5IT 2017-18 Subject: **DataBase Management System** (1st Midterm) Marks: 10

Attempt all four questions, all questions carry equal marks

Q.1 Compare the File System with DBMS.

Answer

1. **Duplicate Data –** As all files are independent of each other, so, duplicate data may be present in more than one file.
2. **Inconsistency –** Inconsistency means different copies of same data that are not matching. Data may be inconsistent in file processing system.
3. **Poor Data Integrity –** Data integrity refers to the overall completeness, accuracy and consistency of data. But in Processing system, Poor data integrity often develops.
4. **Data is isolated and separated –** Data are separated in various files. SO, if it is needed to extract data from 2 different files, it will be require to determine which parts of each of the files are needed and how files are related to one another.
5. **Application Programs are dependent on file formats –** In processing suystem, the physical formats of file are entered in application program that process the files. Change in the file format result in program updates and a change which is time consuming and error prone.
6. **Poor Data Security-** Data is stored in different files causing the security problems
7. **Difficult to represent complex object –** Some data objects may be of variable length which might produce difficulty in representation in files.

Comparison of File System vs Database Systems (Limitations to File System)–

If data is too large, it will create so many problems.

1. Too complex to access the data using Physical Details.

Suppose a university has 500 GB of data having files F1,F2,….,Fn. If we want to retrieve the student details who scored more than 80%,then the Program will require the low level details or physical details.

The physical details will be:

- **Location**
- **Name of the file**
- **Format of the file**

To avoid this problem DBMS provides Data Independency. User can access the data from database without knowing any physical details

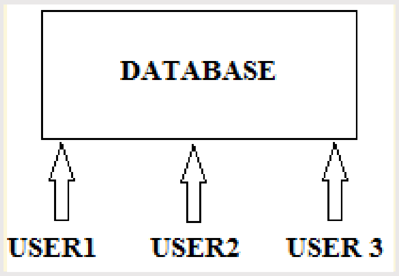
```
Select *
From table
Where condition
```

2. If data is too huge, more IO cost is required to access the data.

Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area. The officer asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of all customers. The bank officer has now two choices: either obtain the list of all customers and extract the needed information manually or ask a system programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same officer needs to trim that list to include only those customers who have an account balance of $10,000 or more. As expected, a program to generate such a list does not exist. Again, the officer has the preceding two options, neither of which is satisfactory.

The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

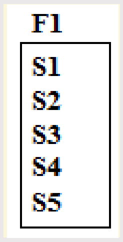
3. Concurrent Execution - Concurrent execution means when multiple users access the same data at same time.



Concurrent Operations restrict by OS

•R<->W (If one user is reading, then other user cannot be write).

•W<->R/W (If one user is writing, then other user cannot be read or write).



U1: update student S1

U2: update student S3

OS controls concurrency at file levels. i.e. OS does not allow both the users to update simultaneously even both the users update different data's i.e. S1 & S3.

To avoid this problem –

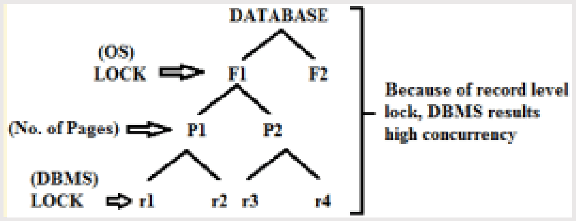
DBMS allows/controls at record level.

U1: update S1

LOCK (S1 Record)

U2: update S3

LOCK (S3 record)



4. Security - Operating System gives file level security. For example, Student is a single file which contains various fields like SID, Sname, Marks, Address .. A password can be set to Student file but not to its fields. So here, Os level security is managed by password. If password matches, then the user will see the entire data. DBMS provides different level of security. For instance, in the student file security will be decided according to

- **Faculty can see SID, Sname, Marks only**
- **Admin can see SID, Sname, Marks and Address.**

if two file are made for faculty as well as for admin, then there will be problem of data redundancy.

Q.2 Differentiate between (i) Candidate Key and Super Key (ii) Entity and Entity Set (iii) Generalization and Specialization.

Answer

(i) Basically, a *Candidate Key* is a *Super Key* from which no more Attribute can be pruned.

A *Super Key* identifies uniquely rows/tuples in a table/relation of a database. It is composed by a set of attributes that combined can assume values unique over the rows/tuples of a table/relation. A *Candidate Key* is built by a Super Key, iteratively removing/pruning non-key attributes, keeping an invariant: the newly created *Key* still need to uniquely identifies the rows/tuples.

A *Candidate Key* might be seen as a *minimal Super Key*, in terms of attributes.

Candidate Keys can be used to reference uniquely rows/tuples but from the RDBMS engine perspective the burden to maintain indexes on them is far heavier.

(ii) An entity is an object that exists and is distinguishable from other objects. For instance, John Harris with S.I.N. 890-12-3456 is an entity, as he can be uniquely identified as one particular person in the universe. An entity may be concrete (a person or a book, for example) or abstract (like a holiday or a concept). An entity set is a set of entities of the same type (e.g., all persons having an account at a bank). Entity sets need not be disjoint. For example, the entity set employee (all employees of a bank) and the entity set customer (all customers of the bank) may have members in common. An entity is represented by a set of attributes.

