В зависимости от задачи, которую выполняет команда T-SQL, он может принадлежать к одному из следующих типов:

* **DDL** (Data Definition Language / Язык определения данных). К этому типу относятся различные команды, которые создают базу данных, таблицы, индексы, хранимые процедуры и т.д. В общем определяют данные.

В частности, к этому типу мы можем отнести следующие команды:

* + **CREATE**: создает объекты базы данных (саму базу даных, таблицы, индексы и т.д.)
  + **ALTER**: изменяет объекты базы данных
  + **DROP**: удаляет объекты базы данных
  + **TRUNCATE**: удаляет все данные из таблиц
* **DML** (Data Manipulation Language / Язык манипуляции данными). К этому типу относят команды на выбору данных, их обновление, добавление, удаление - в общем все те команды, с помощью которыми мы можем управлять данными.

К этому типу относятся следующие команды:

* + **SELECT**: извлекает данные из БД
  + **UPDATE**: обновляет данные
  + **INSERT**: добавляет новые данные
  + **DELETE**: удаляет данные
* **DCL** (Data Control Language / Язык управления доступа к данным). К этому типу относят команды, которые управляют правами по доступу к данным. В частности, это следующие команды:
  + **GRANT**: предоставляет права для доступа к данным
  + **REVOKE**: отзывает права на доступ к данным

В MS SQL Server по умолчанию создается четыре системных баз данных:

* **master**: эта главная база данных сервера, в случае ее отсутствия или повреждения сервер не сможет работать. Она хранит все используемые логины пользователей сервера, их роли, различные конфигурационные настройки, имена и информацию о базах данных, которые хранятся на сервере, а также ряд другой информации.
* **model**: эта база данных представляет шаблон, на основе которого создаются другие базы данных. То есть когда мы создаем через SSMS свою бд, она создается как копия базы model.
* **msdb**: хранит информацию о работе, выполняемой таким компонентом как планировщик SQL. Также она хранит информацию о бекапах баз данных.
* **tempdb**: эта база данных используется как хранилище для временных объектов. Она заново пересоздается при каждом запуске сервера.
* **Атрибуты и ограничения столбцов и таблиц**

### PRIMARY KEY. Первичный ключ уникально идентифицирует строку в таблице. В качестве первичного ключа необязательно должны выступать столбцы с типом int, они могут представлять любой другой тип.

### IDENTITY Атрибут **IDENTITY** позволяет сделать столбец идентификатором. Этот атрибут может назначаться для столбцов числовых типов INT, SMALLINT, BIGINT, TYNIINT, DECIMAL и NUMERIC. При добавлении новых данных в таблицу SQL Server будет инкрементировать на единицу значение этого столбца у последней записи. Как правило, в роли идентификатора выступает тот же столбец, который является первичным ключом, хотя в принципе это необязательно.

### UNIQUE Если мы хотим, чтобы столбец имел только уникальные значения, то для него можно определить атрибут **UNIQUE**.

### NULL и NOT NULL Чтобы указать, может ли столбец принимать значение NULL, при определении столбца ему можно задать атрибут **NULL** или **NOT NULL**. Если этот атрибут явным образом не будет использован, то по умолчанию столбец будет допускать значение NULL. Исключением является тот случай, когда столбец выступает в роли первичного ключа - в этом случае по умолчанию столбец имеет значение NOT NULL.

### DEFAULTАтрибут **DEFAULT** определяет значение по умолчанию для столбца. Если при добавлении данных для столбца не будет предусмотрено значение, то для него будет использоваться значение по умолчанию.

### CHECK Ключевое слово **CHECK** задает ограничение для диапазона значений, которые могут храниться в столбце. Для этого после слова CHECK указывается в скобках условие, которому должен соответствовать столбец или несколько столбцов.

### Оператор CONSTRAINT. Установка имени ограничений.

С помощью ключевого слова **CONSTRAINT** можно задать имя для ограничений. В качестве ограничений могут использоваться PRIMARY KEY, UNIQUE, DEFAULT, CHECK.

Ограничения могут носить произвольные названия, но, как правило, для применяются следующие префиксы:

* "PK\_" - для PRIMARY KEY
* "FK\_" - для FOREIGN KEY
* "CK\_" - для CHECK
* "UQ\_" - для UNIQUE
* "DF\_" - для DEFAULT

Для создания ограничения внешнего ключа на уровне столбца после ключевого слова **REFERENCES** указывается имя связанной таблицы и в круглых скобках имя связанного столбца, на который будет указывать внешний ключ. Также обычно добавляются ключевые слова **FOREIGN KEY**, но в принципе их необязательно указывать. После выражения REFERENCES идет выражение **ON DELETE** и **ON UPDATE**.

