What is SQL?

varshachoudhary

Structured Query Language is a computer language that we use to interact with a relational database. SQL is a tool for organizing, managing, and retrieving archived data from a computer database. The original name was given by IBM as Structured English Query Language, abbreviated by the acronym SEQUEL. When data needs to be retrieved from a database, SQL is used to make the request. The DBMS processes the SQL query retrieves the requested data and returns it to us. Rather, SQL statements describe how a collection of data should be organized or what data should be extracted or added to the database.

In common usage, SQL encompasses DDL and DML commands for CREATE, UPDATE, modified, or other operations on database structure.

Features of SQL

- SQL may be utilized by quite a number of users, which include people with very little programming experience.
- SQL is a non-procedural language.
- We can without difficulty create and replace databases in SQL. It isn't a time-consuming process.
- SQL is primarily based totally on ANSI standards.
- SQL does now no longer have a continuation individual.
- SQL is entered into the SQL buffer on one or extra lines.
- SQL makes use of a termination individual to execute instructions immediately. It makes use of features to carry out a few formatting.
- It uses functions to perform some formatting.

SQL Uses

1. **Data definition:** It is used to define the structure and organization of the stored data and the relationships among the stored data items.
2. **Data retrieval:** SQL can also be used for data retrieval.
3. **Data manipulation:** If the user wants to add new data, remove data, or modifying in existing data then SQL provides this facility also.
4. **Access control:** SQL can be used to restrict a user's ability to retrieve, add, and modify data, protecting stored data against unauthorized access.

5. **Data sharing:** SQL is used to coordinate data sharing by concurrent users, ensuring that changes made by one user do not inadvertently wipe out changes made at nearly the same time by another user.

SQL also differs from other computer languages because it describes what the user wants the computer to do rather than how the computer should do it. (In more technical terms, SQL is a declarative or descriptive language rather than a procedural one.) SQL contains no IF statement for testing conditions, and no GOTO, DO, or FOR statements for program flow control. Rather, SQL statements describe how a collection of data is to be organized, or what data is to be retrieved or added to the database. The sequence of steps to do those tasks is left for the DBMS to determine.

Rules for SQL

- A ';' is used to end SQL statements.
- Statements may be split across lines, but keywords may not.
- Identifiers, operator names, and literals are separated by one or more spaces or other delimiters.
- A comma (,) separates parameters without a clause.
- A space separates a clause.
- Reserved words cannot be used as identifiers unless enclosed with double quotes.
- Identifiers can contain up to 30 characters.
- Identifiers must start with an alphabetic character.
- Characters and date literals must be enclosed within single quotes.
- Numeric literals can be represented by simple values.
- Comments may be enclosed between /* and */ symbols and maybe multi-line.

How does SQL Function?

A server machine is used in the implementation of the structured query language (SQL), processing database queries and returning results. The following are some of the software elements that the SQL process goes through.

AD

1. **Parser:** The parser begins by replacing some of the words in the SQL statement with unique symbols, a process known as tokenization. The statement is then examined for the following:

2. **Correctness:** The parser checks to see if the SQL statement complies with the rules, or SQL semantics, that guarantee the query statement's accuracy. The parser, for instance, looks to see if the SQL command ends with a semicolon. The parser returns an error if the semi-colon is absent.
3. **Authorization:** The parser additionally confirms that the user executing the query has the required permissions to alter the relevant data.
4. **Relational Engine:** The relational engine, also known as the query processor, develops a strategy for efficiently retrieving, writing, or updating the relevant data. For instance, it looks for queries that are similar to others, uses earlier data manipulation techniques, or develops a new one. Byte code, an intermediate-level representation of the SQL statement, is used to write the plan. To efficiently perform database searches and modifications, relational databases use byte code.
5. **Storage Engine:** The software element that interprets the byte code and executes the intended SQL statement is known as the storage engine, also known as the database engine. The data in the database files on the physical disc storage is read and stored. The storage engine delivers the outcome to the requesting application after completion.

Role of SQL

SQL plays many different roles:

- SQL is an interactive question language. Users type SQL instructions into an interactive SQL software to retrieve facts and show them on the screen, presenting a convenient, easy-to-use device for ad hoc database queries.
- SQL is a database programming language. Programmers embed SQL instructions into their utility packages to access the facts in a database. Both user-written packages and database software packages (consisting of document writers and facts access tools) use this approach for database access.
- SQL is a client/server language. Personal computer programs use SQL to communicate over a network with database servers that save shared facts. This client/server architecture is utilized by many famous enterprise-class applications.
- SQL is Internet facts access language. Internet net servers that interact with company facts and Internet utility servers all use SQL as a widespread language for getting access to company databases, frequently through embedding SQL databases get entry to inside famous scripting languages like PHP or Perl.
- SQL is a distributed database language. Distributed database control structures use SQL to assist distribute facts throughout many linked pc structures. The DBMS software program on every gadget makes use of SQL to speak with the opposite structures, sending requests for facts to get entry to.
- SQL is a database gateway language. In a pc community with a mixture of various DBMS products, SQL is frequently utilized in a gateway that lets one logo of DBMS speak with every other logo. SQL has for this reason emerged as a useful, effective device for linking people, pc packages, and pc structures to the facts saved in a relational database.

SQL Injection

A cyberattack known as SQL injection involves tricking the database with SQL queries. To retrieve, alter, or corrupt data in a SQL database, hackers use SQL injection. To execute a SQL injection attack, for instance, they might enter a SQL query in place of a person's name in a submission form.

What is SQL Server?

Microsoft's relational database management system, which uses SQL to manipulate data, is formally known as SQL Server. There are various editions of the MS SQL Server, and each is tailored for particular workloads and requirements.

Finally, SQL is not a particularly structured language, especially when compared with highly structured languages such as C, Pascal, or Java. Instead, SQL statements resemble English sentences, complete with "noise words" that don't add to the meaning of the statement but make it read more naturally. SQL has quite a few inconsistencies and also some special rules to prevent you from constructing SQL statements that look perfectly legal but that don't make sense.

Despite the inaccuracy of its name, SQL has emerged as the standard language for using relational databases. SQL is both a powerful language and one that is relatively easy to learn. So, SQL is a database management language. The database administrator is answerable for handling a minicomputer or mainframe database and makes use of SQL to outline the database shape and manipulate get entry to the saved data.

How do SQL Commands Work?

Developers use structured query language (SQL) commands, which are specific keywords or SQL statements, to work with data stored in relational databases. The following are categories for SQL commands.

Data Definition Language

SQL commands used to create the database structure are known as data definition language (DDL). Based on the needs of the business, database engineers create and modify database objects using DDL. The CREATE command, for instance, is used by the database engineer to create database objects like tables, views, and indexes.

Data Query Language

Data retrieval instructions are written in the data query language (DQL), which is used to access relational databases. The SELECT command is used by software programs to filter and return particular results from a SQL table.

Data Manipulation Language

A relational database can be updated with new data using data manipulation language (DML) statements. The INSERT command, for instance, is used by an application to add a new record to the database.

Data Control language

Data control language (DCL) is a programming language used by database administrators to control or grant other users access to databases. For instance, they can allow specific applications to manipulate one or more tables by using the GRANT command.

Transaction Control Language

To automatically update databases, the relational engine uses transaction control language (TCL). For instance, the database can reverse a mistaken transaction using the ROLLBACK command.

Last Updated : 10 May, 2023

14

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. SQL | Procedures in PL/SQL
5. SQL | Difference between functions and stored procedures in PL/SQL
6. SQL SERVER – Input and Output Parameter For Dynamic SQL
7. Difference between T-SQL and PL-SQL
8. Difference between SQL and T-SQL
9. SQL Server | Convert tables in T-SQL into XML
10. SQL SERVER | Bulk insert data from csv file using T-SQL command

Previous Next

Article Contributed By :

varshachoudhary
varshachoudhary

Vote for difficulty

SQL Data Types

sakshijain1

The data type of a column can be defined as basically what type of data format in which each cell will store the data – it can be any type of integer, character, money, date and time, binary, etc. In this article, we'll get to learn about SQL Data Types in detail.

SQL Data Types

An SQL developer must aware of what type of data will be stored inside each column while creating a table. The data type guideline for SQL is to understand what type of data is expected inside each column and it also identifies how SQL will interact with the stored data.

SQL (Structured Query Language) is a language used to interact with relational databases. SQL data types define the type of data that can be stored in a database column or variable. Here are the most common SQL data types:

Data Type	Properties
Numeric data types	These are used to store numeric values. Examples include INT, BIGINT, DECIMAL, and FLOAT.
Character data types	These are used to store character strings. Examples include CHAR, VARCHAR, and TEXT.
Date and time data types	These are used to store date and time values. Examples include DATE, TIME, and TIMESTAMP.
Binary data types	These are used to store binary data, such as images or audio files. Examples include BLOB and BYTEA.
Boolean data type	This data type is used to store logical values. The only possible values are TRUE and FALSE.
Interval data types	These are used to store intervals of time. Examples include INTERVAL YEAR, INTERVAL MONTH, and INTERVAL DAY.

Data Type	Properties
Array data types	These are used to store arrays of values. Examples include ARRAY and JSON.
XML data type	This data type is used to store XML data.
Spatial data types	These are used to store geometric or geographic data. Examples include POINT, LINE, and POLYGON.

Different databases may have different variations of these data types, or they may have additional data types not listed here. Understanding SQL data types are important for creating tables and working with data in a database, as it affects how data is stored and processed.

AD

Like in other programming languages, SQL also has certain datatypes available. A brief idea of all the datatypes is discussed below.

Binary Datatypes

There are four subtypes of this datatype which are given below:

Data Type	Description
Binary	The maximum length of 8000 bytes(Fixed-Length binary data)
varbinary	The maximum length of 8000 bytes(Variable Length binary data)
varbinary(max)	The maximum length of 231 bytes(SQL Server 2005 only).(Variable Length binary data)
image	Maximum Length of 2,147,483,647 bytes(Variable Length binary data)

Exact Numeric Datatype

There are nine subtypes which are given below in the table. The table contains the range of data in a particular type.

Data Type	From	To
BigInt	-2^63 (-9,223,372,036,854,775,808)	2^63-1 (9,223,372,036,854,775,807)
Int	-2^31 (-2,147,483,648)	2^31-1 (2,147,483,647)
smallint	-2^15 (-32,768)	2^15-1 (32,767)
tinyint	0	2^8-1
bit	0	1
decimal	-10^38+1	10^38+1
numeric	-10^38+1	10^38+1
money	-922,337,203,685,477,5808	922,337,203,685,477,5808
smallmoney	-2,147,748	2,147,748

Approximate Numeric Datatype

The subtypes of this datatype are given in the table with the range.

