DBMS Solution

Q1)

1. Define the terms
2. Data Abstraction : it is the process of hiding unnecessary or irrelevant details from the end user.
3. Schema : Schema is the overall description of the database. The basic structure of how the data will be stored in the database is called schema.

1. Differentiate between :-

1. Strong Entity set and weak entity Set

|  |  |
| --- | --- |
| Strong Entity | weak entity |
| Strong entity always has a primary key. | While a weak entity has a partial discriminator key. |
| Strong entity is not dependent on any other entity. | Weak entity depends on strong entity |

1. Attribute and Derived Attribute

|  |  |
| --- | --- |
| Attribute | Derived Attribute |
| attribute refers to a database component, such as a table. It also may refer to a database column. Attributes describe the instances in the column of a database. | If an attribute's value can be derived from the values of other attributes, then the attribute is derivable, and is said to be a derived attribute |

1. E-R diagram for student – course Registration system

Course

student

Course registration

Q2 A)

1. Define :-
2. Super key :- It is single key or a group of multiple keys that can uniquely identify attributes in a table.
3. Candidate key :- It is a minimal of super key

Q2 B)

1. Normalization –

Normalization is the process of organizing the data in the database. Normalization is used to minimize the redundancy from a relation or set of relations. Normalization divides the larger table into smaller and links them using relationships. The normal form is used to reduce redundancy from the database table.

SQL Constraints

Constraints are the rules enforced on data columns on a table. These 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 database. Constraints can either be column level or table level. Column level constraints are applied only to one column whereas, table level constraints are applied to the entire table.

Following are some of the most commonly used constraints available in SQL −

• NOT NULL Constraint − Ensures that a column cannot have a NULL value.

• DEFAULT Constraint − Provides a default value for a column when none is specified.

• UNIQUE Constraint − Ensures that all the values in a column are different.

• PRIMARY Key − Uniquely identifies each row/record in a database table.

• FOREIGN Key − Uniquely identifies a row/record in any another database table.

• CHECK Constraint − The CHECK constraint ensures that all values in a column satisfy certain

conditions.

• INDEX − Used to create and retrieve data from the database very quickly.

b)

i)Ans:

CREATE TABLE Student

(

Id\_code INT(4),

sname VARCHAR(20),

Email VARCHAR(40),

Mob\_No VARCHAR(10)

);

ii)Ans:

ALTER TABLE Student ADD date\_of\_birth DATE;

ALTER TABLE Student ADD CONSTRAINT p\_key PRIMARY KEY(Id\_code);

c)Ans:

DML Commands-

* Insert
* Update
* Delete

INSERT INTO Student VALUES (111, 'Ram Desai', 'ramdesai34@gmail.com', '9989974634','2004-09-13')

UPDATE Student SET sname = 'Ram Kale' WHERE Id\_code= 111;

DELETE FROM Student WHERE Id\_code = 111;

SELECT \* FROM Student;

Q3)

1. Aggregate Functions in DBMS

Aggregate functions are those functions in the DBMS which takes the values of multiple rows of a single column and then form a single value by using a query.

In Database Management System, following are the five aggregate functions:

1. AVG

2. COUNT

3. SUM

4. MIN

5. MAX

1.AVG Function

This function takes the values from the given column and then returns the average of the values. This function works only on the datatypes, which are specified as numeric in the table.

Let's take an example, which describes to you how to use the AVG function in SQL. Suppose we want to calculate the average salary from the Employee\_Details table, then we have to type the following query:

Select AVG(Employee\_salary) from Employee\_Details;

2.COUNT Function

This aggregate function returns the total number of values in the specified column. This function can work on any type of data, i.e., numeric as well as non-numeric. This function does not count the NULL values. If we want to count all the rows with NULL values, then we have to use the Count(\*) function.

Let's take an example, which describes to us how to use the COUNT function in SQL. Suppose a user wants to count the number of employees in the Employee\_Details table, then we have to type the following query:

Select Count(Employee\_ID) from Employee\_Details;

3. SUM Function

This aggregate function sums all the non-NULL values of the given column. Like the AVG function, this function also works only on the numeric data.

Let's take an example, which describes to you how to use the SUM function in SQL. Suppose a user wants to find the sum of salary from the Employee\_Details table, then we have to type the following query:

Select SUM(Employee\_salary) from Employee\_Details;

4. MAX Function

This function returns the value, which is maximum from the specified column.

Let's take an example, which describes to you how to use the MAX function in SQL. Suppose we want to find the maximum price of the Cars\_Price column from the Cars table, then we have to type the following query:

Select MAX(Cars\_Price) from Cars;

5. MIN Function

This function returns the value, which is minimum from the specified column.

Let's take an example, which describes to you how to use the MIN function in SQL. Suppose we want to find the minimum price of the Bikes\_Price column from the Bikes table, then we have to type the following query:

Select MIN(Bikes\_Price) from Bikes;

SYNONYM:

A SYNONYM provides another name for database object, referred to as original object, that may exist on a local or another server. A synonym belongs to schema, name of synonym should be unique. A synonym cannot be original object for an additional synonym and synonym cannot refer to user-defined function.

Syntax –

CREATE SYNONYM synonym\_name for table\_name

Drop Synonym synonym\_name

Example:

CREATE SYNONYM d FOR department;

i)SELECT emp\_id from Employee where Joining \_date< 02-JAN-2013;

ii)SELECT \* FROM Employee order by First\_name asc;

iii) SELECT AVG(Salary), Department from Employee group by Department

iv) CREATE SEQUENCE id\_seq

STARTS WITH 1

INCREMENT BY 1

NO CACHE

NO CYCLE;

OR

CREATE TABLE Employee

( Employee\_id INT UNSIGNED NOT NULL AUTO\_INCREMENT, Primary Key()

--------------

----------------

);