|  |  |
| --- | --- |
| File system | Relational database |
| 1. It is a way of arranging the files in storage medium within a computer. 2. Redundant is present in file system. 3. It doesn’t provide backup and recovery if data is lost. 4. There is no efficient query processing in the file system. 5. There is less data consistency in the file system. 6. It is cheaper to design | 1. It is a software for managing the database. 2. There is no redundant data in dbms. 3. It provides backup and recovery even if data is lost. 4. Efficient query processing is there in DBMS. 5. There is more data consistency because of the process of normalization. 6. It is expensive to design |

**Data definition language:**

The DDL Commands in Structured Query Language are used to create and modify the schema of the database and its objects. The syntax of DDL commands is predefined for describing the data. The commands of Data Definition Language deal with how the data should exist in the database.

The commands present in DDL are:

CREATE

DROP

RENAME

ALTER

TUNCATE

**Data manipulation language:**

Data Manipulation Language (DML) is a subset of SQL (Structured Query Language) used to interact with and manipulate data stored in a database. They allow you to insert new data, update existing data, retrieve specific information, and delete unwanted data. These statements are the core building blocks for applications that manage and utilize data stored in relational databases.

The main commands present in DML are:

UPDATE

INSERT

DELETE

**Data query language:**

Data Query Language (DQL) is a subset of SQL (Structured Query Language) that focuses on retrieving data from a database. DQL is primarily concerned with the **SELECT** statement, which is used to retrieve specific data based on specified criteria. In SQL Server, DQL allows you to retrieve data from one or more tables, filter and sort the data, and perform various operations on the retrieved data.

The main command used in DQL is

SELECT

**Transaction control language:**

Transaction Control Language (TCL) in SQL Server consists of statements used to manage transactions within a database. TCL statements allow you to control the beginning, ending, and behavior of transactions. Transactions are sequences of one or more SQL statements that are executed as a single unit of work, ensuring data integrity and consistency.

TCL statements are crucial for maintaining data consistency and ensuring that transactions behave as expected, even in case of errors or unexpected events. Proper use of these statements can help you manage and control the state of your database in situations where multiple operations need to be performed as a single unit.

For example, consider a banking application where transferring money from one account to another involves deducting from one account and adding to another. Using transactions, you can ensure that either both operations succeed (COMMIT) or neither operation takes effect (ROLLBACK) to maintain the integrity of account balances.

The main commands in TCL are

BEGIN TRANSACTION ,

SAVEPOINT ,

COMMIT,

ROLLBACK

**Data control language:**

DCL stands for "Data Control Language." DCL is a subset of SQL statements that are used to control access and permissions on database objects. DCL statements are used to manage the security aspects of a database, including granting or revoking permissions on tables, views, stored procedures, and other database objects.

There are two main DCL statements in SQL Server:

GRANT

REVOKE

EXAMPLE:

GRANT SELECT, INSERT ON dbo.MyTable TO MyUser;

REVOKE DELETE ON dbo.MyTable FROM RestrictedRole;

**SQL 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.

**Not null constraint in sql server**

In SQL Server, a "NOT NULL" constraint is used to ensure that a column in a table does not contain any NULL values. This constraint enforces that every entry in that column must have a non-null value. If an attempt is made to insert or update a row with a NULL value in a column that has a "NOT NULL" constraint, SQL Server will raise an error and prevent the operation.

Here is the syntax for not null

While creating a table

CREATE TABLE Employees (

EmployeeID INT NOT NULL,

FirstName VARCHAR(50) ,

LastName VARCHAR(50) ,

);

For adding not null to the existing table

ALTER TABLE Employees

ALTER COLUMN FirstName VARCHAR(50) NOT NULL;

**UNIQUE constraint in sql server:**

In SQL Server, a "UNIQUE" constraint ensures that the values in a specific column or combination of columns are unique across all the rows in a table. This means that no two rows can have the same values in the "UNIQUE" columns. It's often used to enforce data integrity and to prevent duplicate entries.

Here is the syntax for unique constraint

While creating a table

create table employees(

id int unique not null

)

Adding unique to the existing table

ALTER TABLE Employees

ADD CONSTRAINT UQ\_id UNIQUE (id);

**DEFAULT constraint in sql server :**

In SQL Server, a "DEFAULT" constraint is used to specify a default value for a column. If an INSERT operation does not provide a value for a column with a "DEFAULT" constraint, the default value is automatically used.

Here is the syntax for default constraint

While creating a table

create table employees(

id int unique not null,

joining\_date datetime default getdate()

)

Adding default to existing table

ALTER TABLE Employees

ADD CONSTRAINT DF\_joining\_date default getdate()for joining\_date;

**Check constraint in sql sever:**

In SQL Server, a check constraint is a type of constraint that is used to enforce data integrity rules by limiting the values that can be inserted or updated in a column within a table. Check constraints are specified at the column level and are evaluated whenever data is modified in the table. If a check constraint condition is not met, the database will reject the operation, ensuring that only valid data is stored in the table.