Data Type	From	To
Float	-1.79E+308	1.79E+308
Real	-3.40E+38	3.40E+38

Character String Datatype

The subtypes are given in below table –

Data Type	Description
char	The maximum length of 8000 characters.(Fixed-Length non-Unicode Characters)

varchar	The maximum length of 8000 characters.(Variable-Length non-Unicode Characters)
varchar(max)	The maximum length of 231 characters(SQL Server 2005 only).(Variable Length non-Unicode data)
text	The maximum length of 2,127,483,647 characters(Variable Length non-Unicode data)

Unicode Character String Datatype

The details are given in below table –

Data Type	Description
nchar	The maximum length of 4000 characters(Fixed-Length Unicode Characters)
Nvarchar	The maximum length of 4000 characters.(Variable-Length Unicode Characters)
nvarchar(max)	The maximum length of 231 characters(SQL Server 2005 only).(Variable Length Unicode data)

Date and Time Datatype

The details are given in the below table.

Data Type	Description
DATE	A data type is used to store the data of date in a record
TIME	A data type is used to store the data of time in a record
DATETIME	A data type is used to store both the data,date, and time in the record.

XML Datatype

XML data type allows storage of XML documents and fragments in a SQL Server database

DataType	Description
XML Datatype	A Datatype used to store data in the format of XML datatype

Spatial Dataype

A datatype is used for storing planar spatial data, such as points, lines, and polygons, in a database table.

DataType	Description
Geometry	A datatype is used for storing planar spatial data, such as points, lines, and polygons, in a database table.

Array Datatype

SQL Server does not have a built-in array datatype. However, it is possible to simulate arrays using tables or XML data types.

Last Updated : 12 Apr, 2023

156

Similar Reads

1. How to Convert Data From SQL to C Data Types?
2. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
3. Configure SQL Jobs in SQL Server using T-SQL
4. SQL vs NO SQL vs NEW SQL
5. SQL SERVER | Bulk insert data from csv file using T-SQL command
6. Numeric and Date-time data types in SQL Server
7. SQL | Procedures in PL/SQL
8. SQL | Difference between functions and stored procedures in PL/SQL
9. SQL SERVER – Input and Output Parameter For Dynamic SQL
10. Difference between SQL and T-SQL

Previous Next

Article Contributed By :

SQL | DDL, DQL, DML, DCL and TCL Commands

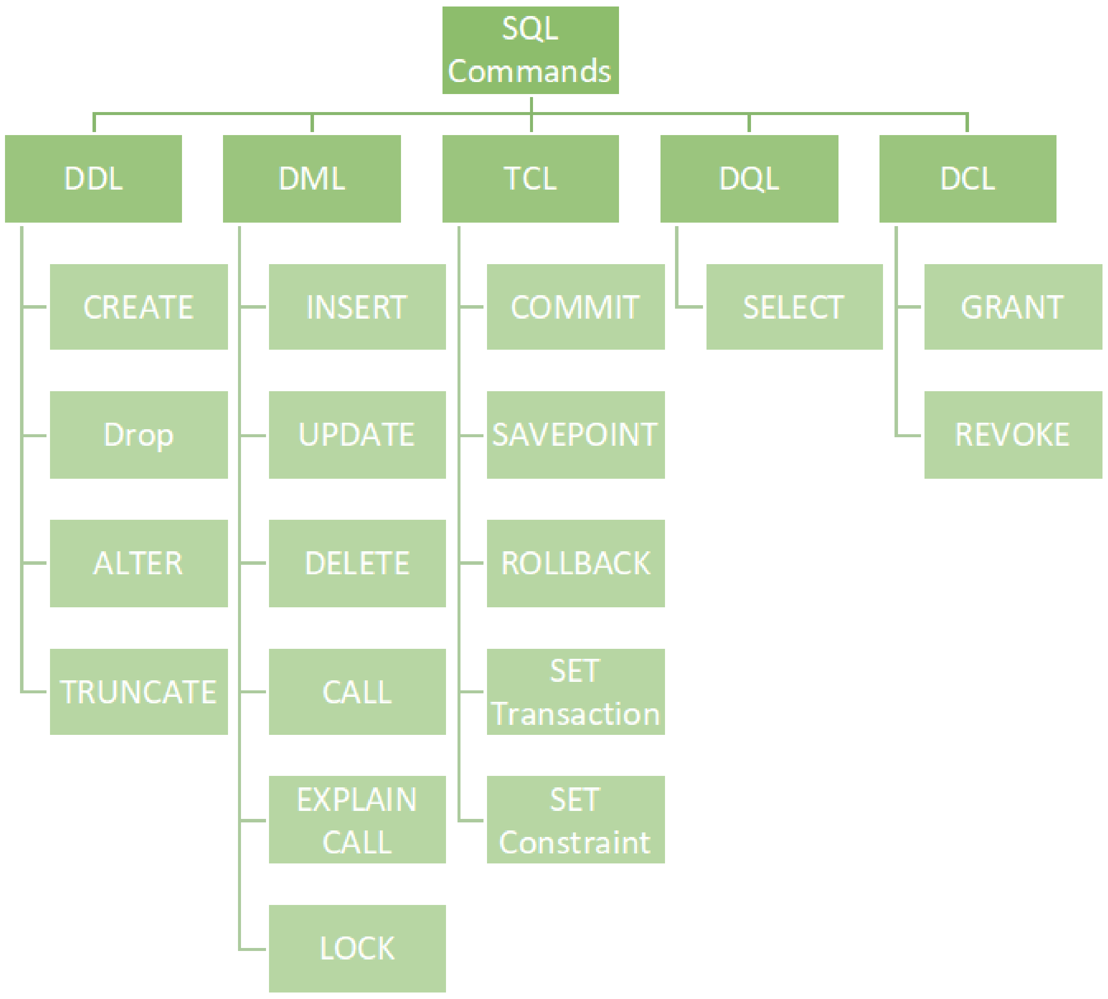
Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like CREATE, DROP, INSERT, etc. to carry out the required tasks.

SQL commands are like instructions to a table. It is used to interact with the database with some operations. It is also used to perform specific tasks, functions, and queries of data. SQL can perform various tasks like creating a table, adding data to tables, dropping the table, modifying the table, set permission for users.

These SQL commands are mainly categorized into five categories:

1. DDL – Data Definition Language
2. DQL – Data Query Language
3. DML – Data Manipulation Language
4. DCL – Data Control Language
5. TCL – Transaction Control Language

Now, we will see all of these in detail.



DDL (Data Definition Language)

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

List of DDL commands:

- CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
- DROP: This command is used to delete objects from the database.
- ALTER**:** This is used to alter the structure of the database.
- TRUNCATE**:** This is used to remove all records from a table, including all spaces allocated for the records are removed.
- COMMENT: This is used to add comments to the data dictionary.

- **RENAME:** This is used to rename an object existing in the database.

DQL (Data Query Language)

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.

List of DQL:

- **SELECT:** It is used to retrieve data from the database.

DML(Data Manipulation Language)

The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

List of DML commands:

- **INSERT**: It is used to insert data into a table.
- **UPDATE:** It is used to update existing data within a table.
- **DELETE**: It is used to delete records from a database table.
- **LOCK:** Table control concurrency.
- **CALL:** Call a PL/SQL or JAVA subprogram.
- **EXPLAIN PLAN:** It describes the access path to data.

DCL (Data Control Language)

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

List of DCL commands:

GRANT: This command gives users access privileges to the database.

Syntax:

GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

REVOKE: This command withdraws the user's access privileges given by using the GRANT command.

Syntax:

REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

TCL (Transaction Control Language)

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure. You can explore more about transactions ***here***. Hence, the following TCL commands are used to control the execution of a transaction:

BEGIN: Opens a Transaction.

COMMIT**:** Commits a Transaction.

Syntax:

COMMIT;

ROLLBACK**:** Rollbacks a transaction in case of any error occurs.

Syntax:

ROLLBACK;

SAVEPOINT**:** Sets a save point within a transaction.

Syntax:

SAVEPOINT SAVEPOINT_NAME;

Last Updated : 10 May, 2023 700

Similar Reads

1. SQL | DDL, DML, TCL and DCL

SQL | Comments

Comments are used to explain sections of SQL statements or to prevent SQL statements from being executed. As is any programming language comments matter a lot in SQL also. In this set, we will learn about writing comments in any SQL snippet.

Comments can be written in the following three formats:

1. Single-line comments
2. Multi-line comments
3. In-line comments

Single Line Comments

Comments starting and ending in a single line are considered single-line comments. A line starting with '–' is a comment and will not be executed.

Syntax:

— single line comment

— another comment

*SELECT * FROM Customers;*

Let's see an example in which we have one table name Customers in which we are fetching all the data records.

Query:

```
SELECT * FROM customers;
-- This is a comment that explains
   the purpose of the query.
```

Multi-Line Comments

Comments starting in one line and ending in different lines are considered as multi-line comments.

A line starting with '/*' is considered as starting point of the comment and is terminated when '*/' is encountered.

Syntax:

/ multi line comment*

*another comment */*

*SELECT * FROM Customers;*

Let's consider that we want to fetch order_date with the year 2022.

Query:

```
/*
This is a multi-line comment that explains
the purpose of the query below.
The query selects all the orders from the orders
table that were placed in the year 2022.
*/
SELECT * FROM orders WHERE YEAR(order_date) = 2022;
```

In-Line Comments

In-line comments are an extension of multi-line comments, comments can be stated in between the statements and are enclosed in between '/*' and '*/'.

Syntax:

*SELECT * FROM /* Customers; */*

Let's Suppose we have on a table with name orders and we are fetching customer_name.

Query:

```
SELECT customer_name,
/* This column contains the name of
the customer / order_date /
This column contains the date the
order was placed */ FROM orders;
```

More Examples:

There are a few more examples of Multiline comments and inline comments.

```
Multi line comment ->
/* SELECT * FROM Students;
SELECT * FROM STUDENT_DETAILS;
SELECT * FROM Orders; */
SELECT * FROM Articles;

In line comment ->
SELECT * FROM Students;
SELECT * FROM /* STUDENT_DETAILS;
SELECT * FROM Orders;
SELECT * FROM */ Articles;
```

This article is contributed by **Pratik Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 18 Apr, 2023

62

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. Django project to create a Comments System
5. SQL | Procedures in PL/SQL
6. SQL | Difference between functions and stored procedures in PL/SQL

SQL | TRANSACTIONS

What are Transactions?

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: **success** or **failure**.
Example of a transaction to transfer $150 from account A to account B:
1. read(A)
2. A := A – 150
3. write(A)
4. read(B)
5. B := B + 150
6. write(B)

Incomplete steps result in the failure of the transaction. A database transaction, by definition, must be atomic, consistent, isolated and durable.
These are popularly known as ACID properties. These properties can ensure the concurrent execution of multiple transactions without conflict.

How to implement Transactions using SQL?

AD

Following commands are used to control transactions. It is important to note that these statements cannot be used while creating tables and are only used with the DML Commands such as – INSERT, UPDATE and DELETE.

