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

Internal Assessment Test 1 – Sept. 2019

Sub:	**Database Management Systems**					Sub Code:	**17CS53**	Branch:	**ISE**	
Date:	07-09-19	Duration:	90 min's	Max Marks:	50	Sem / Sec:	V- A /B (ISE)			OBE

Scheme and Solution

1. Discuss Main characteristics of DBMS approach and how it is different from the traditional file system. [10]

1. Self-Description: A database system includes —in addition to the data stored that is of relevance to the organization— a complete definition/description of the database's structure and constraints. This meta-data (i.e., data about data) is stored in the so-called system catalog, which contains a description of the structure of each file, the type and storage format of each field, and the various constraints on the data (i.e., conditions that the data must satisfy).

The system catalog is used not only by users (e.g., who need to know the names of tables and attributes, and sometimes data type information and other things), but also by the DBMS software, which certainly needs to "know" how the data is structured/organized in order to interpret it in a manner consistent with that structure.

2. Insulation between Programs and Data; Data Abstraction:

Program-Data Independence: In traditional file processing, the structure of the data files accessed by an application is "hard-coded" in its source code. (E.g., Consider a file descriptor in a COBOL program: it gives a detailed description of the layout of the records in a file by describing, for each field, how many bytes it occupies.)

In contrast, DBMS access programs, in most cases, do not require such changes, because the structure of the data is described (in the system catalog) separately from the programs that access it and those programs consult the catalog in order to ascertain the structure of the data

3. Data Abstraction:
A data model is used to hide storage details and present the users with a conceptual view of the database. Programs refer to the data model constructs rather than data storage details

Example by which to illustrate this concept: Suppose that you are given the task of developing a program that displays the contents of a particular data file. Specifically, each record should be displayed as follows:

Record #i: value of first field value of second field ... ... value of last field

To keep things very simple, suppose that the file in question has fixed-length records of 57 bytes with six fixed-length fields of lengths 12, 4, 17, 2, 15, and 7 bytes, respectively, all of which are ASCII strings. Developing such a program would not be difficult. However, the obvious solution would be tailored specifically for a file having the particular structure described here and would be of no use for a file with a different structure.

4. Multiple Views of Data: Different users (e.g., in different departments of an organization) have different "views" or perspectives on the database. For example, from the point of view of a Bursar's

Office employee, student data does not include anything about which courses were taken or which grades were earned. (This is an example of a subset view.)

A good DBMS has facilities for defining multiple views. This is not only convenient for users, but also addresses security issues of data access.

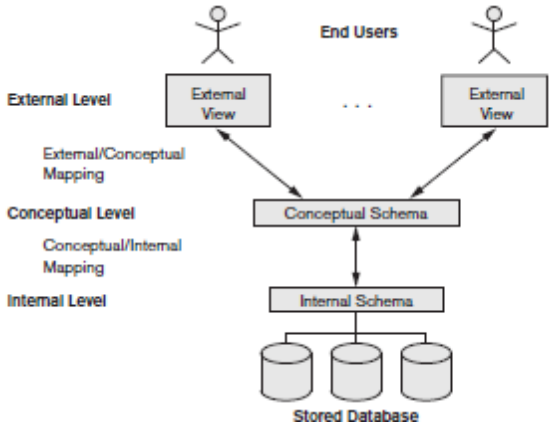
5. Data Sharing and Multi-user Transaction Processing: As you learned about (or will) in the OS course, the simultaneous access of computer resources by multiple users/processes is a major source of complexity. The same is true for multi-user DBMS's.

Arising from this is the need for concurrency control, which is supposed to ensure that several users trying to update the same data do so in a "controlled" manner so that the results of the updates are as though they were done in some sequential order

This gives rise to the concept of a transaction, which is a process that makes one or more accesses to a database and which must have the appearance of executing in isolation from all other transactions (even ones that access the same data at the "same time") and of being atomic (in the sense that, if the system crashes in the middle of its execution, the database contents must be as though it did not execute at all).

Applications such as airline reservation systems are known as online transaction processing applications.

2 a. Draw the three schema architecture with neat diagram. Why do we need mapping among the schema levels? [06]



The goal of the three-schema architecture, illustrated in the above Figure , is to separate the user applications and the physical database. In this architecture, schemas can be defined at the following three levels:

1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

2. The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. A high-level data model or an implementation data model can be used at this level.

