

Software enginnering Assignment

MODULE 5 - Database



March 22, 2022

Tops Technologies

905, Samedh Tower, CG Rd, Ahmedabad.

|  |  |
| --- | --- |
| Q 1 | What do you understand By Database? |
| A 1 | In computer, database is used to store, organize and large amount of data. Using database, one can retrieve / fetch, update and delete the data systematically. |
|  |  |
| Q 2 | What is Normalization? |
| A 2 | Database normalization is the process of structuring a database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. |
|  |  |
| Q 3 | What is Difference between DBMS and RDBMS? |
| A 3 | |  |  | | --- | --- | | **RDBMS** | **DBMS** | | Data stored is in table format | Data stored is in the file or navigational or hierarchical format | | Multiple data elements are accessible together | Individual access of data elements | | Data in the form of a table are linked together | No connection between data | | Support distributed database | No support for distributed database | | Data is stored in a large amount | Data stored is a small amount | | Multiple layers of security while data handling. | Low security while handling data | | Requires higher quality of operating system & hardware. | Requires lower quality of operating system & hardware. | | Data fetching is fast because of relational approach. | Data fetching is slower for the large amount of data. | | Higher software and hardware necessities. | Low software and hardware necessities. | | **Example**. Oracle, MS SQL Server. | **Example**. Dbase, Fox base, Microsoft Access. | |
|  |  |
| Q 4 | What is MF Cod Rule of RDBMS Systems? |
| A 4 | Codd's twelve rules are a set of thirteen rules (numbered zero to twelve) proposed by Edgar F. Codd, a pioneer of the relational model for databases, designed to define what is required from a database management system in order for it to be considered relational, i.e., a relational database management system (RDBMS).  **Rule zero**  This rule states that for a system to qualify as an **RDBMS**, it must be able to manage database entirely through the relational capabilities.  **Rule 1: Information rule**  All information (including metadata) is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.  **Rule 2: Guaranteed Access**  Each unique piece of data (atomic value) should be accessible by: **Table Name + Primary Key (Row) + Attribute (column)**.  **NOTE:** Ability to directly access via POINTER is a violation of this rule.  **Rule 3: Systematic treatment of NULL**  Null has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Also, Primary key must not be null, ever. Expression on NULL must give null.  **Rule 4: Active Online Catalog**  Database dictionary (catalog) is the structure description of the complete Database and it must be stored online. The Catalog must be governed by same rules as rest of the database. The same query language should be used on catalog as used to query database.  **Rule 5: Powerful and Well-Structured Language**  One well-structured language must be there to provide all manners of access to the data stored in the database. Example: **SQL**, etc. If the database allows access to the data without the use of this language, then that is a violation.  **Rule 6: View Updating Rule**  All the view that are theoretically updatable should be updatable by the system as well.  **Rule 7: Relational Level Operation**  There must be Insert, Delete, and Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.  **Rule 8: Physical Data Independence**  The physical storage of data should not matter to the system. If say, some file supporting table is renamed or moved from one disk to another, it should not affect the application.  **Rule 9: Logical Data Independence**  If there is change in the logical structure (table structures) of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.  **Rule 10: Integrity Independence**  The database should be able to enforce its own integrity rather than using other programs. Key and Check constraints, trigger etc., should be stored in Data Dictionary. This also make RDBMS independent of front-end.  **Rule 11: Distribution Independence**  A database should work properly regardless of its distribution across a network. Even if a database is geographically distributed, with data stored in pieces, the end user should get an impression that it is stored at the same place. This lays the foundation of distributed database.  **Rule 12: Non-subversion Rule**  If low level access is allowed to a system it should not be able to subvert or bypass integrity rules to change the data. This can be achieved by some sort of looking or encryption. |
|  |  |
| Q 5 | What do you understand By Data Redundancy? |
| A 5 | Data redundancy means when the same piece of data is stored in two or more separate places |
|  |  |
| Q 6 | What is DDL Interpreter? |
| A 6 | DDL Interpreter DDL expands to Data Definition Language. DDL Interpreter as the name suggests **interprets the DDL statements such as schema definition statements like create, delete, etc**. The result of this interpretation is a set of a table that contains the meta-data which is stored in the data dictionary. |
|  |  |
| Q 7 | What is DML Compiler in SQL? |
| A 7 | DML compiler **translates DML statements in a query language into a low-level instruction** and the generated instruction can be understood by Query Evaluation Engine. |
|  |  |
| Q8 | What is SQL Key Constraints? Write an example of SQL Key Constraints. |
| A 8 | SQL constraints are used to specify rules for the data in a table.  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](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value * [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different * [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table * [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Prevents actions that would destroy links between tables * [CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that the values in a column satisfies a specific condition * [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column if no value is specified * [CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly |
|  |  |
| Q 9 | What is save Point? How to create a save Point write a Query? |
| A 9 | Save point is a command in SQL that is used with the rollback command.  It is a command in Transaction Control Language that is used to mark the transaction in a table.  Consider you are making a very long table, and you want to roll back only to a certain position in a table then; this can be achieved using the save point.  If you made a transaction in a table, you could mark the transaction as a certain name, and later on, if you want to roll back to that point, you can do it easily by using the transaction's name.  Save point is helpful when we want to roll back only a small part of a table and not the whole table. In simple words, we can say save point is a bookmark in SQL. |
|  |  |
| Q 10 | What is trigger? How to create a Trigger in SQL? |
| A 10 | Triggers are the SQL codes that are automatically executed in response to certain events on a particular table. These are used to maintain the integrity of the data. There cannot be two triggers with similar action time and event for one table. |