1. BEGIN TRANSACTION: It indicates the start point of an explicit or local transaction.

Syntax:

```
BEGIN TRANSACTION transaction_name ;
```

2. SET TRANSACTION: Places a name on a transaction.

Syntax:

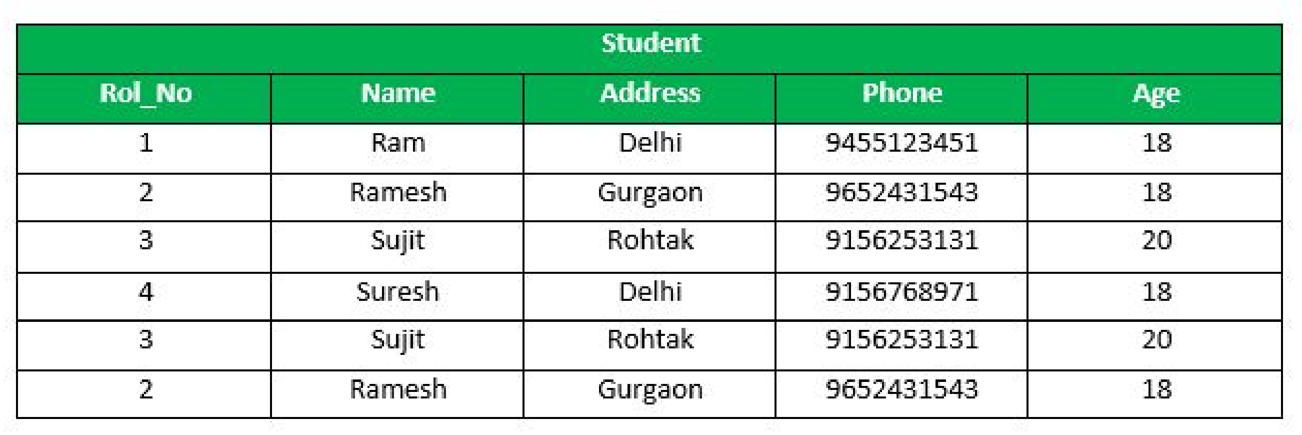
```
SET TRANSACTION [ READ WRITE | READ ONLY ];
```

3. COMMIT: If everything is in order with all statements within a single transaction, all changes are recorded together in the database is called **committed**. The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.

Syntax:

```
COMMIT;
```

Example: Sample table 1

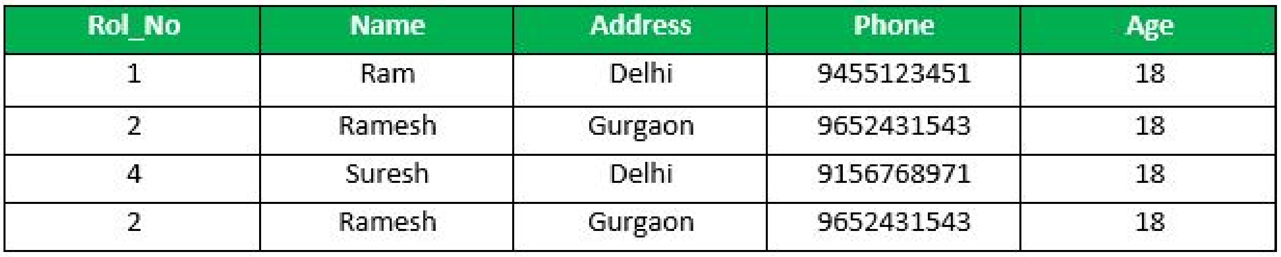


Student				
Rol_No	Name	Address	Phone	Age
1	Ram	Delhi	9455123451	18
2	Ramesh	Gurgaon	9652431543	18
3	Sujit	Rohtak	9156253131	20
4	Suresh	Delhi	9156768971	18
3	Sujit	Rohtak	9156253131	20
2	Ramesh	Gurgaon	9652431543	18

Following is an example which would delete those records from the table which have age = 20 and then COMMIT the changes in the database.
Queries:

```
DELETE FROM Student WHERE AGE = 20;
COMMIT;
```

Output:
Thus, two rows from the table would be deleted and the SELECT statement would look like,



Rol_No	Name	Address	Phone	Age
1	Ram	Delhi	9455123451	18
2	Ramesh	Gurgaon	9652431543	18
4	Suresh	Delhi	9156768971	18
2	Ramesh	Gurgaon	9652431543	18

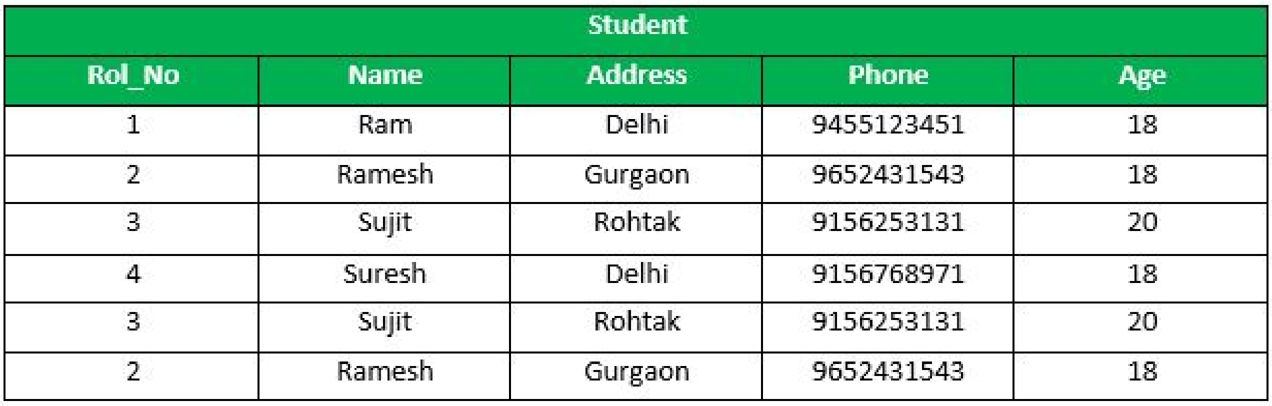
4. ROLLBACK: If any error occurs with any of the SQL grouped statements, all changes need to be aborted. The process of reversing changes is called **rollback**. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.
Syntax:

```
ROLLBACK;
```

Example:
From the above example **Sample table1**,
Delete those records from the table which have age = 20 and then ROLLBACK the changes in the database.
Queries:

```
DELETE FROM Student WHERE AGE = 20;
ROLLBACK;
```

Output:



Student				
Rol_No	Name	Address	Phone	Age
1	Ram	Delhi	9455123451	18
2	Ramesh	Gurgaon	9652431543	18
3	Sujit	Rohtak	9156253131	20
4	Suresh	Delhi	9156768971	18
3	Sujit	Rohtak	9156253131	20
2	Ramesh	Gurgaon	9652431543	18

5. SAVEPOINT: creates points within the groups of transactions in which to ROLLBACK.
A SAVEPOINT is a point in a transaction in which you can roll the transaction back to a certain point without rolling back the entire transaction.

Syntax for Savepoint command:

```
SAVEPOINT SAVEPOINT_NAME;
```

This command is used only in the creation of SAVEPOINT among all the transactions.
In general ROLLBACK is used to undo a group of transactions.
Syntax for rolling back to Savepoint command:

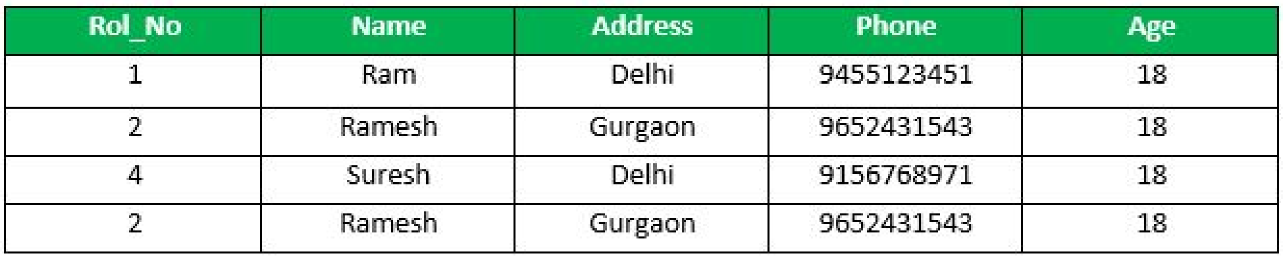
```
ROLLBACK TO SAVEPOINT_NAME;
```

you can ROLLBACK to any SAVEPOINT at any time to return the appropriate data to its original state.
Example:
From the above example **Sample table1**,
Delete those records from the table which have age = 20 and then ROLLBACK the changes in the database by keeping Savepoints.
Queries:

```
SAVEPOINT SP1;
//Savepoint created.
DELETE FROM Student WHERE AGE = 20;
//deleted
```

```
SAVEPOINT SP2;
//Savepoint created.
```

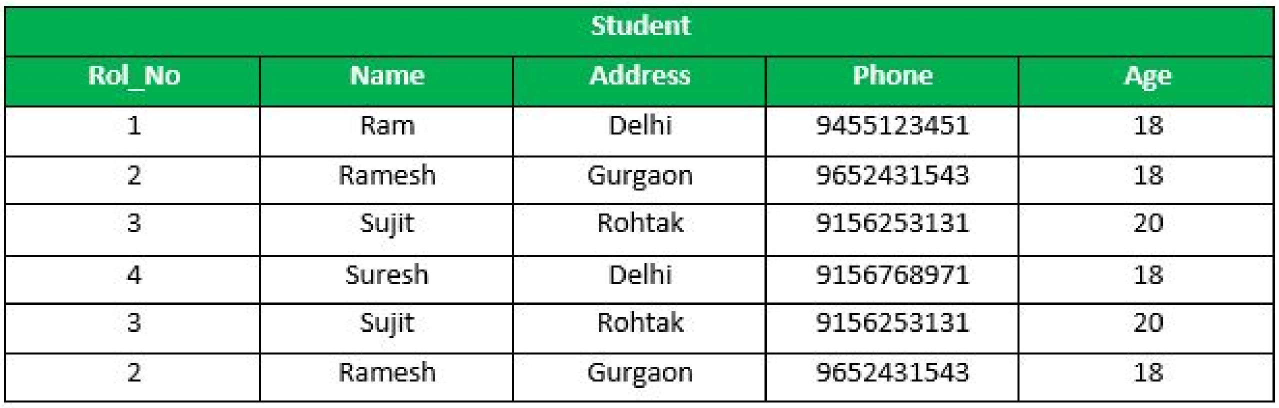
Here SP1 is first SAVEPOINT created before deletion.In this example one deletion have taken place.
After deletion again SAVEPOINT SP2 is created.
Output:



Rol_No	Name	Address	Phone	Age
1	Ram	Delhi	9455123451	18
2	Ramesh	Gurgaon	9652431543	18
4	Suresh	Delhi	9156768971	18
2	Ramesh	Gurgaon	9652431543	18

Deletion have been taken place, let us assume that you have changed your mind and decided to ROLLBACK to the SAVEPOINT that you identified as SP1 which is before deletion.
deletion is undone by this statement ,

```
ROLLBACK TO SP1;
//Rollback completed.
```



Student				
Rol_No	Name	Address	Phone	Age
1	Ram	Delhi	9455123451	18
2	Ramesh	Gurgaon	9652431543	18
3	Sujit	Rohtak	9156253131	20
4	Suresh	Delhi	9156768971	18
3	Sujit	Rohtak	9156253131	20
2	Ramesh	Gurgaon	9652431543	18

6. RELEASE SAVEPOINT:- This command is used to remove a SAVEPOINT that you have created.

Syntax:

```
RELEASE SAVEPOINT SAVEPOINT_NAME
```

Once a SAVEPOINT has been released, you can no longer use the ROLLBACK command to undo transactions performed since the last SAVEPOINT.

