Date: / /
Page No.

Chotai Sharanam Maheshbhai

MCA21040

PS01CMCA53:- Database Management Systems.

MCA Sem I (CAIA)

Theory Assignment

Date: 30/10/2021
Page No.

1) Define: Data, Entity, Attribute, Datatype, Information, Database, DBMS, RDBMS, Primary key, Foreign key.

→ Data: A collection of facts is called a data. Such as, numbers, words observations and description of things. Additionally, data can be classified ~~as~~ into two types: Quantitative and Qualitative.

- Entity: Real world objects are called entities. It can be identified very easily. It can be an individual living or non-living thing.
- Attribute: Properties of an entity is known as attributes. It describes an entity thoroughly.
- Data-type:- Attributes, which is also known as domain, are rules that describe the legal values of a field type. They are used to constrain the values allowed in particular attribute for a table.
- Information: Information is a group of data processed in a meaningful way. It is well structured.
- Database: Organized collection of coherent and meaningful data, structured information recorded or stored electronically in a computer system is called Database.
- DBMS: Set of programs designed to create and maintain (by means of CRUD operations) is called a Database Management System.

- RDBMS: A DBMS which is based on relational model and stores data in a form of relations (or to be said table-structure) is known as Relational Database Management System.
- Primary key: A key through which we can uniquely identifies a row in each table is known as Primary key (PK, for short). A single table can consist Primary key made of using one or more attributes or columns.
- Foreign key: A key borrowed from another related table in order to make the relationship between two tables is called Foreign key (abbreviated as FK). FK must match actual values and data types with the related PK.

2) Full form of: DBMS, RDBMS, SQL, PL/SQL.

=> DBMS: Database Management System
RDBMS: Relational Database Management System
SQL: Structured Query Language.
PL/SQL: Procedural language: extensions to the Structured Query Language.

3) List of any four entities with atleast three attributes of them.

=> - Student: UID, Name, Class, Email-ID, Address, Joining-Date.
- Worker: Reg-No, Name, Job-Id, Salary, Dept, Hire-date, Permanent-location.
- Client: Client-Id, Name, type, contact
- Book: U-No, shelf, Issued-by, Author, Published-On-Date.

4) Difference between DBMS and RDBMS.

→ In DBMS, data are stored in a file format. Meanwhile, tables are used to store in RDBMS.
- In DBMS, data are usually stored in a hierarchical arrangement and it deals with small amount of data. In RDBMS, data are stored in a table format where data can be extracted easily, hence, it can deal with large quantity of data.
- DBMS supports single user at a time. In contrast to it, RDBMS allows multiple users to access database simultaneously.
- RDBMS supports distributed databases, unlike DBMS.
- Data can be normalized in RDBMS, but not in DBMS.

5) Explain the benefits of DBMS.

→ DBMS makes it possible for end users to create, read, update and delete data in database. It is a layer between programs and data. Compared to the file based data management system, DBMS has many advantages. Some of those advantages are as follows:
- Reducing Data Redundancy:- The file based data management systems contained multiple files that were stored in many different locations in a system or even across multiple systems. Because of this, there were sometimes

multiple copies of the same file which lead to data redundancy. This is prevented in a database as there is a single database and any change in it is reflected immediately. Hence, there is no chance of encountering duplicate data.

- Sharing of Data: Users of a database can share the data among themselves. There are various levels of authorisation to access the data, and consequently data can only be shared based on the correct authorisation protocols being followed. Many remote users can ~~only~~ also access the database simultaneously and share the data between themselves.
- Data Integrity: Data is accurate and consistent in the database. Data Integrity is important as there are multiple databases in DBMS. All of these databases contain data that is visible to multiple users. Therefore it is necessary to ensure that the data is correct and consistent in all the databases and for all the users.
- Data Security: Data security is a vital concept ~~of~~ in a database. Only authorised users should be allowed to access the database and their identity should be authenticated using a username and password. Unauthorised users should not be allowed to access the database under any circumstances as it violates the integrity constraints.