Here is the syntax for the check constraint

While creating a table

CREATE TABLE Students

(

StudentID INT PRIMARY KEY,

Name NVARCHAR(255),

Age INT CHECK (Age >= 18)

);

Adding check to the existing table

ALTER TABLE your\_table

ADD CONSTRAINT constraint\_name CHECK (condition);

To disable and enable check constraint

ALTER TABLE Students

NOCHECK CONSTRAINT CHK\_StudentAge;

To enable check constraint which is disabled

ALTER TABLE Students

CHECK CONSTRAINT CHK\_StudentAge;

**Primary key constraint in sql server :**

In SQL Server, a primary key constraint is a database object that enforces the uniqueness of values in one or more columns of a table. The primary key constraint also ensures that the specified column(s) do not contain NULL values. It is used to uniquely identify each row in a table and provides a way to establish relationships between tables (through foreign keys).

We can give primary key for more than one column but we can give it one time for a table.

Here is the syntax for creating primary key in sql server

While creating a table

CREATE TABLE Students

(

StudentID INT PRIMARY KEY,

Name NVARCHAR(255),

Age INT

);

Add primary key for an existing table

ALTER TABLE Employees

ADD CONSTRAINT PK\_Employees PRIMARY KEY (EmployeeID, DepartmentID);

**Foreign key in sql server**

In SQL Server, a foreign key (often abbreviated as FK) is a database constraint that enforces referential integrity between two tables. It specifies a relationship between a column (or columns) in one table (the referencing or child table) and the primary key or unique constraint of another table (the referenced or parent table). The foreign key constraint ensures that the values in the referencing column(s) match values in the referenced primary key or unique constraint.

Here is the syntax for creating foreign key constraint

While creating a table

CREATE TABLE Orders

(

OrderID INT PRIMARY KEY,

CustomerID INT,

ProductID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

Add foreign key to the existing table

ALTER TABLE Orders

ADD CONSTRAINT FK\_CustomerID

FOREIGN KEY (CustomerID)

REFERENCES Customers(CustomerID);

* A foreign key constraint can be linked to a primary key or unique constraint in another table
* When a value other than null is entered that is not there in a referenced column then it will through an error while inserting the row in the table
* Foreign key can reference the tables only with in the same database, cross data base referential integrity must be implemented through triggers.
* foreign key is not enforced on temp tables

to disable all the constraints on a table

ALTER TABLE <table name> NOCHECK CONSTRAINT ALL

To enable all the constraints on a table

ALTER TABLE <table name> WITH CHECK CHECK CONSTRAINT ALL

**Data types in sql server**

**Numeric data types**

* TINYINT: stores an 8bit integer value range from 0 to 255.
* SMALLINT: Stores a 16-bit integer value ranging from -32,768 to 32,767.
* INT: Stores a 32-bit integer value ranging from -2,147,483,648 to 2,147,483,647.
* BIGINT: Stores a 64-bit integer value ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
* MONEY: store monetary values with different precession
* FLOAT: Stores decimal values

**Date and time data types:**

* DATETIME: stores date and time value including fraction of seconds
* SMALLDATETIME: Stores date and time values with less precision.
* DATE: Stores date values only.
* TIME: Stores time values only.
* DATETIME2: Stores date and time values with greater fractional seconds precision.
* DATETIMEOFFSET: Stores date and time values with time zone information.

**Character data types:**

* CHAR(n): Stores fixed-length character strings.
* VARCHAR(n): Stores variable-length character strings.
* NCHAR(n): Stores fixed-length Unicode character strings.
* NVARCHAR(n): Stores variable-length Unicode character strings.
* TEXT: Stores large text values (deprecated, use VARCHAR(MAX) or NVARCHAR(MAX) instead).
* NTEXT: Stores large Unicode text values (deprecated, use NVARCHAR(MAX) instead).

**Binary data types:**

* BINARY(n): Stores fixed-length binary data.
* VARBINARY(n): Stores variable-length binary data.
* IMAGE: Stores large binary data (deprecated, use VARBINARY(MAX) instead).

**Inserting rows from one table to another table:**

INSERT INTO TargetTable

SELECT Column1, Column2, Column3

FROM SourceTable;

INSERT INTO TargetTable (FirstName, LastName, Age)

SELECT FirstName, LastName, Age

FROM SourceTable

WHERE Age >= 18;

SELECT \* INTO new\_table FROM old\_table

But the above statement have some limitations

* we cannot change the column attributes while creating a new table
* indexes, triggers, primary key ,constraints, identity, computed columns will not be inherited from the old table
* permission of the old table will not be inherited to the new table
* if we use to copy a large data set there will be some performance and transaction log issues

**Identity in SQL server :**

The IDENTITY keyword is a property in SQL Server. **When a table column is defined with an identity property, its value will be auto-generated incremental value**. This value is created by the server automatically. Therefore, we can't manually enter a value into an identity column as a user. Hence, if we mark a column as identity, SQL Server will populate it in an auto-increment manner.