It is used to initiate a database transaction and used to specify characteristics of the transaction that follows.

Last Updated : 25 Oct, 2022 165

Similar Reads

1. PL/SQL Transactions
2. Auto-commit commands and Compensating transactions in SQL
3. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
4. Configure SQL Jobs in SQL Server using T-SQL
5. SQL vs NO SQL vs NEW SQL
6. SQL | Procedures in PL/SQL
7. SQL | Difference between functions and stored procedures in PL/SQL
8. SQL SERVER – Input and Output Parameter For Dynamic SQL
9. Difference between SQL and T-SQL
10. SQL Server | Convert tables in T-SQL into XML

Previous Next

Article Contributed By :

GeeksforGeeks

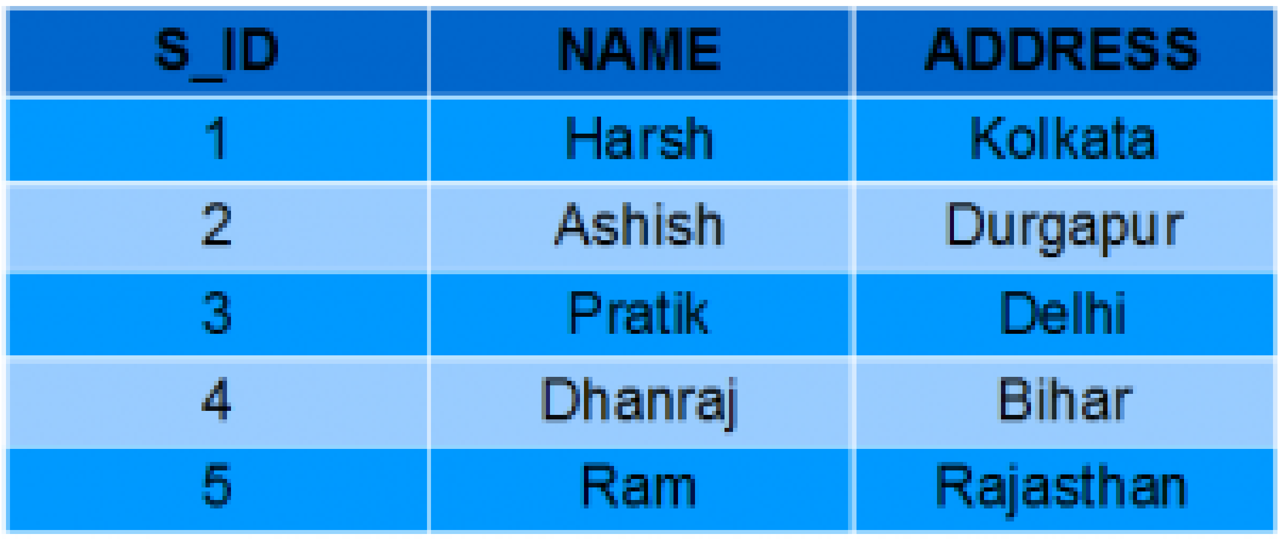
Vote for difficulty

SQL | Views

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition. In this article we will learn about creating , deleting and updating Views.

Sample Tables:

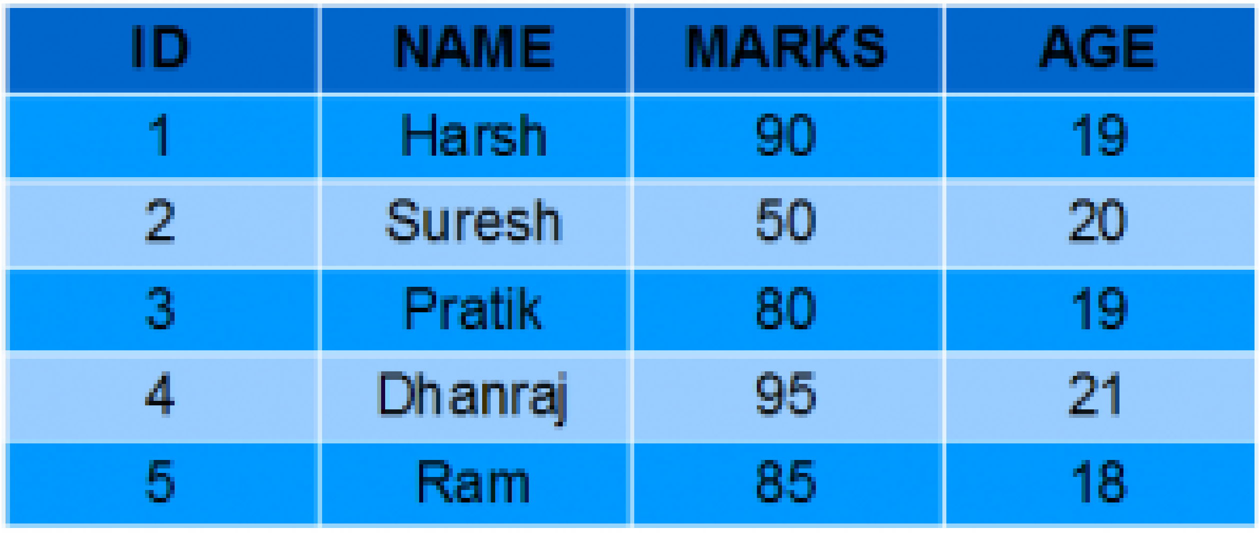
StudentDetails



S_ID	NAME	ADDRESS
1	Harsh	Kolkata
2	Ashish	Durgapur
3	Pratik	Delhi
4	Dhanraj	Bihar
5	Ram	Rajasthan

StudentMarks

AD



ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

CREATING VIEWS

We can create View using **CREATE VIEW** statement. A View can be created from a single table or multiple tables. **Syntax**:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;

view_name: Name for the View
table_name: Name of the table
condition: Condition to select rows
```

Examples:

Creating View from a single table:

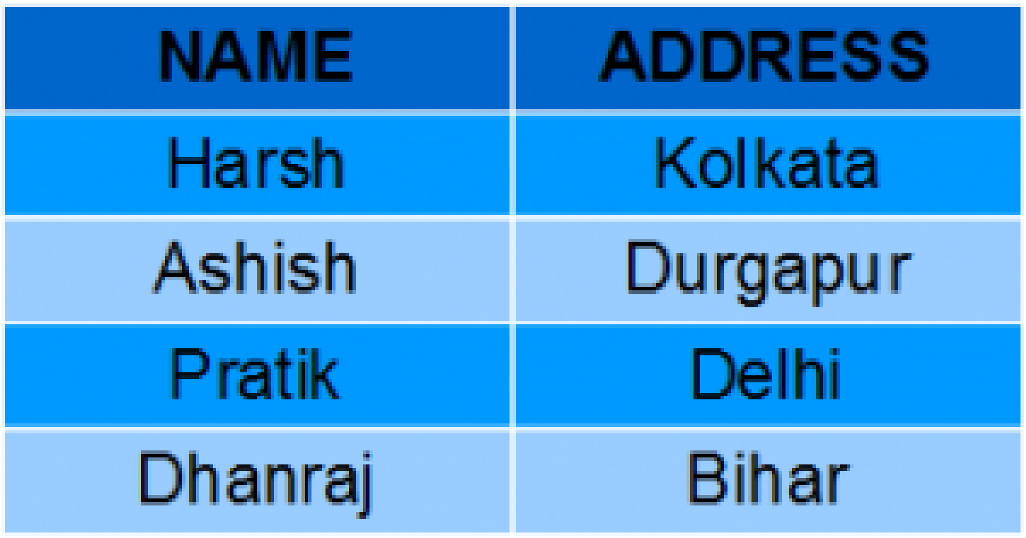
- In this example we will create a View named DetailsView from the table StudentDetails. Query:

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM StudentDetails
WHERE S_ID < 5;
```

- To see the data in the View, we can query the view in the same manner as we query a table.

```
SELECT * FROM DetailsView;
```

Output:



NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

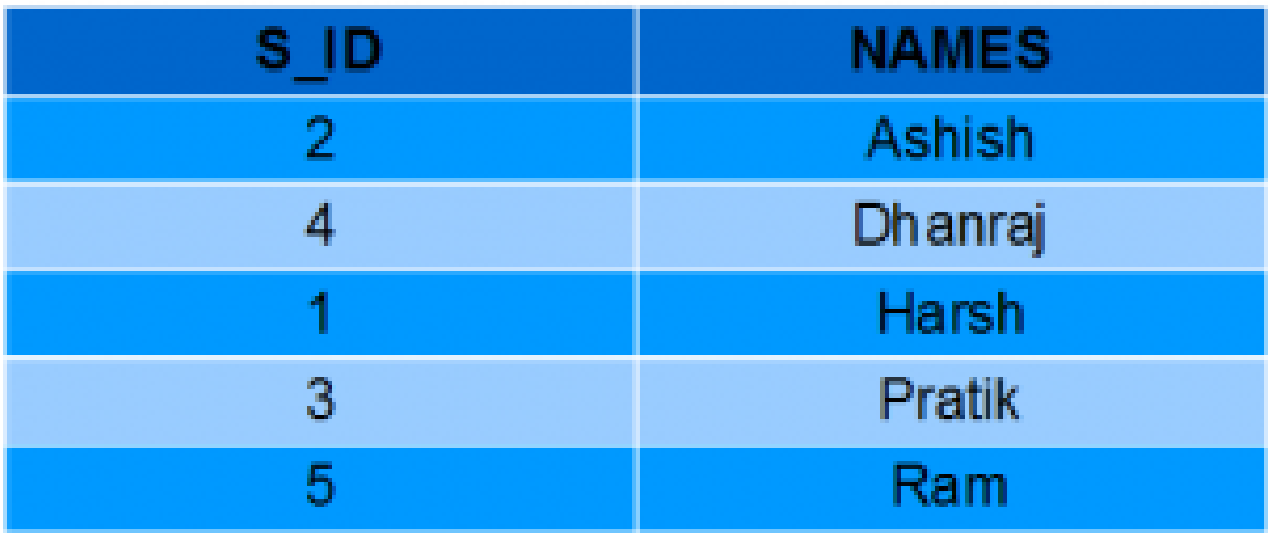
- In this example, we will create a view named StudentNames from the table StudentDetails. Query:

```
CREATE VIEW StudentNames AS
SELECT S_ID, NAME
FROM StudentDetails
ORDER BY NAME;
```