3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the

rest of the database from that user group. A high-level data model or an implementation data model can be used at this level

The three-schema architecture can be used to explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. We can define two types of data independence:

1. Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), or to reduce the database (by removing a record type or data item). In the latter case, external schemas that refer only to the remaining data should not be affected. Only the view definition and the mappings need be changed in a DBMS that supports logical data independence. Application programs that reference the external schema constructs must work as before, after the conceptual schema undergoes a logical reorganization. Changes to constraints can be applied also to the conceptual schema without affecting the external schemas or application programs.

2. Physical data independence is the capacity to change the internal schema without having to change the conceptual (or external) schemas. Changes to the internal schema may be needed because some physical files had to be reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema.

2 b. Define the following terms i)Relational Schema ii)Relational State [04]

Relational Schema: The design of one table, containing the name of the table (i.e. the name of the relation), and the names of all the columns, or attributes.

Example: STUDENT(Name, SID, Age, GPA)

- **Degree of a Relation**: the number of attributes in the relation's schema.
- **Tuple, t, of R(A1, A2, A3, …, An):** an ORDERED set of values, < v1, v2, v3, …, vn>, where each v_i is a value from dom(A_i).

relation state *r* of the relation schema *R*(*A*1, *A*2, … , *An*), also denoted by *r*(*R*), is a set of *n*-tuples *r* = {*t*1, *t*2, … , *tm*}. Each *n*-**tuple** *t* is an ordered list of *n* values *t* =<*v*1, *v*2, ... , *vn*>, where each value *vi*, $1 \leq i \leq n$, is an element of dom (*Ai*) or is a special NULL value. The *i*th value in tuple *t*, which corresponds to the attribute *Ai*, is referred to as *t*[*Ai*] or *t*.*Ai* (or *t*[*i*] if we use the positional notation). The terms **relation intension** for the schema *R* and **relation extension** for a relation state *r*(*R*) are also commonly used.

3. Construct an E-R diagram for a Movie database Considering the following requirements:
   i. Each movie identifies by its title and year of release, it has a length in minutes and can have zero or more quotes, language.
   ii. Production companies are identified by Name, they have the address, each production company can produce one or more movies.
   iii. Actors are identified by Name and Date of Birth, they can act one or more movies and each actor has a role in movies.
   iv. Directors are identified by Name, Date of Birth, So each Director can direct one or more movies and each movie can be directed by one or more Directors.

Each movie belongs to anyone category like Horror, action, Drama etc. and ratings [10]

Movie(title, length, year,genre, plot _outline)
Production company(name,address)
Director(name,Date_ of_Birth)
Actor(name,Date_ of_Birth)
Quote(actor_name)

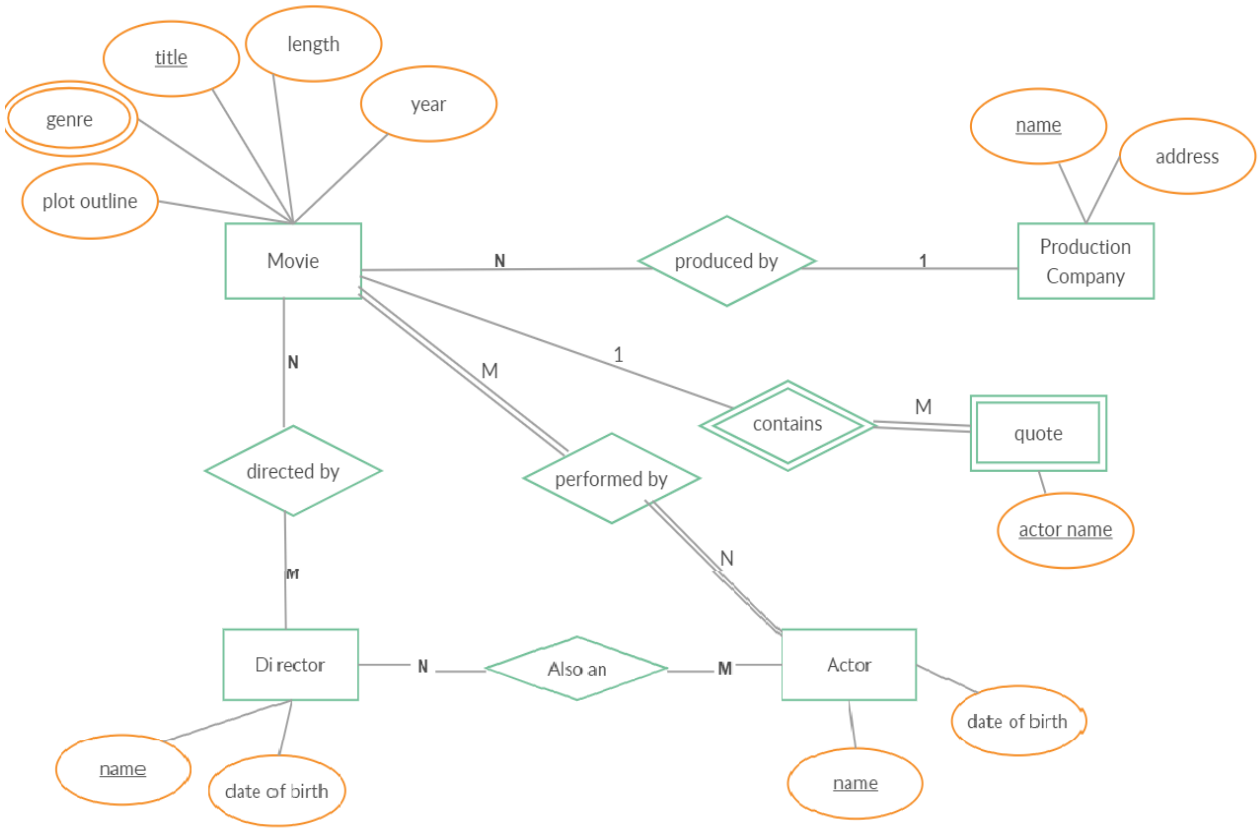
Primary keys: movie_title, Production_name, Actor_name,Director_name
Foreign Keys: Production_name, Actor_name,Director_name in the relation Movie