Here is the syntax for identity column

Identity (seed, increment)

Seed – it is the starting value of an identity column by default its value is 1.

Increment--- It is the value that needs to be added to the seed by default its value is 1,

**While creating a table**

Create table students

(

Id int Identity (1,1) primary key

)

**For and existing table**

Alter table student

Add personid int identity(10,1) primary key;

* We can add new identity column but not identity to an existing column
* If we try to insert a value into an existing identity column explicitly it will through an error as cannot insert identity column explicitly
* if we want to insert a value into an existing identity column explicitly then set identity\_insert on  
  SET IDENTITY\_INSERT YourTableName ON;  
  after inserting the value   
  SET IDENTITY\_INSERT YourTableName OFF;

**Update, delete and truncate in SQL server:**

1. The update statement is used to update the existing records in a table.
2. Delete statement is used to delete selected rows in a existing records in a table
3. Truncate is used to delete all the existing rows in a table but it will not delete the schema of the table the table schema and properties will remains same only the data will be deleted.

syntax for update, delete and truncate

**update**

UPDATE Customers

SET City = 'New York'

WHERE CustomerID = 1;

**delete**

DELETE FROM Orders

WHERE OrderID = 1001;

**truncate**

TRUNCATE TABLE Products;

**computed columns in SQL Server:**

The computed column in a table contains values which are derived or calculated from another columns in a table using pre defined expressions or formulas

The computed columns are used for performing mathematical calculations, string concatenation

Applying date calculations and operations like which reduce manual calculation

**Syntax to add a computed column**

**While creating a table**

CREATE TABLE dbo.Products

(

ProductID int IDENTITY (1,1) NOT NULL

, QtyAvailable smallint

, UnitPrice money

, InventoryValue AS QtyAvailable \* UnitPrice

);

**To add computed column to an existing table**

ALTER TABLE dbo.Products ADD RetailValue AS (QtyAvailable \* UnitPrice \* 1.5) PERSISTED;

**To make existing column as a computed column**

ALTER TABLE dbo.Products DROP COLUMN RetailValue;

GO

ALTER TABLE dbo.Products ADD RetailValue AS (QtyAvailable \* UnitPrice \* 1.5);

GO

**Select-order by clause**

It is used to order the result set of a query by the specified column list and, optionally, limit the rows returned to a specified range. The order in which rows are returned in a result set are not guaranteed unless an ORDER BY clause is specified.

SELECT ProductID, Name FROM Production.Product

WHERE Name LIKE 'Lock Washer%'

ORDER BY ProductID;

**Limitations of order by**

The total size of the columns specified in an order by clause cannot exceed 8,060 bytes

If a table name is aliased in the FROM clause, only the alias name can be used to qualify its columns in the ORDER BY clause.

In a query that uses UNION, EXCEPT, or INTERSECT operators, ORDER BY is allowed only at the end of the statement. This restriction applies only to when you specify UNION, EXCEPT, and INTERSECT in a top-level query and not in a subquery

**Predicates in sql server**

A predicate is an expression that evaluates to true, false, or unknown. Predicates are used in the search condition of where and having clauses, the join conditions of from clauses, and other constructs where a Boolean value is required.

Sql server provides the following predicates.

**Comparison Predicates:**

=: Equal to

<> or !=: Not equal to

<: Less than

<=: Less than or equal to

>: Greater than

>=: Greater than or equal to

**Logical predicates**

**And:** combines two or more conditions and returns true on if all the conditions are true

**Or:** combines two or more conditions and returns true on if any one of the conditions is true

**Not:** negates a conditions, ie., it give opposite result

**Between predicate:**

Test whether the values is in the specific range or not

SELECT \* FROM Orders WHERE OrderDate BETWEEN '2023-01-01' AND '2023-12-31';

**in predicate**

test weather the value is the list or not

SELECT \* FROM Customers WHERE Country IN ('USA', 'Canada', 'UK');

**is null predicate**

test the null values in the column

SELECT \* FROM Products WHERE Description IS NULL;

**like predicate**

search for the specific pattern in a column

**order of processing queries in SQL server**

1. FROM/JOIN --The FROM and/or JOIN clauses are executed first to determine the data of interest.

2. WHERE --The WHERE clause is executed to filter out records that do not meet the constraints.

3. GROUP BY--The GROUP BY clause is executed to group the data based on the values in one or more columns.

4. HAVING --The HAVING clause is executed to remove the created grouped records that don’t meet the constraints.

5. SELECT --The SELECT clause is executed to derive all desired columns and expressions.

6. ORDER BY--The ORDER BY clause is executed to sort the derived values in ascending or descending order.

7. LIMIT/OFFSET --Finally, the LIMIT and/or OFFSET clauses are executed to keep or skip a specified number of rows.