- If we now query the view as,

```
SELECT * FROM StudentNames;
```

Output:



S_ID	NAMES
2	Ashish
4	Dhanraj
1	Harsh
3	Pratik
5	Ram

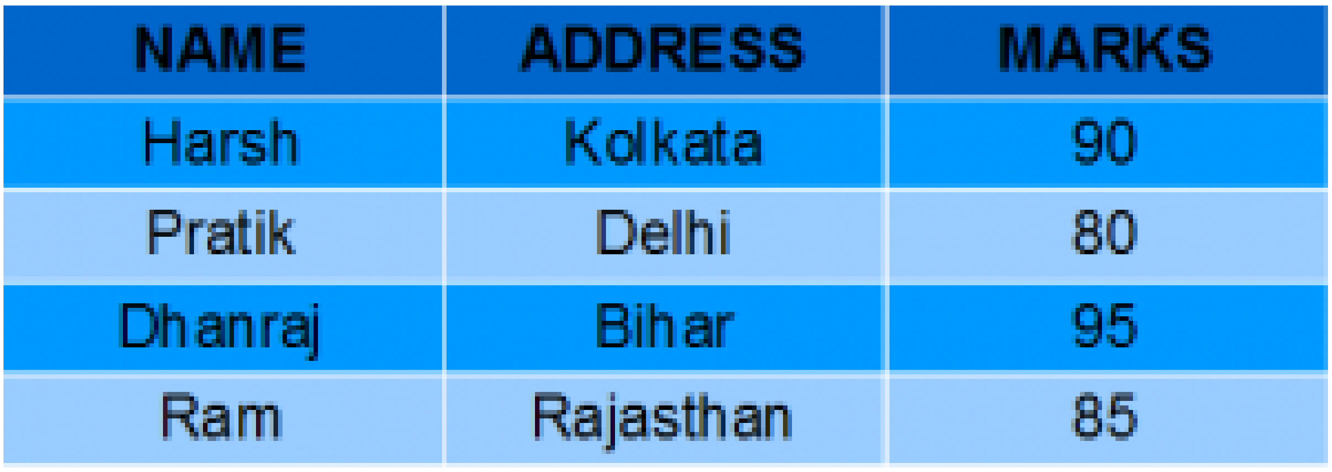
- **Creating View from multiple tables**: In this example we will create a View named MarksView from two tables StudentDetails and StudentMarks. To create a View from multiple tables we can simply include multiple tables in the SELECT statement. Query:

```
CREATE VIEW MarksView AS
SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

- To display data of View MarksView:

```
SELECT * FROM MarksView;
```

- Output:



NAME	ADDRESS	MARKS
Harsh	Kolkata	90
Pratik	Delhi	80
Dhanraj	Bihar	95
Ram	Rajasthan	85

LISTING ALL VIEWS IN A DATABASE

We can list View using the SHOW FULL TABLES statement or using the information_schema table. A View can be created from a single table or multiple tables.

Syntax (Using SHOW FULL TABLES):

```
use "database_name";
show full tables where table_type like "%VIEW";
```

Syntax (Using information_schema) :

```
select * from information_schema.views where table_schema = "database_name";

OR

select table_schema,table_name,view_definition from information_schema.views
where table_schema = "database_name";
```

DELETING VIEWS

We have learned about creating a View, but what if a created View is not needed any more? Obviously we will want to delete it. SQL allows us to delete an existing View. We can delete or drop a View using the DROP statement. **Syntax**:

```
DROP VIEW view_name;

view_name: Name of the View which we want to delete.
```

For example, if we want to delete the View **MarksView**, we can do this as:

```
DROP VIEW MarksView;
```

UPDATING VIEWS

There are certain conditions needed to be satisfied to update a view. If any one of these conditions is **not** met, then we will not be allowed to update the view.

1. The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
2. The SELECT statement should not have the DISTINCT keyword.
3. The View should have all NOT NULL values.
4. The view should not be created using nested queries or complex queries.
5. The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view.

- We can use the **CREATE OR REPLACE VIEW** statement to add or remove fields from a view. **Syntax**:

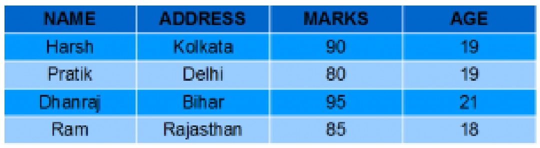
```
CREATE OR REPLACE VIEW view_name AS
SELECT column1,column2,..
FROM table_name
WHERE condition;
```

- For example, if we want to update the view **MarksView** and add the field AGE to this View from **StudentMarks** Table, we can do this as:

```
CREATE OR REPLACE VIEW MarksView AS
SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS,
StudentMarks.AGE
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

- If we fetch all the data from MarksView now as:

```
SELECT * FROM MarksView;
```

- Output: 

NAME	ADDRESS	MARKS	AGE
Harsh	Kolkata	90	19
Pratik	Delhi	80	19
Dhanraj	Bihar	95	21
Ram	Rajasthan	85	18

- **Inserting a row in a view**: We can insert a row in a View in a same way as we do in a table. We can use the INSERT INTO statement of SQL to insert a row in a View.

Syntax:

```
INSERT INTO view_name(column1, column2 , column3,..)
VALUES(value1, value2, value3..);

view_name: Name of the View
```

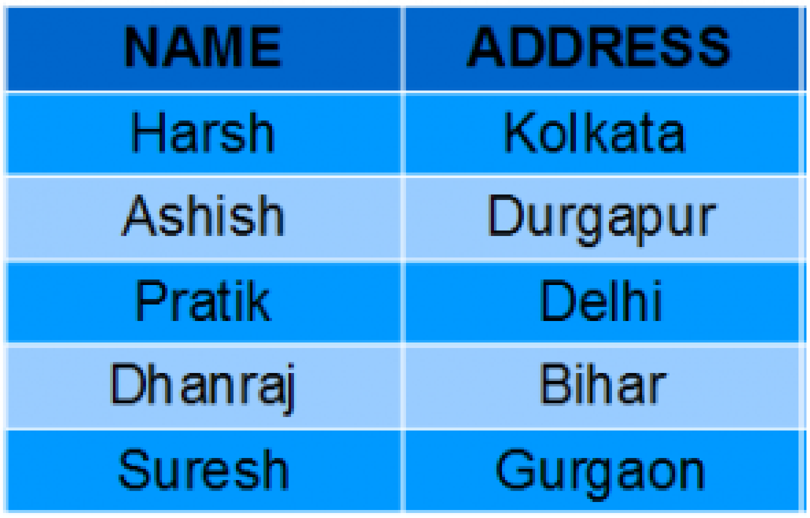
Example: In the below example we will insert a new row in the View DetailsView which we have created above in the example of “creating views from a single table”.

```
INSERT INTO DetailsView(NAME, ADDRESS)
VALUES("Suresh","Gurgaon");
```

- If we fetch all the data from DetailsView now as,

```
SELECT * FROM DetailsView;
```

Output:



NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar
Suresh	Gurgaon

- **Deleting a row from a View**: Deleting rows from a view is also as simple as deleting rows from a table. We can use the DELETE statement of SQL to delete rows from a view. Also deleting a row from a view first delete the row from the actual table and the change is then reflected in the view.

Syntax:

```
DELETE FROM view_name
WHERE condition;

view_name:Name of view from where we want to delete rows
condition: Condition to select rows
```

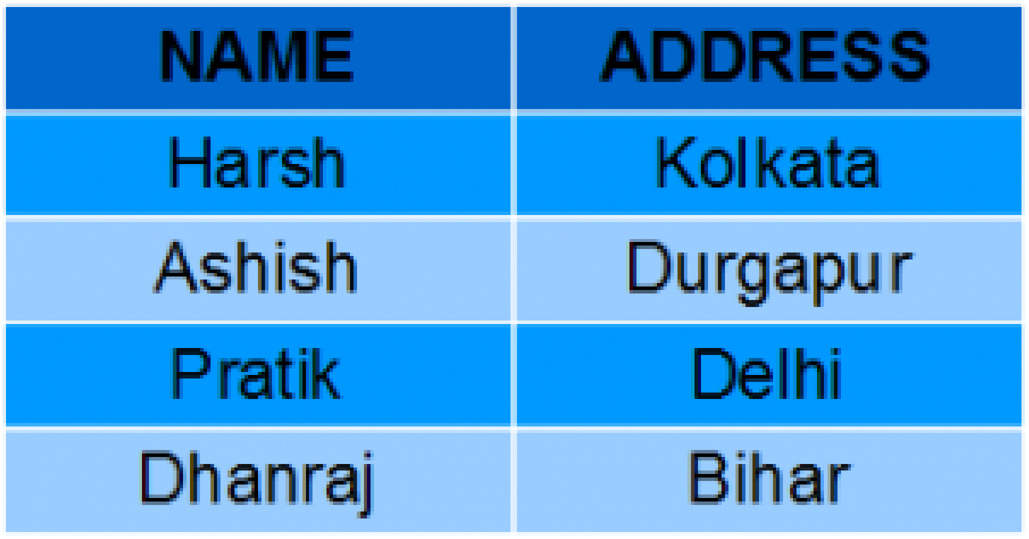
Example: In this example, we will delete the last row from the view DetailsView which we just added in the above example of inserting rows.

```
DELETE FROM DetailsView
WHERE NAME="Suresh";
```

- If we fetch all the data from DetailsView now as,

```
SELECT * FROM DetailsView;
```

Output:



NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

WITH CHECK OPTION

The WITH CHECK OPTION clause in SQL is a very useful clause for views. It is applicable to an updatable view. If the view is not updatable, then there is no meaning of including this clause in the CREATE VIEW statement.

- The WITH CHECK OPTION clause is used to prevent the insertion of rows in the view where the condition in the WHERE clause in CREATE VIEW statement is not satisfied.
- If we have used the WITH CHECK OPTION clause in the CREATE VIEW statement, and if the UPDATE or INSERT clause does not satisfy the conditions then they will return an error.

Example: In the below example we are creating a View SampleView from StudentDetails Table with WITH CHECK OPTION clause.

```
CREATE VIEW SampleView AS
SELECT S_ID, NAME
FROM  StudentDetails
WHERE NAME IS NOT NULL
WITH CHECK OPTION;
```

In this View if we now try to insert a new row with null value in the NAME column then it will give an error because the view is created with the condition for NAME column as NOT NULL. For example,though the View is updatable but then also the below query for this View is not valid:

```
INSERT INTO SampleView(S_ID)
VALUES(6);
```

NOTE*: The default value of NAME column is null.*

Uses of a View: A good database should contain views due to the given reasons:

1. **Restricting data access** – Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table.
2. **Hiding data complexity** – A view can hide the complexity that exists in multiple tables join.
3. **Simplify commands for the user** – Views allow the user to select information from multiple tables without requiring the users to actually know how to perform a join.
4. **Store complex queries** – Views can be used to store complex queries.
5. **Rename Columns** – Views can also be used to rename the columns without affecting the base tables provided the number of columns in view must match the number of columns specified in select statement. Thus, renaming helps to hide the names of the columns of the base tables.
6. **Multiple view facility** – Different views can be created on the same table for different users.