Relationships

1. Movie produced by production company
2. Movie performed by actor
3. Movie directed by director
4. A director also a actor

Cordinality ratio:

One production company can produce N number of Movies
M directors can deirect N number of Movies



4. Consider the following relation schema and write the Relation Algebra [10]
Sailors(sid: integer, sname: string, rating: integer, age: real)
Boats(bid: integer, bname: string, color: string)

Reserves(sid: integer, bid: integer, day: date)

i) Retrieve names of sailors who have reserved a red and a green boat.

T1←Π(sanme) (σ(color=”red”)((sailors* reserves)* boats)

T2←Π(sanme) (σ(color=”green”)((sailors* reserves)* boats)

T3←T1UT2

ii) Retrieve sid and names of sailors with age more than 20, who have Reserved a Red boat.

T1←Π(sid,sanme) (σ(color=”red” and age>20)((sailors* reserves)* boats)

iii) Retrieve the boat names which are not reserved by any Sailors.

T1←Π(banme)(boats)

T2←Π(banme) (reserves* boats)

T3←T1-T2

iv) Retrieve the sailor names who have reserved yellow boat on 01-08-2019

T1←Π(sanme) (σ(color=”yellow” and day=”01-10-2019”)((sailors* reserves)* boats)

v) Retrieve the boat names which has reserved by highest rating sailors.

T1(maxrating)← ℑ max(rating) sailors

T2← sailors s ⋈ s.rating =maxrating (t1)

T3← T2 ⋈ t2.sid=r sid reserves r

T4← Π(banme)(t3*boats)

5. Explain the following SQL Commands with example

i) insert

INSERT INTO target [(field1[, field2[, ...]])]
VALUES (value1[, value2[, ...]);

So, to add a User record for user Jim Jones, we would issue the following INSERT query:

INSERT INTO User (FirstName, LastName, UserID, Dept, EmpNo, PCType) 6
VALUES ("Jim", "Jones", "Jjones","Finance", 9, "DellDimR450");

Obviously populating a database by issuing such a series of SQL commands is both tedious and prone to error, which is another reason why database applications have front-ends. Even without a specifically designed front-end, many database systems - including MS Access - allow data entry direct into tables via a spreadsheet-like interface.

ii) Delete

Now that we know how to add new records and to update existing records it only remains to learn how to delete records before we move on to look at how we search through and collate data. As you would expect SQL provides a simple command to delete complete records. The syntax of the command is:

DELETE [table.*]
FROM table
WHERE criteria;

iii) drop

If you have already executed the original CREATE TABLE command your database will already contain a table called User, so let's get rid of that using the DROP command:

DROP TABLE User;

iv) alter

Once a table is created it's structure is not necessarily fixed in stone. In time requirements change and the structure of the database is likely to evolve to match your wishes. SQL can be used to change the structure of a table, so, for example, if we need to add a new field to our User table to tell us if the user has Internet access, then we can execute an SQL ALTER TABLE command as shown below:

ALTER TABLE User ADD COLUMN Internet BOOLEAN;

To delete a column the ADD keyword is replaced with DROP, so to delete the field we have just added the SQL is:

