(1) A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.

(2) What is Normalization?

Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

The inventor of the relational model Edgar Codd proposed the theory of normalization of data with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined Raymond F. Boyce to develop the theory of Boyce-Codd Normal Form.

(3)What is Difference between DBMS and RDBMS?

A database is a collection of organised or arranged data that can be easily accessed, updated/ modified or controlled. Information within the database can be easily placed into rows and columns, or tables. Meanwhile, database management enables the user to store, manage and access data. Knowing the definitions and the difference between relational database management system and database management system will help candidates to prepare well.

(4) What is MF Cod Rule of RDBMS Systems?

Rules

Rule 0: The foundation rule:

For any system that is advertised as, or claimed to be, a relational data base management system, that system must be able to manage data bases entirely through its relational capabilities.

Rule 1: The information rule:

All information in a relational data base is represented explicitly at the logical level and in exactly one way – by values in tables.

Rule 2: The guaranteed access rule:

Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.

Rule 3: Systematic treatment of null values:

Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type.

Rule 4: Dynamic online catalog based on the relational model:

The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.

Rule 5: The comprehensive data sublanguage rule:

A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all of the following items:

Data definition.

View definition.

Data manipulation (interactive and by program).

Integrity constraints.

Authorization.

Transaction boundaries (begin, commit and rollback).

Rule 6: The view updating rule:

All views that are theoretically updatable are also updatable by the system.

Rule 7: Relational Operations Rule / Possible for high-level insert, update, and delete:

The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.

Rule 8: Physical data independence:

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.

Rule 9: Logical data independence:

Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

Rule 10: Integrity independence:

Integrity constraints specific to a particular relational data base must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

Rule 11: Distribution independence:

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only.

Rule 12: The nonsubversion rule:

If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a-time).

(5)What do you understand By Data Redundancy?

Sometimes data redundancy happens by accident while other times it is intentional. Accidental data redundancy can be the result of a complex process or inefficient coding while intentional data redundancy can be used to protect data and ensure consistency — simply by leveraging the multiple occurrences of data for disaster recovery and quality checks.

If data redundancy is intentional, it’s important to have a central field or space for the data. This allows you to easily update all records of redundant data when necessary. When data redundancy isn’t purposeful, it can lead to a variety of issues which we’ll discuss below.

Understanding database versus file-based data redundancy

Data redundancy can be found in a database, which is an organized collection of structured data that’s stored by a computer system or the cloud. A retailer may have a database to track the products they stock. If the same product gets entered twice by mistake, data redundancy takes place.

The same retailer may keep customer files in a file storage system. If a customer purchases from the company more than once, their name may be entered multiple times. Duplicate entries of the customer name is considered redundant data.

Regardless of whether data redundancy occurs in a database or in a file storage system, it can be problematic. Fortunately, data replication can help prevent data redundancy by storing the same data in multiple locations. With data replication, companies can ensure consistency and receive the information they need at any time.

(6) What is DDL Interpreter?

It interprets DDL statements and record them in tables containing metadata.

Its a language in database through which you can make the logical design of the schemas ....

Submitted by: Administrator

(7) What is DML Compiler in SQL?

DML Compiler: Translates DML statements in a query language within low level instructions understandable through the query evaluation engine. Attempts to transforms users request within an equivalent and well-organized from for executing the query understandable through Data Manager, Interprets DDL statements and records them within a set of tables containing Meta data in a form that can be used through other elements of a DBMS.

(8) What is SQL Key Constraints writing an Example of SQL Key Constraints

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 - 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 - Prevents actions that would destroy links between tables

CHECK - Ensures that the values in a column satisfies a specific condition

DEFAULT - Sets a default value for a column if no value is specified

CREATE INDEX - Used to create and retrieve data from the database very quickly

(9) What is save Point? How to create a save Point write a Query?

Savepoint 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 savepoint.

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.

Savepoint 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 savepoint is a bookmark in SQL.

Let us see the practical examples to understand this concept more clearly. We will use the MySQL database for writing all the queries.

To create a table in the database, first, we need to select the database in which we want to create a table.

(10) What is trigger and how to create a Trigger in SQL?

In this chapter, we will discuss Triggers in PL/SQL. Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events −

A database manipulation (DML) statement (DELETE, INSERT, or UPDATE)

A database definition (DDL) statement (CREATE, ALTER, or DROP).

A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers

Triggers can be written for the following purposes −

Generating some derived column values automatically

Enforcing referential integrity

Event logging and storing information on table access

Auditing

Synchronous replication of tables

Imposing security authorizations

Preventing invalid transactions

Creating Triggers

The syntax for creating a trigger is −

CREATE [OR REPLACE ] TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col\_name]

ON table\_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE

Declaration-statements

BEGIN

Executable-statements

EXCEPTION

Exception-handling-statements

END;

Where,

CREATE [OR REPLACE] TRIGGER trigger\_name − Creates or replaces an existing trigger with the trigger\_name.

{BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.

{INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.

[OF col\_name] − This specifies the column name that will be updated.

[ON table\_name] − This specifies the name of the table associated with the trigger.

[REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.

[FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.

WHEN (condition) − This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.