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 21 Mar, 2023 197

Similar Reads

SQL | Creating Roles

shreyanshi_arun

A role is created to ease the setup and maintenance of the security model. It is a named group of related privileges that can be granted to the user. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles:

1. You can grant or revoke privileges to users, thereby automatically granting or revoking privileges.
2. You can either create Roles or use the system roles pre-defined.

Some of the privileges granted to the system roles are as given below:

System Roles	Privileges Granted to the Role
Connect	Create table, Create view, Create synonym, Create sequence, Create session, etc.
Resource	Create Procedure, Create Sequence, Create Table, Create Trigger etc. The primary usage of the Resource role is to restrict access to database objects.
DBA	All system privileges

Creating and Assigning a Role

First, the (Database Administrator)DBA must create the role. Then the DBA can assign privileges to the role and users to the role.

Syntax:

AD

CREATE ROLE manager;

Role created.

Arguments

- *role_name:* Is the name of the role to be created
- owner_name: AUTHORIZATION: If no user is specified, the user who executes CREATE ROLE will be the owner of the role. The role's members can be added or removed at the discretion of the role's owner or any member of an owning role.

Permissions

It requires either membership in the fixed database role db_securityadmin or the CREATE ROLE permission on the database. The following authorizations are also necessary when using the AUTHORIZATION option:

1. It takes IMPERSONATE permission from that user in order to transfer ownership of a role to another user.
2. It takes membership in the recipient role or the ALTER permission on that role to transfer ownership of one role to another.
3. An application role must have ALTER permission in order to transfer ownership of a role to it.

In the syntax: 'manager' is the name of the role to be created.

- Now that the role is created, the DBA can use the GRANT statement to assign users to the role as well as assign privileges to the role.
- It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user.
- If a role is identified by a password, then GRANT or REVOKE privileges have to be identified by the password.

Grant Privileges To a Role

```
GRANT create table, create view
TO manager;
Grant succeeded.
```

Grant a Role To Users

```
GRANT manager TO SAM, STARK;
Grant succeeded.
```

Revoke Privilege from a Role

```
REVOKE create table FROM manager;
```

Drop a Role

```
DROP ROLE manager;
```

Explanation

Firstly it creates a manager role and then allows managers to create tables and views. It then grants Sam and Stark the role of managers. Now Sam and Stark can create tables and views. If users have multiple roles granted to them, they receive all of the privileges associated with all of the roles. Then create table privilege is removed from the role 'manager' using Revoke. The role is dropped from the database using drop.

Last Updated : 30 May, 2023

95

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. Database Roles in CQL (Cassandra Query Language)
5. Granting Permissions to Roles in Cassandra
6. Key Roles for Data Analytics project
7. Privilege and Roles in DBMS

SQL indexes

MrinalVerma

An index is a schema object. It is used by the server to speed up the retrieval of rows by using a pointer. It can reduce disk I/O(input/output) by using a rapid path access method to locate data quickly. An index helps to speed up select queries and where clauses, but it slows down data input, with the update and the insert statements. Indexes can be created or dropped with no effect on the data. In this article, we will see how to create, delete, and uses the INDEX in the database.

For example, if you want to reference all pages in a book that discusses a certain topic, you first refer to the index, which lists all the topics alphabetically and is then referred to one or more specific page numbers.

Creating an Index:

Syntax:

```
CREATE INDEX index
ON TABLE column;
```

where the **index** is the name given to that index and **TABLE** is the name of the table on which that index is created and **column** is the name of that column for which it is applied.

For multiple columns:

Syntax:

```
CREATE INDEX index
ON TABLE (column1, column2,.....);
```

Unique Indexes: Unique indexes are used for the maintenance of the integrity of the data present in the table as well as for fast performance, it does not allow multiple values to enter into the table.

Syntax:

```
CREATE UNIQUE INDEX index
ON TABLE column;
```

When should indexes be created:

- A column contains a wide range of values.
- A column does not contain a large number of null values.
- One or more columns are frequently used together in a where clause or a join condition.

When should indexes be avoided:

- The table is small
- The columns are not often used as a condition in the query
- The column is updated frequently

Removing an Index: Remove an index from the data dictionary by using the **DROP INDEX** command.

Syntax:

```
DROP INDEX index;
```

To drop an index, you must be the owner of the index or have the **DROP ANY INDEX** privilege.

Altering an Index: To modify an existing table's index by rebuilding, or reorganizing the index.

```
ALTER INDEX IndexName
ON TableName REBUILD;
```

Confirming Indexes: You can check the different indexes present in a particular table given by the user or the server itself and their uniqueness.

Syntax:

```
select * from USER_INDEXES;
```

It will show you all the indexes present in the server, in which you can locate your own tables too.

Renaming an index: You can use the system-stored procedure sp_rename to rename any index in the database.

Syntax:

```
EXEC sp_rename
    index_name,
    new_index_name,
    N'INDEX';
```

Lastly, let us discuss why should we use indexing?

Indexing is an important topic when considering advanced MySQL, although most people know about its definition and usage they don’t understand when and where to use it to change the efficiency of our queries or stored procedures by a huge margin.

Here are some scenarios along with their explanation related to Indexing:

1. When executing a query on a table having huge data (> 100000 rows), MySQL performs a full table scan which takes much time and the server usually gets timed out. To avoid this always check the explain option for the query within MySQL which tells us about the state of execution. It shows which columns are being used and whether it will be a threat to huge data. On basis of the columns repeated in a similar order in conditions, we can create an index for them in the same order to maximize the speed of the query.
2. The order of the index is of huge importance as we can use the same index in many scenarios. Using only one index we can utilize it in more than one query which different conditions. like for example, in a query, we make a join with a table based on customer_id wards we also join another join based on customer_id and order_date. Then we can simply create a single index by the order of customer_id, order_date which would be used in both cases. This also saves storage.
3. We should also be careful to not make an index for each query as creating indexes also take storage and when the amount of data is huge it will create a problem. Therefore, it’s important to carefully consider which columns to index based on the needs of your application. In general, it’s a good practice to only create indexes on columns that are frequently used in queries and to avoid creating indexes on columns that are rarely used. It’s also a good idea to periodically review the indexes in your database and remove any that are no longer needed.
4. Indexes can also improve performance when used in conjunction with sorting and grouping operations. For example, if you frequently sort or group data based on a particular column, creating an index on that column can greatly improve performance. The index allows MySQL to quickly access and sort or group the data, rather than having to perform a full table scan.
5. In some cases, MySQL may not use an index even if one exists. This can happen if the query optimizer determines that a full table scan is faster than using the index. This can occur

when the table is small, the query is highly selective, or the table has a high rate of updates.

Last Updated : 10 Mar, 2023

90

Similar Reads

1. Multiple Indexes vs Multi-Column Indexes
2. SQL queries on clustered and non-clustered Indexes
3. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
4. Configure SQL Jobs in SQL Server using T-SQL
5. SQL vs NO SQL vs NEW SQL
6. Secondary Indexes on MAP Collection in Cassandra
7. Secondary Indexes on LIST Collection in Cassandra
8. Secondary Indexes on SET Collection in Cassandra
9. How to Compare Indexes of Tables From Two Different Databases in Oracle?
10. SQL | Procedures in PL/SQL

Previous Next

Article Contributed By :

MrinalVerma
MrinalVerma

Vote for difficulty

Current difficulty : Easy

Easy Normal Medium Hard Expert

Improved By : sunny94, khushboogoyal499, swchoi442, varshachoudhary, adichavan095

Article Tags : DBMS Indexing, DBMS-SQL, SQL-basics, DBMS, SQL

Practice Tags : DBMS, SQL

SQL | SEQUENCES

SQL sequences specifies the properties of a sequence object while creating it. An object bound to a user-defined schema called a sequence produces a series of numerical values in accordance with the specification used to create it. The series of numerical values can be configured to restart (cycle) when it runs out and is generated in either ascending or descending order at a predetermined interval. Contrary to identity columns, sequences are not linked to particular tables. Applications use a sequence object to access the next value in the sequence. The application has control over how sequences and tables interact. A sequence object can be referred to by user applications, and the values can be coordinated across various rows and tables.

Let's suppose that sequence is a set of integers 1, 2, 3, … that are generated and supported by some database systems to produce unique values on demands.

- A sequence is a user-defined schema-bound object that generates a series of numeric values.
- Sequences are frequently used in many databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them.
- The sequence of numeric values is generated in an ascending or descending order at defined intervals and can be configured to restart when it exceeds max_value.

Different Features of Sequences

1. A sequence is a database object that generates and produces integer values in sequential order.
2. It automatically generates the primary key and unique key values.
3. It may be in ascending or descending order.
4. It can be used for multiple tables.
5. Sequence numbers are stored and generated independently of tables.
6. It saves time by reducing application code.
7. It is used to generate unique integers.
8. It is used to create an auto number field.
9. Useful when you need to create a unique number to act as a primary key.

10. Oracle provides an object called a Sequence that can generate numeric values. The value generated can have maximum of 38 digits
11. Provide intervals between numbers.

Syntax:

CREATE SEQUENCE sequence_name

AD

START WITH initial_value

INCREMENT BY increment_value

MINVALUE minimum value

MAXVALUE maximum value

CYCLE|NOCYCLE ;

Following is the sequence query creating a sequence in ascending order.

Example 1:

```
CREATE SEQUENCE sequence_1
start with 1
increment by 1
minvalue 0
maxvalue 100
cycle;
```

The above query will create a sequence named *sequence_1*. The sequence will start from 1 and will be incremented by 1 having maximum value of 100. The sequence will repeat itself from the start value after exceeding 100.

Example 2: Following is the sequence query creating a sequence in descending order.

```
CREATE SEQUENCE sequence_2
start with 100
increment by -1
```

```
min value 1
max value 100
cycle;
```

The above query will create a sequence named *sequence_2*. The sequence will start from 100 and should be less than or equal to a maximum value and will be incremented by -1 having a minimum value of 1.

Example To Use Sequence:

Create a table named students with columns as id and name.

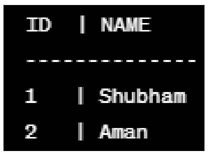
```
CREATE TABLE students
(
ID number(10),
NAME char(20)
);
```

Now insert values into a table

```
INSERT into students VALUES
(sequence_1.nextval,'Shubham');
INSERT into students VALUES
(sequence_1.nextval,'Aman');
```

where *sequence_1.nextval* will insert id's in the id column in a sequence as defined in sequence_1.

Output:



```
ID  | NAME
--------------
1   | Shubham
2   | Aman
```

Cache Management

To enhance performance, SQL Server employs cache management for sequence numbers. The CACHE argument determines the number of sequence numbers pre-allocated by the server.