- Privacy: The privacy rule in a database means only the authorized users can access a database according to its privacy constraints. There are levels of database access and a user can only view the data he/she is allowed to. For instance, Access constraints in social networking sites are different for various accounts a user may want to access.
- Backup and Recovery: DBMS automatically takes care of backup and recovery. Users don't need to backup data periodically because this is taken care of by DBMS. It also restores the database after a crash or system failure to its previous condition / state.
- Data Consistency: Data consistency is ensured in a database because there is no data redundancy. All data appears consistently across the database and the data is same for all the users viewing the database. Moreover, any changes made to the database are immediately reflected to all the users and there is no data inconsistency.

6) Difference between Primary Key, Foreign Key and Alternate Key.

→ Databases are used to store massive amounts of information which is stored across multiple tables. Each table might be running into thousands of rows and there will be many duplicate rows with redundant information. To deal with this and to relate the multiple tables present in database, we

use SQL keys.

→ Basically, Super key in a SQL, is a single key or group of multiple keys that can uniquely identify tuples in a table. It can contain redundant attributes that might not be important for identifying tuples. Then comes candidate keys, which are subset of super keys and contain only those attributes which are required to uniquely identify tuples.

- From set of the candidate keys database designer or database administrator (DBA) chooses a candidate key and calls it or declares it as a primary key of that particular table and others in the set of candidate keys are known as alternate keys.
- Foreign key is an attribute which is a primary key in its parent table but is included as an attribute in another host table. It allows us to create a relationship between two tables within the database. Foreign keys may accept non-unique and null values.

7) List various data types in oracle. Write the difference between CHAR, VARCHAR and NVARCHAR datatype.

⇒ According to 'Application Developer's Guide - Fundamentals 10g Release 2 (10.2) by ORACLE Database, the datatypes supported by oracle database can be divided into the following categories:

- Oracle built-in datatypes: for characters, numbers, datetime, raw data, large objects (LOBs), and row addresses (ROWID)
- ANSI datatypes and data types from the IBM products SQL/DS and DB2,
- User defined types, which use Oracle builtin datatypes and other user-defined datatypes.
- Oracle supplied types, which are SQL-based interfaces for defining new types.

* Oracle precompilers recognize other datatypes in embedded SQL programs, called external datatypes.

- Oracle built-in datatypes are listed below:
  * Character Datatypes:
    - CHAR
    - NCHAR
    - VARCHAR2 and VARCHAR
    - NVARCHAR2
    - CLOB
    - NCLOB
    - LONG
  * Number Datatype
  * Date Datatype
  * Binary Datatype
    - BLOB
    - BFILE
    - RAW
    - LONG RAW.
- CHAR v/s VARCHAR v/s NVARCHAR :-

- NVARCHAR (and NCHAR) can store Unicode characters, whereas, char and varchar cannot store Unicode Characters.
- CHAR (and NCHAR) are fixed-length which will reserve storage space for number of characters you specify even if you don't use up all that space. Meanwhile, VARCHAR and NVARCHAR are variable-length which will only use up spaces for the characters you store.
- CHAR can ~~take~~ be of size 2000 bytes per row, maximum. Whereas, Varchar and Nvarchar can store upto 4000 bytes/row.

8) List DDL Statements. Also explain the syntax of each command with an example.

→ Data Definition Language actually consists of the SQL commands that can be used to define the database schema, to create and modify and delete the structure of database objects in database.

- ~~list of~~ DDL commands :-

1) CREATE : to create the database ~~ob~~ or its objects, such as table, index, function, view, procedure, trigger.
   - There are two CREATE statements, in SQL, available :
     1) CREATE DATABASE <Name of Database>;
        ↳ to make new database.
        e.g. CREATE DATABASE Hospital;

2) CREATE TABLE <Name of Table>
(
column1 data-type(size),
column2 data-type(size)
);

↳ Using this command, you can create new table within database, whose columns are attributes ~~of~~ of that table alongwith its datatype and maximum assignable size.

e.g.

```
CREATE TABLE Patient
(
Patient_NO  int(3),
Name  varchar(20),
DOB  date,
Admission_date  date,
Discharge_date  date,
Contact_No  number(10)
);
```

This is how we can introduce schema of Patient table. Patient_No, Name, DOB, etc. are columns of ~~the~~ table. And int, varchar, date, etc. are domains or datatypes of each respective attributes.