ALTER TABLE User DROP COLUMN Internet;

v) update

the UPDATE command, with syntax:

UPDATE table
SET newvalue
WHERE criteria;

For example, let's assume that we want to move user Jim Jones from the Finance department to Marketing. Our SQL statement would then be:

UPDATE User
SET Dept="Marketing"
WHERE EmpNo=9;

6. Draw the DBMS component module diagram; explain their interactions with every module. [10]

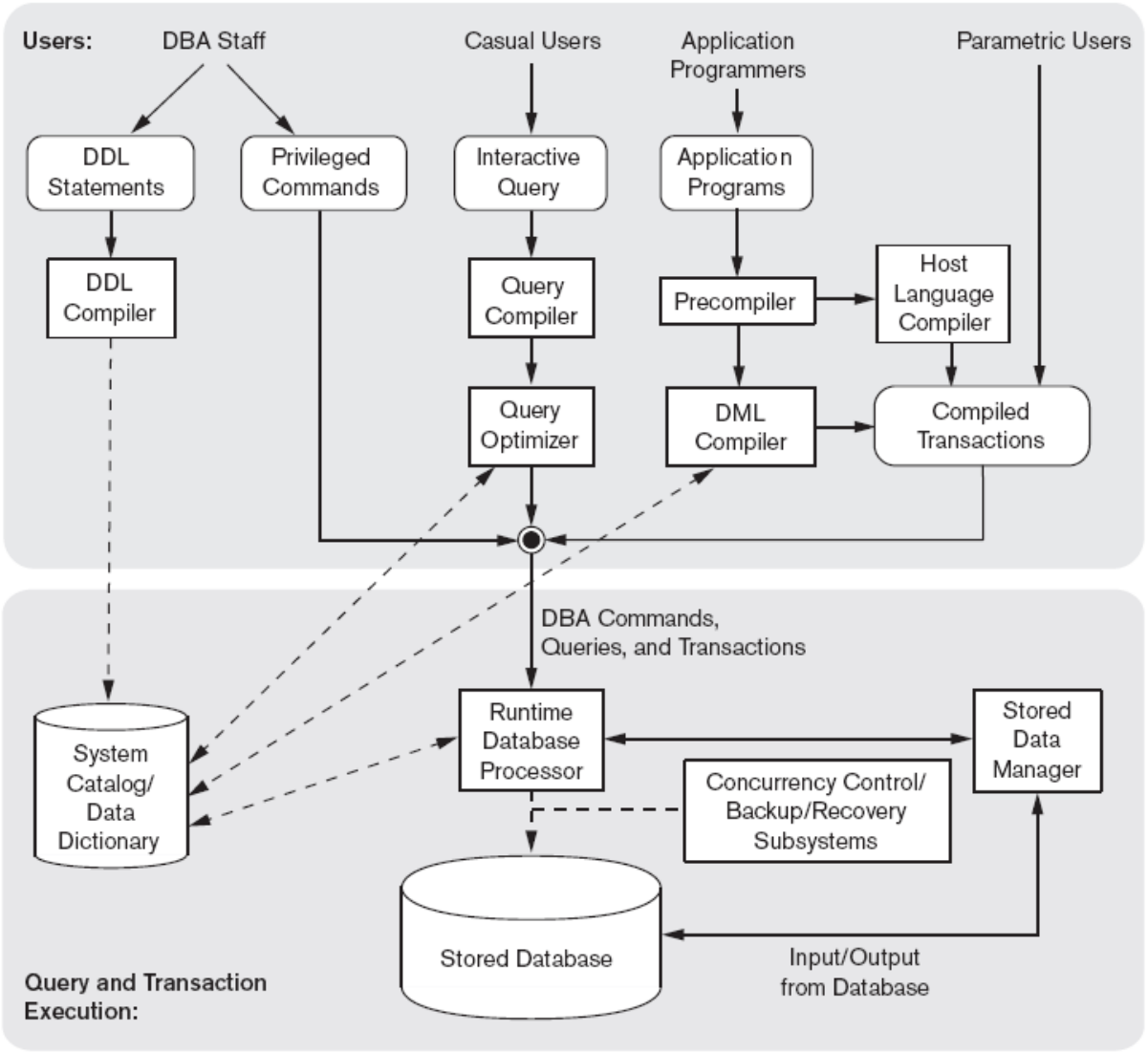


Figure 2.3
Component modules of a DBMS and their interactions.

