office 50 does not yet exist, this record could potentially corrupt the data integrity.

Every office value in the ENGINEERS table should be a valid office in the OFFICES table. This rule is called a referential integrity *constraint*.

Note that a valid reference is not the same as a correct reference. Referential integrity does not correct a mistake such as assigning an engineer to the *wrong* office; it only verifies that the office actually exists

**UNIT IV**

**IX. Form the SQL command for the following (Tables storing details of all the players participated in Olympics over various years) :**

**a) Create table with the attributes (player name, sex, country, participated event, medal, year).**

SQL>Create table Olympics(player name char(4),sex char(6),country char(10),participated event char(10),medal int(4),year int(4));

**b) How to insert only player name, sex and year into the table ?**

SQL>insert into Olympics values('Mary','Female','India','high jump',4,1998);

**c) Query for showing the Indian female players who participated in 1980 Olympics.**

SQL>Select sex,country,year from Olympics Groupby sex, where country="India" having count(year)=1980;

**d) Show the different Indian players participated in various Olympics in alphabetic order.**

SQL>SELECT player name,country FROM Olympics where country="India" ORDER BY player name ASC

**e) Update the country of player P.T Usha by India**

SQL>Update Olympics set Country="india" where player name="P.T Usha";

**f) Find the total gold medal won by china in 1980.**

SQL>Select medal,country,year from Olympics where year=1980 having count(medal)="gold";

**g) How u can find out the countries which are not participated in 1980 Olympics?**

SQL>select country from Olympics where country="NULL";

**OR**

**X Given two tables "Persons" Table :**

| ***P_Id*** | ***Last Name*** | ***First Name*** | ***Address*** | ***City*** |
|---|---|---|---|---|
| **1** | **Hansen** | **Ola** | **Timoteivn 10** | **Sandnes** |
| **2** | **Svendon** | **Tove** | **Borgvn** | **Sandnes** |
| **3** | **Pettersen** | **Kari** | **Storgt 20** | **Stavenger** |

| **Q_Id** | **Order No** | **P_Id** |
|---|---|---|
| **1** | **77895** | **3** |

**The "Orders" table :**

| | | |
|---|---|---|
| **2** | **44678** | **3** |
| **3** | **22456** | **1** |
| **4** | **24562** | **1** |
| **5** | **34764** | **15** |

**Form the SQL command :**

**(a) To list all the persons and their orders—if any, from the tables above. Show the output of the query as well.**

SQl> select Persons.P_Id,Orders.Order No,Persons.First name,Persons.last name from orders INNER JOIN persons ON Orders.P_Id=Persons.P_Id;

**b) To list all the persons with any orders. Show the output of the query as well**

SQL>select name from persons where P_Id="not null";