2) ALTER: to alter the structure of the database. We can add, delete/drop or modify columns in the existing table. ~~Alt~~ Also to add or delete various constraints on the existing table.

- ALTER TABLE - ADD
  ↳ ADD is used to add columns into the existing table. Sometimes we may require to add additional information, in that case, we can use this instead of redefining whole DB.
  Syntax: ALTER TABLE table_name
  ADD (column1 datatype (size),
  column2 datatype (size)...
  );
  e.g. ALTER TABLE Patient
  Add Pending_Bill_Amt int (4);
- ALTER TABLE - DROP
  ↳ DROP COLUMN is used to drop column in a table.
  Syntax: ALTER TABLE tableName
  DROP COLUMN column_Name;
  e.g.: Alter Table Patient
  Drop Column Contact_No;
- ALTER TABLE - MODIFY
  ↳ It is used to modify the existing columns in a table.
  Syntax: ALTER TABLE table_name
  MODIFY column_name column_type;
  e.g. ALTER TABLE Patient
  Modify Name Varchar2 (50);
  ↳ This Query will change datatype of 'Name' attribute from VARCHAR to VARCHAR2 and size will be of 50 bytes, instead of 20, as declared earlier.

3) RENAME: to rename an object existing in the database.

- To rename table

Syntax: ALTER TABLE table-name
RENAME TO new-table-name;

e.g.: ALTER TABLE Patient
RENAME TO Admitted_Patients;

This is how we can give a relative name to any table afterwards.

- To rename Column

Syntax: ALTER TABLE table_name
RENAME COLUMN old-name To newname;

e.g. ALTER TABLE Patient
RENAME COLUMN DOB TO BIRTHDATE;

Using this command, you can solve typographical-errors in schema of database.

4) TRUNCATE: to remove all records from a table, including all spaces allocated for the records are removed.

It is used to mark the extents of a table for deallocation / empty for reuse. This statement is logically, rather physically.

Syntax: TRUNCATE TABLE table-name;

e.g. TRUNCATE TABLE Patient;

After executing this command, you won't able to get a single patient's information, but can see table's structure (using DESC command, of course).

5) DROP: to delete objects from database, such as existing database, table, index, or view.
syntax: DROP Object object_name;
e.g. DROP DATABASE Hospital;
↳ To remove whole database from the system/server. You must have admin's rights to execute this.
e.g. DROP TABLE Admitted_Patients;
↳ To delete table along with its schema.

9) List DML Statements. Also explain the syntax of each command with an example.
→ SQL Commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL Statements. These statements donot implicitly commit the current transaction.
- List of DML statements available in oracle:
Insert, Update, Delete, Select and other statements (which we are not going to discuss here) are: Call, explain plan, lock table, merge.
- "The select statement is a limited form of DML statement in that it can only access data in the database. It cannot manipulate data stored in the database, although it can manipulate the accessed data before returning the results of the query." - Oracle doc 11g (11.1)

Some authors also say that SELECT command has a special standing and mention DQL or Data Query Language. But since syllabus of this subject relies on oracle (10g), I, personally, am considering oracle's doc, hence, select is DML statement.