|  |  |
| --- | --- |
| Task 1 | Create student and exam tables. |
| Ans 1 | Create database assignment;  Create table student  (  RollNo int(4) PRIMARY KEY AUTO\_INCREMENT,  Name varchar(30) not null,  Branch varchar(20) not null  )  insert into student (Name, Branch) values ('Jay', 'Computer Science') ;  insert into student (Name, Branch) values ('Suhani', 'Electronics & Communication') ;  insert into student (Name, Branch) values ('Kirti', 'Electronics & Communication')  Create table exam  (  RollNo int(4) Not Null,  S\_code varchar(10) not null ,  Marks varchar(10) not null,  P\_code varchar(10) not null,  foreign key (RollNo) references student (RollNo)  )  insert into exam (RollNo, S\_code, Marks, P\_code) values (1, 'CS11', 50, 'CS');  insert into exam (RollNo, S\_code, Marks, P\_code) values (1, 'CS12', 60, 'CS');  insert into exam (RollNo, S\_code, Marks, P\_code) values (2, 'EC101', 66, 'EC');  insert into exam (RollNo, S\_code, Marks, P\_code) values (2, 'EC101', 70, 'EC');  insert into exam (RollNo, S\_code, Marks, P\_code) values (3, 'EC102', 45, 'EC');  insert into exam (RollNo, S\_code, Marks, P\_code) values (3, 'EC102', 50, 'EC') |
|  |  |
| Task 2 | Create table with personal data. |
| Ans 2 | Create table Person  (  FirstName varchar(30),  LastName varchar(30),  Address varchar(50),  City varchar(20),  Age int(3)  )  INSERT INTO `person`(`FirstName`, `LastName`, `Address`, `City`, `Age`) VALUES ('Mickey','Mouse','123 Fantasy Way','Anheim','73');  INSERT INTO `person`(`FirstName`, `LastName`, `Address`, `City`, `Age`) VALUES ('Bat','Man','321 Cavern Ave','Gotham','54');  INSERT INTO `person`(`FirstName`, `LastName`, `Address`, `City`, `Age`) VALUES ('Wonder','Woman','987 Truth Way','Paradise','39');  INSERT INTO `person`(`FirstName`, `LastName`, `Address`, `City`, `Age`) VALUES ('donald','Duck','555 Quack Street','Mallard','65');  INSERT INTO `person`(`FirstName`, `LastName`, `Address`, `City`, `Age`) VALUES ('Bugs','Bunny','567 Carrot Street','Rascal','58');  INSERT INTO `person`(`FirstName`, `LastName`, `Address`, `City`, `Age`) VALUES ('Wiley','Coyote','999 Acme Way','Canyon','61');  INSERT INTO `person`(`FirstName`, `LastName`, `Address`, `City`, `Age`) VALUES ('Cat','Woman','234 Perrfect Street','Hairball','32');  INSERT INTO `person`(`FirstName`, `LastName`, `Address`, `City`, `Age`) VALUES ('Tweety','Bird','543','Itotltaw','28') |
|  |  |
| Task 3 | Create Employee and Incentive tables. |
| Ans 3 | CREATE TABLE Employee(  Employee\_ID INT(3) PRIMARY KEY AUTO\_INCREMENT,  First\_Name VARCHAR(30) NOT NULL,  Last\_Name VARCHAR(30) NOT NULL,  Salary INT(7) NOT NULL,  Joining\_date DATETIME NOT NULL,  Department VARCHAR(20) NOT NULL  );  INSERT INTO employee (First\_Name, Last\_Name, Salary, Joining\_date, Department)  VALUES ('John', 'Abraham', 1000000, '2013-01-01', 'Banking');  INSERT INTO employee (First\_Name, Last\_Name, Salary, Joining\_date, Department)  VALUES ('Michael', 'Clarke', 800000, '2013-01-01', 'Insurance');  INSERT INTO employee (First\_Name, Last\_Name, Salary, Joining\_date, Department)  VALUES ('Roy', 'Thomas', 700000, '2013-02-01', 'Banking');  INSERT INTO employee (First\_Name, Last\_Name, Salary, Joining\_date, Department)  VALUES ('Tom', 'Jose', 600000, '2013-02-01', 'Insurance');  INSERT INTO employee (First\_Name, Last\_Name, Salary, Joining\_date, Department)  VALUES ('Jerry', 'Pinto', 650000, '2013-02-01', 'Insurance');  INSERT INTO employee (First\_Name, Last\_Name, Salary, Joining\_date, Department)  VALUES ('Philip', 'Mathew', 750000, '2013-01-01', 'Services');  INSERT INTO employee (First\_Name, Last\_Name, Salary, Joining\_date, Department)  VALUES ('TestName1', '123', 650000, '2013-01-01', 'Services');  INSERT INTO employee (First\_Name, Last\_Name, Salary, Joining\_date, Department)  VALUES ('TestName2', 'Lname%', 600000, '2013-02-01', 'Insurance')  Create table Incentive  (  Employee\_ref\_ID Int(3) Not Null,  Incentive\_date Date NOT Null,  Incentive\_amount Int(5),  FOREIGN KEY (Employee\_ref\_ID) REFERENCES employee (Employee\_ID)  )  Insert into Incentive (Employee\_ref\_ID, Incentive\_date, Incentive\_amount) values (1, '2013-02-01', 5000);  Insert into Incentive (Employee\_ref\_ID, Incentive\_date, Incentive\_amount) values (2, '2013-02-01', 3000);  Insert into Incentive (Employee\_ref\_ID, Incentive\_date, Incentive\_amount) values (3, '2013-02-01', 4000);  Insert into Incentive (Employee\_ref\_ID, Incentive\_date, Incentive\_amount) values (1, '2013-01-01', 4500);  Insert into Incentive (Employee\_ref\_ID, Incentive\_date, Incentive\_amount) values (2, '2013-01-01', 3500) |
|  |  |
| Task 3a | Get First\_Name from employee table using Tom name “Employee Name”. |
| Ans 3a | SELECT `Employee\_ID`, `First\_Name`, `Last\_Name`, `Salary`, `Joining\_date`, `Department`  FROM `employee`  Where `First\_Name` = 'Tom' |
|  |  |
| Task 3b | Get FIRST\_NAME, Joining Date, and Salary from employee table. |
| Ans 3b | SELECT `Employee\_ID`, `First\_Name`, `Joining\_date`, `Salary` FROM `employee` |
|  |  |
| Task 3c | Get all employee details from the employee table order by First\_Name Ascending and Salary descending? |
| Ans 3c | SELECT `Employee\_ID`, `First\_Name`, `Last\_Name`, `Salary`, `Joining\_date`, `Department`  FROM `employee`  order by `First\_Name` asc, salary DESC |
|  |  |
| Task 3d | Get employee details from employee table whose first name contains ‘J’. |
| Ans 3d | SELECT `Employee\_ID`, `First\_Name`, `Last\_Name`, `Salary`, `Joining\_date`, `Department`  FROM `employee`  WHERE `First\_Name` like '%J%' |
|  |  |
| Task 3e | Get department wise maximum salary from employee table order by salary ascending? |
| Ans 3e | SELECT Department, max(Salary) Max\_Salary  FROM employee  Group by Department  Order by Max\_salary |
|  |  |
| Task 3f | Select first\_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amount greater than 3000 |
| Ans 3f | SELECT `First\_Name`, incentive\_amount FROM `employee`  join incentive on Employee\_ID = Employee\_ref\_ID  where incentive\_amount > 3000 |
|  |  |
| Task 3g | Create After Insert trigger on Employee table which insert records in view table |
| Ans 3g | CREATE TABLE Employee1(  Emp\_ID INT(3) PRIMARY KEY AUTO\_INCREMENT,  First\_Nm VARCHAR(30) NOT NULL,  Last\_Nm VARCHAR(30) NOT NULL,  Sal INT(7) NOT NULL,  Joining\_dt DATETIME NOT NULL,  Dept VARCHAR(20) NOT NULL,  Action\_name VARCHAR(30) NOT NULL  )  DELIMITER $$  create trigger empoyee\_insert after insert on Employee for each ROW  BEGIN  INSERT INTO employee1(Emp\_ID, First\_Nm, Last\_Nm, Sal, Joining\_dt, Dept, Action\_name)  values (new.Employee\_ID, new.First\_Name, new.Last\_Name, new.Salary, new.Joining\_date, new.Department, 'Record Inserted');  END |
|  |  |
| Task 4 | Create table Salesperson and Customer |
| Ans 4 | Create table salesperson  (  sno int(3) primary key,  sname varchar(30) NOT Null,  city varchar(20) NOT Null,  comm decimal NOT Null  ) |