Users

- DDL compiler
- Query compiler
- Stored data manger
- Runtime processor
- System catalog

7.Explain the select, project and join operations in relational algebra with suitable example.[10]
Selection Operator(σ)
Selection and Projection are unary operators.
The selection operator is sigma: σ

The selection operation acts like a filter on a relation by returning only a certain number of tuples.
σC(R) Returns only those tuples in R that satisfy condition C
A condition C can be made up of any combination of comparison or logical operators that operate on the attributes of R. Comparison operators: >,<,<=,>=, ≠,==

Logical operators
┐ - not, v - or
Example
Select only those Employees in the CS department:
σ Dept= 'CS'(EMP)

Projection(π)

Projection is also a Unary operator.
The Projection operator is pi: π
Projection limits the attributes that will be returned from the original relation.
The general syntax is: π attributes R
Where attributes is the list of attributes to be displayed and R is the relation.
The resulting relation will have the same number of tuples as the original relation (unless there are duplicate tuples produced).
The degree of the resulting relation may be equal to or less than that of the original relation
Project only the names and departments of the employees:
πname, dept(EMP)

THETA JOIN: Similar to a CARTESIAN PRODUCT followed by a SELECT. The condition c is called a join condition.
R(A1, A2, ..., Am, B1, B2, ..., Bn) R1(A1, A2, ..., Am) c R2 (B1, B2, ..., Bn)
EQUIJOIN: The join condition c includes one or more equality comparisons involving attributes from R1 and R2. That is, c is of the form:
(Ai=Bj) AND ... AND (Ah=Bk); 1<i,h<m, 1<j,k<n
In the above EQUIJOIN operation:
Ai, ..., Ah are called the join attributes of R1
Bj, ..., Bk are called the join attributes of R2
Example of using EQUIJOIN:
Retrieve each DEPARTMENT's name and its manager's name:
T DEPARTMENT MGRSSN = SSN EMPLOYEE
RESULT
DNAME,FNAME,LNAME
(T)
NATURAL JOIN (*):
In an EQUIJOIN R R1 c R2, the join attribute of R2 appear redundantly in the result relation R. In a NATURAL JOIN, the redundant join attributes of R2 are eliminated from R. The equality condition is implied and need not be specified.
R R1 *(join attributes of R1),(join attributes of R2) R

Example: Retrieve each EMPLOYEE's name and the name of the DEPARTMENT he/she works for:
T EMPLOYEE *(DNO),(DNUMBER) DEPARTMENT
RESULT
FNAME,LNAME,DNAME
(T)
If the join attributes have the same names in both relations, they need not be specified and we can write R R1 * R2.
Example: Retrieve each EMPLOYEE's name and the name of his/her SUPERVISOR:

8. In SQL Which command is used for table creation? Specify the list constraints in SQL, Give the proper syntax of each constraint usage in SQL with examples.[10]

CREATE TABLE name(col1 datatype, col2 datatype, …);

So, to create our User table we enter the following command:

CREATE TABLE User (FirstName TEXT, LastName TEXT, UserID TEXT, Dept TEXT, EmpNo INTEGER, PCType TEXT);

The TEXT datatype, supported by many of the most common DBMS, specifies a string of characters of any length. In practice there is often a default string length which varies by product. In some DBMS TEXT is not supported, and instead a specific string length has to be declared. Fixed length strings are often called CHAR(x), VCHAR(x) or VARCHAR(x), where x is the string length. In the case of INTEGER there are often multiple flavors of integer available. Remembering that larger integers require more bytes for data storage, the choice of int size is usually a design decision that ought to be made up front.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified
- **INDEX** - Used to create and retrieve data from the database very quickly

CREATE TABLE EMPLOYEE (… , Dno INT NOT NULL DEFAULT 1,
CONSTRAINT EMPPK PRIMARY KEY (Ssn),
CONSTRAINT EMPSUPERFK FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn) ON DELETE SET NULL ON UPDATE CASCADE,
CONSTRAINT EMPDEPTFK FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber) ON DELETE SET DEFAULT ON UPDATE CASCADE);

CREATE TABLE DEPARTMENT (… , Mgr_ssn CHAR(9) NOT NULL DEFAULT ‘888665555’, … ,
CONSTRAINT DEPTPK PRIMARY KEY(Dnumber),
CONSTRAINT DEPTSK UNIQUE (Dname),
CONSTRAINT DEPTMGRFK FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) ON DELETE SET DEFAULT ON UPDATE CASCADE);

CREATE TABLE DEPT_LOCATIONS (… , PRIMARY KEY (Dnumber, Dlocation),

```
        FOREIGN KEY (Dnumber) REFERENCES
DEPARTMENT(Dnumber)                    ON DELETE CASCADE
ON UPDATE CASCADE);
```