- 1) INSERT: to insert data into table.
  ↳ The INSERT INTO statement of SQL is used to insert a new row in a table. There are two ways of using INSERT INTO statement for inserting rows:
  1) Only Values: First method is to specify only the value of data to be inserted without the column names.
  Syntax: INSERT INTO table_name VALUES (value1, value2, ... valuen);
  e.g. INSERT into Patient VALUES (1, 'SBA', '18-JAN-2011', '20-OCT-2020', '22-OCT-2020', 1234567890);
  2) Column name and values both: In the second method, we will specify both the columns ~~and~~ which we want to fill and their corresponding values as shown below:
  INSERT INTO table_name (column1, col2, ... coln)
  VALUES (value1, value2, ...)
  In this method we can exclude optional columns. For example, if patient is being admitted today, Date_of_Discharge field should be empty. Using second method, we

can exclude 'Discharge_Date' column, unlike first method.

e.g. INSERT INTO Patient (Patient_NO, Name, Admission_date, ContactNo)
Values (2, 'CBC', '20-APR-2013', 1555566666);

Note: Columns for which the values are not provided are filled by null unless default values were provided by DB designer.

- Using SELECT in INSERT INTO statement
  ↳ We can use the SELECT statement with INSERT INTO statement to copy rows from one table and insert them into another table. The use of this statement is similar to that of INSERT INTO statement. The difference is that the SELECT statement is used here to select data from a different table. The different ways of using INSERT INTO SELECT statements are shown below:

⇒ Inserting all columns of a table: We can copy all the data of a table and insert into a different table.
INSERT INTO first_table SELECT * from second_table;

⇒ Inserting specific columns of a table: To copy only those columns of a table which we want to insert into different table.
INSERT INTO first_table (names of columns 1)
Select namesOfCols2 FROM second_table;

=> Copying specific rows from a table: By using where clause in above queries we can copy specific rows from second table to first table.

INSERT INTO table1 SELECT * from table2 WHERE condition;

e.g. INSERT INTO ~~abm~~ admitted_patients SELECT * from patients WHERE Discharge date = NULL;

If patients are admitted and not being discharged yet, we are putting their data into ADMITTED PATIENTS table.

2) UPDATE: to update the data of an existing table in database. We can update single columns as well as multiple columns using update statement as per our requirements.

Basic syntax: UPDATE tablename SET column1 = value1, col2 = val2, ... WHERE condition;

SET statement is used to set new values to the particular column and WHERE clause is used to select the rows for which the columns are needed to be updated. If we have not used the WHERE clause then the columns in all the ROWS will be updated.

- Updating Single columns:

e.g. UPDATE Patient SET Name = 'Sharanam' where Patient_No = 68;

Here we are updating single column ('Name') only. And also looking at WHERE clause, our

Date: / /
Page No.

~~to~~ condition will select single row, due to primary key of the Patient table.
If we remove that where clause, name of every patient will be stored/replaced with name 'Shashanam'.

3) DELETE: to delete existing records from a table. We can delete a single record or multiple records depending upon the condition we specify in the WHERE clause.
Syntax: DELETE FROM tablename
WHERE condition;
If we omit the WHERE clause then all of the records will be deleted.
e.g. DELETE FROM Patient WHERE contact-no = 1555566666;
It will delete the entry we insested previously, having contact number 1555566666.

4) SELECT: to retrieve or fetch data from a database. We can fetch either the entire table or according to some specified rules. The data returned is stored in a result table. (Which is also known as result-set).
With the SELECT clause of Select statement, we specify the columns that we want to be displayed in the query result and, optionally, which column headings we prefer to see above the result table. The select clause is the first clause and is one of the last clauses of the select statement that the DB server evaluates. The reason for this is that before we can determine what to

include in the final result set, we need to know all of the possible column that could be included in the final result set.

Syntax: SELECT column1, column2, column3, ...
FROM table_name
[WHERE condition]
[ORDER BY column1, ..., columnN [ASC|DESC]]
[GROUP BY columnname]

eg. SELECT Patient_No, Name
FROM Patient;

↳ This query will give table of 2 columns having all the rows as an output.

eg. SELECT * FROM Patient
WHERE Patient_No = 1;

↳ This will give record of only one record whose patient number is 1, but of all the columns of patient table.

- We can sort the output using ORDER BY clause:

eg. SELECT Patient_No, Name FROM Patient
ORDER BY Name ASC;

↳ Output will be sorted in alphabetical order (of column Name) here.
ASC stands for Asceding, which is optional and default.