For instance, let's consider a new sequence with a starting value of 1 and a cache size of 15. When the first number is required, values 1 through 15 are fetched from memory and made available. The last cached value (15) is written to the system tables on disk. As all 15 numbers are utilized, the next request (for number 16) triggers the allocation of a new cache. The latest cached value (30) is then stored in the system tables.

In the event of a SQL Server shutdown after using 22 numbers, the next intended sequence number (23) in memory replaces the previously stored number in the system tables. Upon restarting, when a sequence number is needed, the starting number (23) is read from the system tables. A cache of 15 numbers (23-38) is allocated to memory, and the next number outside the cache (39) is written to the system tables.

If an abnormal shutdown occurs, such as a power failure, the sequence restarts from the number read from the system tables (39). Any sequence numbers allocated to memory but not requested by users or applications are lost. This behavior can result in gaps, but it ensures that a single sequence object will never issue the same value twice unless it is defined as CYCLE or manually restarted.

The cache is managed in memory by tracking the current value (the last issued value) and the remaining values in the cache. Hence, the cache consumes memory equivalent to two instances of the sequence object's data type.

When the cache argument is set to NO CACHE, the current sequence value is written to the system tables every time a sequence is used. This approach may slow down performance due to increased disk access, but it reduces the likelihood of unintended gaps. However, gaps can still occur if numbers are requested using the NEXT VALUE FOR or sp_sequence_get_range functions but are either unused or used within uncommitted transactions.

This article is contributed by **ARSHPREET SINGH**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 05 Jun, 2023 67

Similar Reads

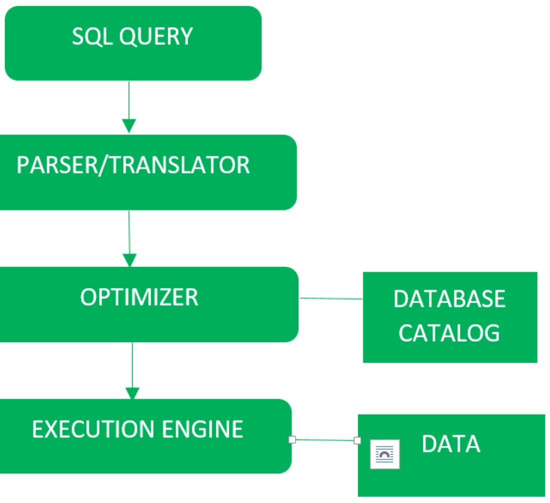
1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. SQL | Procedures in PL/SQL
5. SQL | Difference between functions and stored procedures in PL/SQL
6. SQL SERVER – Input and Output Parameter For Dynamic SQL
7. Difference between SQL and T-SQL

SQL | Query Processing

priyankagujral

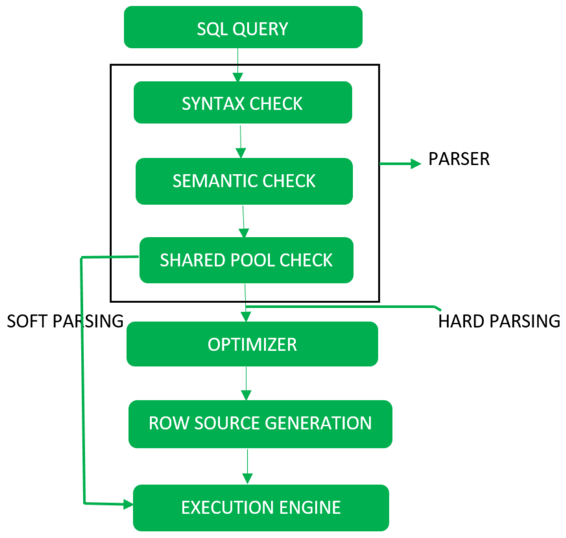
Query Processing includes translations on high level Queries into low level expressions that can be used at physical level of file system, query optimization and actual execution of query to get the actual result.

Block Diagram of Query Processing is as:



Detailed Diagram is drawn as:

AD



It is done in the following steps:

- **Step-1:**
  Parser: During parse call, the database performs the following checks- Syntax check, Semantic check and Shared pool check, after converting the query into relational algebra. Parser performs the following checks as (refer detailed diagram):

  1. **Syntax check** – concludes SQL syntactic validity. Example:

     ```
     SELECT * FORM employee
     ```

     Here error of wrong spelling of FROM is given by this check.
  2. **Semantic check** – determines whether the statement is meaningful or not. Example: query contains a tablename which does not exist is checked by this check.
  3. **Shared Pool check** – Every query possess a hash code during its execution. So, this check determines existence of written hash code in shared pool if code exists in shared pool then database will not take additional steps for optimization and execution.

  Hard Parse and Soft Parse –
  If there is a fresh query and its hash code does not exist in shared pool then that query has to pass through from the additional steps known as hard parsing otherwise if hash code exists then query does not passes through additional steps. It just passes directly to execution engine (refer detailed diagram). This is known as soft parsing.
  Hard Parse includes following steps – Optimizer and Row source generation.

- **Step-2:**
  Optimizer: During optimization stage, database must perform a hard parse atleast for one unique DML statement and perform optimization during this parse. This database never optimizes DDL unless it includes a DML component such as subquery that require optimization.
  It is a process in which multiple query execution plan for satisfying a query are examined and most efficient query plan is satisfied for execution.
  Database catalog stores the execution plans and then optimizer passes the lowest cost plan for execution.

Row Source Generation –
The Row Source Generation is a software that receives a optimal execution plan from the optimizer and produces an iterative execution plan that is usable by the rest of the database. the iterative plan is the binary program that when executes by the sql engine produces the result set.

- **Step-3:**
  Execution Engine: Finally runs the query and display the required result.

Last Updated : 14 Aug, 2018 52

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Online Transaction Processing (OLTP) and Online Analytic Processing (OLAP)
3. SQL Query to Add Email Validation Using Only One Query
4. SQL Query to Check if Date is Greater Than Today in SQL
5. SQL Query to Add a New Column After an Existing Column in SQL
6. SQL Query to Convert Rows to Columns in SQL Server
7. Query Processing in Distributed DBMS
8. Configure SQL Jobs in SQL Server using T-SQL
9. SQL vs NO SQL vs NEW SQL
10. SQL | SELECT Query

Previous Next

Article Contributed By :

priyankagujral
priyankagujral

Vote for difficulty

Current difficulty : Medium

Types of Keys in Relational Model (Candidate, Super, Primary, Alternate and Foreign)

Read Discuss

Pre-Requisite: DBMS | Relational Model Introduction and Codd Rules

Keys are one of the basic requirements of a relational database model. It is widely used to identify the tuples(rows) uniquely in the table. We also use keys to set up relations amongst various columns and tables of a relational database.

Different Types of Keys in the Relational Model

1. Candidate Key
2. Primary Key
3. Super Key
4. Alternate Key
5. Foreign Key

1. Candidate Key: The minimal set of attributes that can uniquely identify a tuple is known as a candidate key. For Example, STUD_NO in STUDENT relation.

- It is a minimal super key.
- It is a super key with no repeated data is called a candidate key.
- The minimal set of attributes that can uniquely identify a record.
- It must contain unique values.
- It can contain NULL values.
- Every table must have at least a single candidate key.
- A table can have multiple candidate keys but only one primary key (the primary key cannot have a NULL value, so the candidate key with a NULL value can't be the primary key).
- The value of the Candidate Key is unique and may be null for a tuple.
- There can be more than one candidate key in a relationship.

Example:

```
STUD_NO is the candidate key for relation STUDENT.
```

Table STUDENT

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

- The candidate key can be simple (having only one attribute) or composite as well.

Example:

```
{STUD_NO, COURSE_NO} is a composite
candidate key for relation STUDENT_COURSE.
```

Table STUDENT_COURSE

STUD_NO	TEACHER_NO	COURSE_NO
1	001	C001
2	056	C005

Note: In SQL Server a unique constraint that has a nullable column, **allows** the value '**null**' in that column **only once**. That's why the STUD_PHONE attribute is a candidate here, but can not be a 'null' value in the primary key attribute.

2. Primary Key: There can be more than one candidate key in relation out of which one can be chosen as the primary key. For Example, STUD_NO, as well as STUD_PHONE, are candidate keys for relation STUDENT but STUD_NO can be chosen as the primary key (only one out of many candidate keys).

- It is a unique key.
- It can identify only one tuple (a record) at a time.
- It has no duplicate values, it has unique values.
- It cannot be NULL.
- Primary keys are not necessarily to be a single column; more than one column can also be a primary key for a table.

Example:

```
STUDENT table -> Student(STUD_NO, SNAME,
ADDRESS, PHONE) , STUD_NO is a primary key
```

Table STUDENT

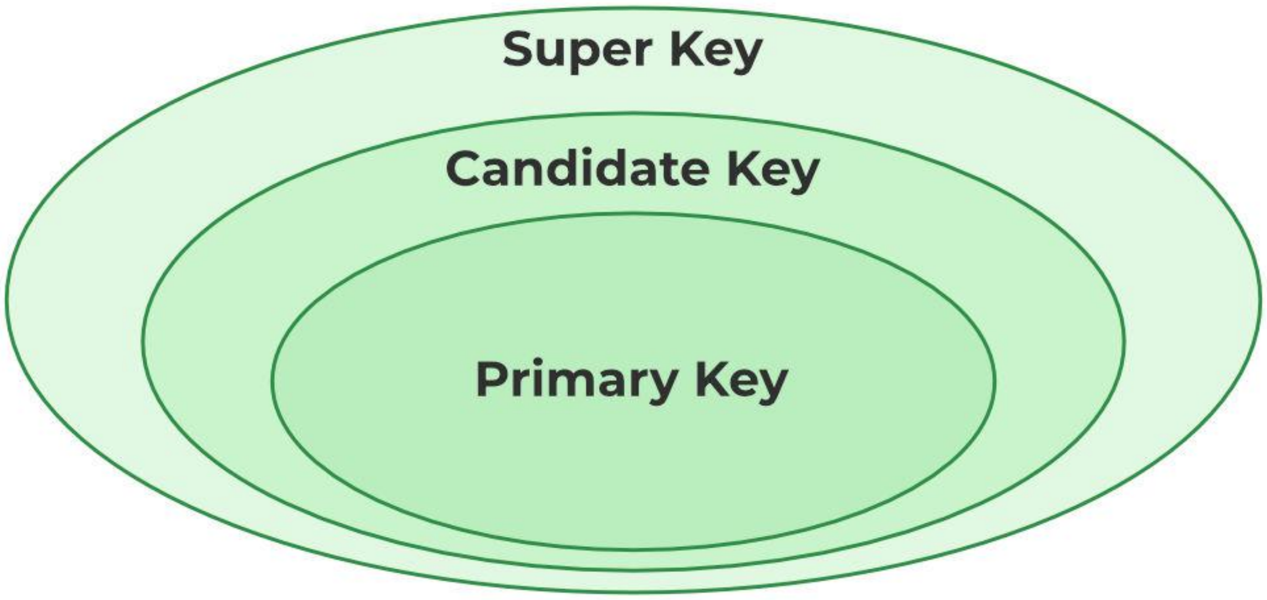
STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

3. Super Key: The set of attributes that can uniquely identify a tuple is known as Super Key. For Example, STUD_NO, (STUD_NO, STUD_NAME), etc. A super key is a group of single or multiple keys that identifies rows in a table. It supports NULL values.