E.g. name, S.I.N., street, city for ``customer'' entity.

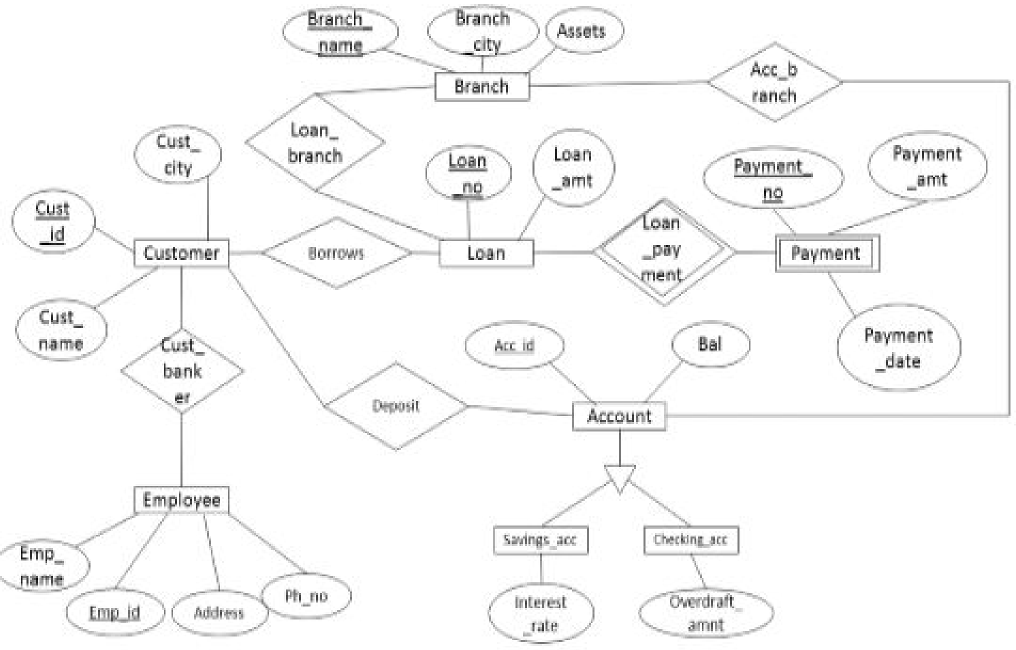
The domain of the attribute is the set of permitted values (e.g. the telephone number must be seven positive integers).

(iii) **Generalization** is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entity to make further higher level entity.

Specialization

Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, some higher level entities may not have lower-level entity sets at all.

Q.3 Generate an E-R diagram of Bank enterprise.



Q.4 Explain various types of join with examples.
Answer

1. JOIN or INNER JOIN :

In this kind of a JOIN, we get all records that match the condition in both the tables, and records in both the tables that do not match are not reported.
In other words, INNER JOIN is based on the single fact that : ONLY the matching entries in BOTH the tables SHOULD be listed.
Note that a JOIN without any other JOIN keywords (like INNER, OUTER, LEFT, etc) is an INNER JOIN. In other words, JOIN is a Syntactic sugar for INNER JOIN (see : Difference between JOIN and INNER JOIN).
2. OUTER JOIN :

OUTER JOIN retrieves
Either, the matched rows from one table and all rows in the other table Or, all rows in all tables (it doesn't matter whether or not there is a match).

There are three kinds of Outer Join :

2.1 LEFT OUTER JOIN or LEFT JOIN
This join returns all the rows from the left table in conjunction with the matching rows from the right table. If there are no columns matching in the right table, it returns NULL values.
2.2 RIGHT OUTER JOIN or RIGHT JOIN
This JOIN returns all the rows from the right table in conjunction with the matching rows from the left table. If there are no columns matching in the left table, it returns NULL values.
2.3 FULL OUTER JOIN or FULL JOIN
This JOIN combines LEFT OUTER JOIN and RIGHT OUTER JOIN. It returns row from either table when the conditions are met and returns NULL value when there is no match.
In other words, OUTER JOIN is based on the fact that : ONLY the matching entries in ONE OF the tables (RIGHT or LEFT) or BOTH of the tables(FULL) SHOULD be listed.
Note that `OUTER JOIN` is a loosened form of `INNER JOIN`.
3. NATURAL JOIN :

It is based on the two conditions :

1. the JOIN is made on all the columns with the same name for equality.
2. Removes duplicate columns from the result.

This seems to be more of theoretical in nature and as a result (probably) most DBMS don't even bother supporting this.

4. CROSS JOIN :

It is the Cartesian product of the two tables involved. The result of a CROSS JOIN will not make sense in most of the situations. Moreover, we wont need this at all (or needs the least, to be precise).
5. SELF JOIN :

It is not a different form of JOIN, rather it is a JOIN (INNER, OUTER, etc) of a table to itself.
JOINs based on Operators

Depending on the operator used for a JOIN clause, there can be two types of JOINs. They are
1. Equi JOIN
2. Theta JOIN

1. Equi JOIN :

For whatever JOIN type (INNER, OUTER, etc), if we use ONLY the equality operator (=), then we say that the JOIN is an EQUI JOIN.

2. Theta JOIN :

This is same as EQUI JOIN but it allows all other operators like >, <, >= etc.