10) Explain various operators with an example.

→ An SQL operator is a special word or character used to perform tasks or operations. These tasks can be anything from complex comparisons, to basic

Date: / /
Page No.

arithmetic operations. SQL Operators are primarily used within the WHERE clause of an SQL statement. This is the part of the statement that is used to filter data by a specific condition or conditions. There are two general classes of operators: unary and binary. Oracle Database Lite SQL also supports set operators.

1) Unary operators: It uses only one operand and typically appears in the following format:
operator operand
2) Binary operators: It uses two operands. A binary operators appears with its operands in the following format:
operand1 operator operand2.
3) Set Operators: They combines sets of rows returned by queries, instead of individual data items. All set operators have equal precedence. Oracle DB Lite supports the following set operators.
- Union
- Union All
- Intersect
- Minus

- The levels of precedence among Oracle DB Lite SQL operators from high to low are listed in the following table. Operators listed on the same line have the same level of precedence.

Precedence level	SQL Operator
1	Unary +- arithmetic operators, PRIOR
2	* / arithmetic operators
3	Binary +- arithmetic operators, || character op-

4	All comparison operators
5	NOT logical operator
6	AND logical operator
7	OR logical operator

4) Other operators : Other operators with special formats accept more than two operands. If an operator receives a null operator, the result is always null. The only operator that does not follow this rule is Concat.

- Arithmetic operators : They manipulate numeric operands. The '-' operator is also used in date arithmetic. Supported arithmetic operators are listed ~~in~~ here: (examples are in curly brackets.)

Operator	Description
+ (unary)	Makes operand positive. {SELECT +3 FROM dual;}
- (unary)	Negates operand {SELECT -4 FROM dual;}
/	Divisions (numbers and dates) {SELECT SAL/10 FROM EMP;}
*	Multiplication {SELECT 5*SAL FROM EMP;}
+	Addition (numbers and dates) {SELECT SAL+200 FROM EMP;}
-	Subtraction (numbers and dates) {SELECT SAL-100 FROM EMP;}

- Character Operators: Character operators used in expressions to manipulate character strings are listed below:

Date: / /
Page No.

Operator - Description
|| Concatenates character strings
{ SELECT 'Name of employee is: ' ||
ENAME FROM EMP; }

↳ Concatenating character strings:
With oracle db lite, you can concatenate character strings with the following results.
- Concatenating two character strings results in another character string.
- Oracle DB lite preserves trailing blanks in character strings by concatenation, regardless of the strings' datatypes.
- Oracle DB Lite provides the CONCAT character function as an alternative to the vertical bar operator. For example, SELECT Concat(Concat (EName, 'is a'), job) FROM EMP WHERE SAL > 2000;

This returns the following output:

CONCAT(CONCAT(ENAME	
KING	is a PRESIDENT
BLAKE	is a MANAGER
CLARK	is a MANAGER
JONES	is a MANAGER
FORD	is a ANALYST
SCOTT	is a ANALYST

6 rows selected.

- Oracle DB lite treats zero length character strings as nulls. When you concatenate a zero length character string with another operand the result is always the other operand. A null value

can only result from the concatenation of two null strings.

- Comparison Operators :- Comparison operators used in conditions that compare one expression with another are listed in following table. The result of a comparison can be TRUE, FALSE or UNKNOWN.