- Adding zero or more attributes to the candidate key generates the super key.
- A candidate key is a super key but vice versa is not true.

Example:

```
Consider the table shown above.
STUD_NO+PHONE is a super key.
```



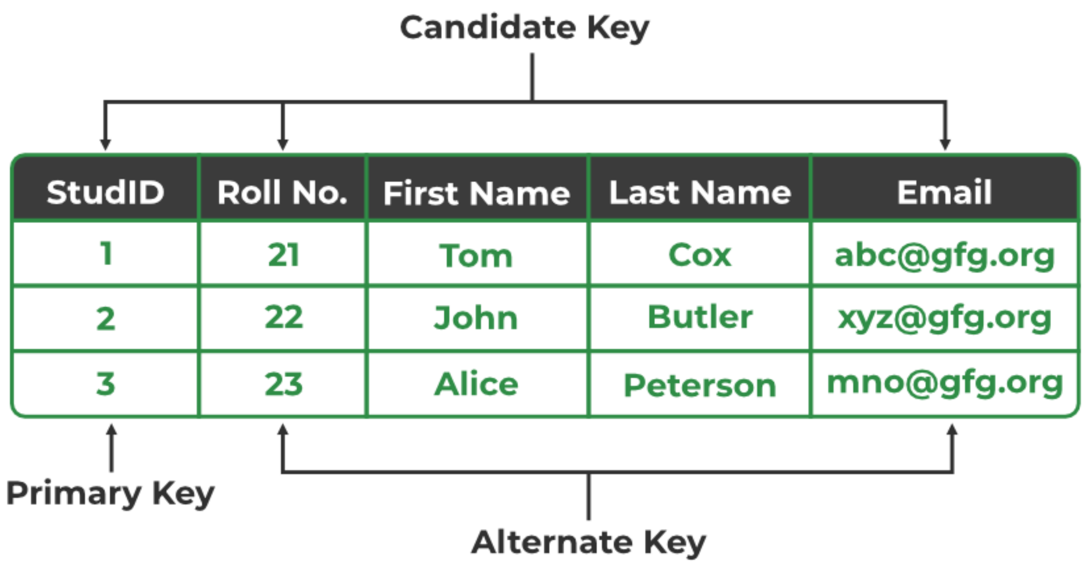
Relation between Primary Key, Candidate Key, and Super Key

4. Alternate Key: The candidate key other than the primary key is called an alternate key.

- All the keys which are not primary keys are called alternate keys.
- It is a secondary key.
- It contains two or more fields to identify two or more records.
- These values are repeated.
- Eg:- SNAME, and ADDRESS is Alternate keys

Example:

```
Consider the table shown above.
STUD_NO, as well as PHONE both,
are candidate keys for relation STUDENT but
PHONE will be an alternate key
(only one out of many candidate keys).
```



StudID	Roll No.	First Name	Last Name	Email
1	21	Tom	Cox	abc@gfg.org
2	22	John	Butler	xyz@gfg.org
3	23	Alice	Peterson	mno@gfg.org

Primary Key, Candidate Key, and Alternate Key

5. Foreign Key: If an attribute can only take the values which are present as values of some other attribute, it will be a foreign key to the attribute to which it refers. The relation which is being referenced is called referenced relation and the corresponding attribute is called referenced attribute the relation which refers to the referenced relation is called referencing relation and the corresponding attribute is called referencing attribute. The referenced attribute of the referenced relation should be the primary key to it.

- It is a key it acts as a primary key in one table and it acts as secondary key in another table.
- It combines two or more relations (tables) at a time.
- They act as a cross-reference between the tables.
- For example, DNO is a primary key in the DEPT table and a non-key in EMP

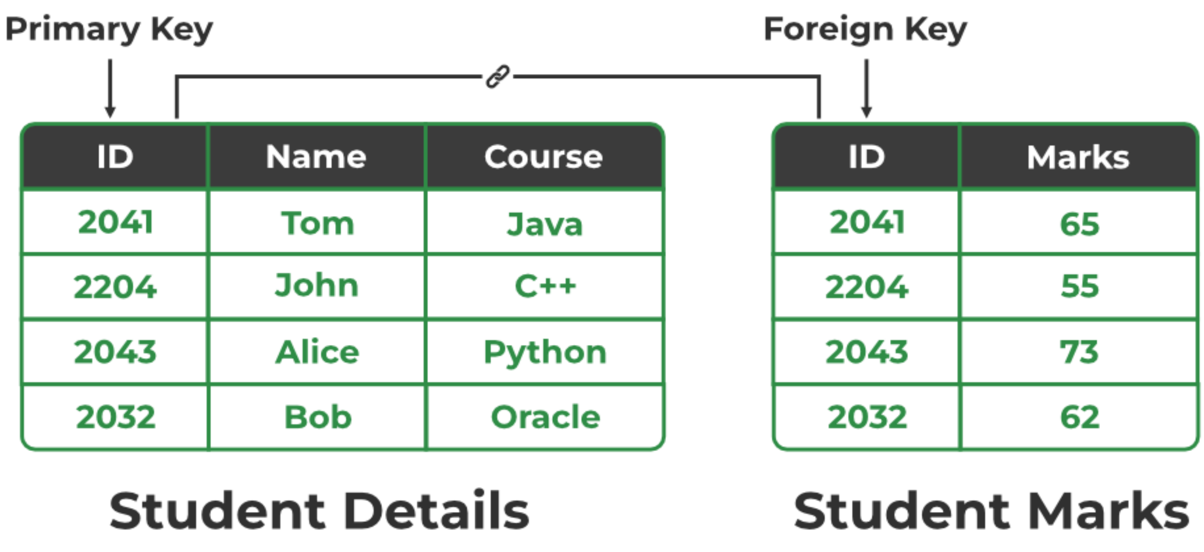
Example:

```
Refer Table STUDENT shown above.
STUD_NO in STUDENT_COURSE is a
foreign key to STUD_NO in STUDENT relation.
```

Table STUDENT_COURSE

STUD_NO	TEACHER_NO	COURSE_NO
1	005	C001
2	056	C005

It may be worth noting that, unlike the Primary Key of any given relation, Foreign Key can be NULL as well as may contain duplicate tuples i.e. it need not follow uniqueness constraint. For Example, STUD_NO in the STUDENT_COURSE relation is not unique. It has been repeated for the first and third tuples. However, the STUD_NO in STUDENT relation is a primary key and it needs to be always unique, and it cannot be null.



ID	Name	Course
2041	Tom	Java
2204	John	C++
2043	Alice	Python
2032	Bob	Oracle

ID	Marks
2041	65
2204	55
2043	73
2032	62

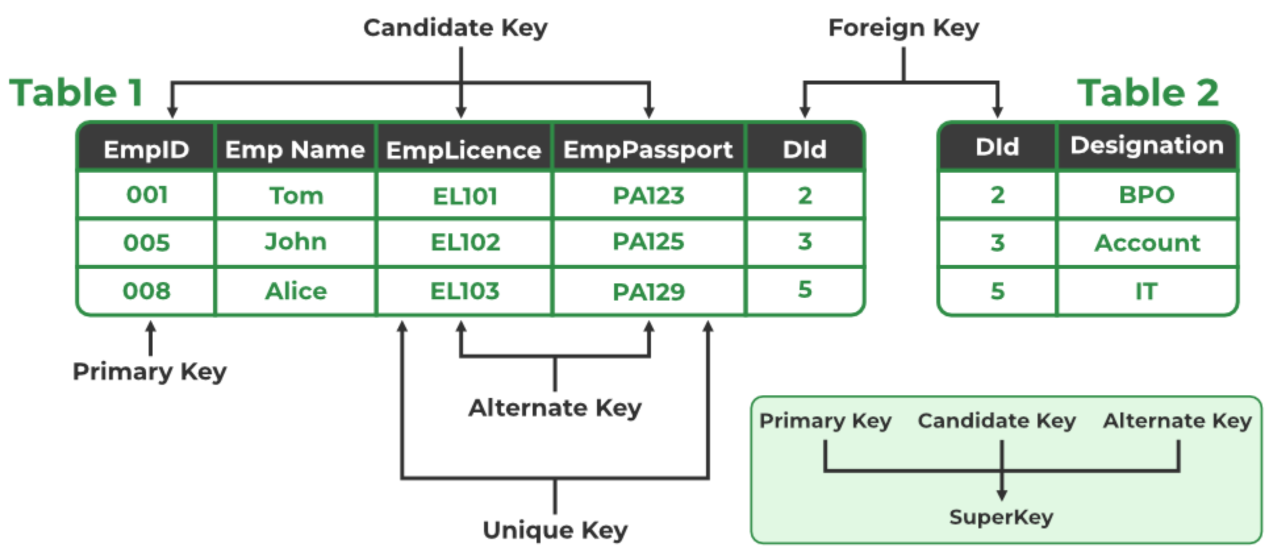
Relation between Primary Key and Foreign Key

6. Composite Key: Sometimes, a table might not have a single column/attribute that uniquely identifies all the records of a table. To uniquely identify rows of a table, a combination of two or more columns/attributes can be used. It still can give duplicate values in rare cases. So, we need to find the optimal set of attributes that can uniquely identify rows in a table.

- It acts as a primary key if there is no primary key in a table
- Two or more attributes are used together to make a composite key.
- Different combinations of attributes may give different accuracy in terms of identifying the rows uniquely.

Example:

```
FULLNAME + DOB can be combined
together to access the details of a student.
```



EmpID	Emp Name	EmpLicence	EmpPassport	DId
001	Tom	EL101	PA123	2
005	John	EL102	PA125	3
008	Alice	EL103	PA129	5

DId	Designation
2	BPO
3	Account
5	IT

Different Types of Keys

FAQs

Why keys are necessary for DBMS?

- Keys are one of the important aspects of DBMS. Keys help us to find the tuples(rows) uniquely in the table. It is also used in developing various relations amongst columns or tables of the database.

What is a Unique Key?

- Unique Keys are the keys that define the record uniquely in the table. It is different from Primary Keys, as Unique Key can contain one NULL value but Primary Key does not contain any NULL values.

What is Artificial Key?

- Artificial Keys are the keys that are used when no attributes contain all the properties of the Primary Key or if the Primary key is very large and complex.

Last Updated : 21 Mar, 2023

378

Similar Reads

1. Difference between Primary and Candidate Key
2. Difference between Super Key and Candidate Key
3. Why Candidate Key is Called a Minimal Super Key?
4. Difference between Primary Key and Foreign Key