Operator	Description
=	Equality Test {SELECT ENAME FROM EMP WHERE SAL = 150;}
!=, ^=, <>	Inequality test {SELECT ENAME FROM EMP WHERE SAL ^= 150;}
>	Greater than test {SELECT NAME, JOB FROM EMP WHERE SAL > 3000;}
<	Less than test {SELECT * FROM EMP WHERE SAL < 3000;}
>=	Greater than or equal to test {SELECT * FROM PRICE WHERE MINPRICE >= 20;}
<=	Less than or equal to test {SELECT * FROM EMP WHERE SAL <= 2000;}
IN	"Equivalent to any member of" test. Equivalent to "=ANY". {SELECT * FROM EMP WHERE ENAME IN ('SMITH', 'WARD');}
ANY/SOME	Compares a value to each value in a list or returned by a query. Must be preceded by =, !=, >, <, <= or >=. Evaluates to FALSE if the query returns no rows. {SELECT * from dept where LOC = SOME ('NEWYORK', 'DALLAS');}
NOT IN	Equivalent to "=!ANY". Evaluates to

FALSE if any member of the set is NULL. {SELECT * FROM DEPT WHERE LOC NOT IN ('NEW YORK','DALLAS'); }

ALL	Compares a value with every value in a list or returned by a query. Must be preceded by =, !=, >, <, <=, or >=. Evaluates to TRUE if the query returns no row. {SELECT * FROM emp where sal >=ALL (1400, 3000); }
[NOT] BETWEEN x and y	[NOT] greater than or equal to x and less than or equal to y. {SELECT ENAME, JOB FROM EMP WHERE SAL BETWEEN 3000 AND 5000; }
Exists	TRUE if a subquery returns at least one row. {SELECT * FROM EMP WHERE EXISTS (SELECT ENAME FROM EMP WHERE MGR IS NULL); }
x [NOT] LIKE y [ESCAPE z]	TRUE if x does [not] match the pattern y. Within y, the character "%" matches any string of zero or more characters except NULL. The character "_" matches any single character. Any character following escape is interpreted literally, useful when y contains a percent (%) or underscore (_). {SELECT * FROM EMP WHERE ENAME LIKE '%E%'; }
IS [NOT] NULL	Tests for nulls. This is the only operator that should be used to test for nulls. {Select * from emp where comm is not null and sal >1500;

- Logical Operators: They manipulate the results of conditions.

Operator	Description
NOT	Returns TRUE if the following condition is FALSE. Returns FALSE if it is TRUE. If it is UNKNOWN, it remains UNKNOWN. {SELECT * FROM EMP WHERE NOT (Job IS NULL);} {SELECT * FROM EMP WHERE NOT (sal BETWEEN 1000 AND 2000);}
AND	Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE; otherwise returns UNKNOWN. {SELECT * FROM EMP WHERE job = 'CLERK' AND deptno = 10;}
OR	Returns TRUE if either component condition is TRUE. Returns FALSE if both are FALSE. Otherwise, returns UNKNOWN. {SELECT * FROM emp WHERE job = 'CLERK' OR deptno = 10;}

- Set Operators: They combine the results of two queries into a single result.

Operator	Description
UNION	Returns all distinct rows selected by either query. {SELECT * FROM (select ename from emp where job = 'clerk' UNION select ename from emp where job = 'ANALYST');}
UNION ALL	Returns all rows selected by either query, including all duplicates.

Date: / /
Page No.

INTERSECT and INTERSECT ALL	Returns all distinct rows selected by both queries. {SELECT * FROM order_list1 INTERSECT SELECT * FROM order_list2;}
MINUS	Returns all distinct rows selected by the first query but not the second. {SELECT * FROM (SELECT SAL FROM EMP WHERE JOB = 'PRESIDENT' MINUS SELECT SAL FROM EMP WHERE JOB = 'MANAGER');}

Note: The syntax of for INTERSECT ALL is supported but it returns same result as INTERSECT.

- Other Operators : (used by Oracle DB Lite)

Operator	Description
(+)	Indicates that the preceding column is the outer join column in a join. {SELECT ENAME, DNAME FROM EMP, DEPT WHERE DEPT.DEPTNO = EMP.DEPTNO (+);}
PRIOR	Evaluates the following expression for the parent row of the current row in a hierarchical, or tree-structured query. In such a query, you must use this operator in the CONNECT BY clause to define the relationship between parent and child rows. {SELECT EMPNO, ENAME, MGR FROM EMP CONNECT BY PRIOR EMPNO = MGR;}