### ON DELETE и ON UPDATE

С помощью выражений **ON DELETE** и **ON UPDATE** можно установить действия, которые выполняться соответственно при удалении и изменении связанной строки из главной таблицы. И для определения действия мы можем использовать следующие опции:

* **CASCADE**: автоматически удаляет или изменяет строки из зависимой таблицы при удалении или изменении связанных строк в главной таблице.
* **NO ACTION**: предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице. То есть фактически какие-либо действия отсутствуют.
* **SET NULL**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение NULL.
* **SET DEFAULT**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение по умолчанию, которое задается с помощью атрибуты DEFAULT. Если для столбца не задано значение по умолчанию, то в качестве него применяется значение NULL.

## Операторы фильтрации

### Оператор IN Оператор **IN** позволяет определить набор значений, которые должны иметь столбцы:

|  |  |
| --- | --- |
| 1 | WHERE выражение [NOT] IN (выражение) |

### Оператор BETWEEN Оператор **BETWEEN** определяет диапазон значений с помощью начального и конечного значения, которому должно соответствовать выражение:

|  |  |
| --- | --- |
| 1 | WHERE выражение [NOT] BETWEEN начальное\_значение AND конечное\_значение |

Оператор LIKE Оператор **LIKE** принимает шаблон строки, которому должно соответствовать выражение.

|  |  |
| --- | --- |
| 1 | WHERE выражение [NOT] LIKE шаблон\_строки |

## Агрегатные функции

* **AVG**: находит среднее значение
* **SUM**: находит сумму значений
* **MIN**: находит наименьшее значение
* **MAX**: находит наибольшее значение
* **COUNT**: находит количество строк в запросе

### All и Distinct

По умолчанию все вышеперечисленных пять функций учитывают все строки выборки для вычисления результата. Но выборка может содержать повторяющие значения. Если необходимо выполнить вычисления только над уникальными значениями, исключив из набора значений повторяющиеся данные, то для этого применяется оператор **DISTINCT**.

|  |  |
| --- | --- |
| 1 | SELECT AVG(DISTINCT ProductCount) AS Average\_Price FROM Products |

## Операторы GROUP BY и HAVING

### GROUP BY Оператор **GROUP BY** определяет, как строки будут группироваться.

### Фильтрация групп. HAVING

Оператор **HAVING определяет, какие группы будут включены в выходной результат, то есть выполняет фильтрацию групп.**

## INNER JOIN

В прошлой теме было рассмотрено неявное соединение таблиц. Оно производилось на основе простой выборки неявно путем сведения данных. Для явного соединения данных из двух таблиц применяется оператор **JOIN**.

## OUTER JOIN

Перед оператором **JOIN** указывается одно из ключевых слов **LEFT**, **RIGHT** или **FULL**, которые определяют тип соединения:

* **LEFT**: выборка будет содержать все строки из первой или левой таблицы
* **RIGHT**: выборка будет содержать все строки из второй или правой таблицы
* **FULL**: выборка будет содержать все строки из обоих таблиц

## UNION

Оператор **UNION** подобно inner join или outer join позволяет соединить две таблицы. Но в отличие от inner/outer join объединения соединяют не столбцы разных таблиц, а два однотипных набора в один. Формальный синтаксис объединения:

## EXCEPT

Оператор EXCEPT позволяет найти разность двух выборок, то есть те строки которые есть в первой выборке, но которых нет во второй. Для его использования применяется следующий формальный синтаксис:

## INTERSECT

Оператор **INTERSECT** позволяет найти общие строки для двух выборок, то есть данный оператор выполняет операцию пересечения множеств. Для его использования применяется следующий формальный синтаксис:

--use Test -- використай БД Email

---- F5 Email

--SELECT Name as [Ім'я], Surname as 'Прізвище', Age as [Вік]

--FROM Employee

--SELECT [Name] + ' ' + Surname as [Full Name]

--FROM Employee

--Select \*

--From Employee

--Order by Salary Desc

--Select Name, Surname, Age, Salary

--From Employee

--Where Salary >= 30000 -- > < >= <= <> != !> !< = And Or

--Order by Age Asc -- Asc - по зростанню, значення по замовчуванню. Desc - по спаданню

--Select Name, Surname, Age, Salary, Position

--From Employee

--Where Salary >30000 And Age = 20

---- I'm comment

--use master

--Drop database Test

-- Drop, Create, Alter... Email

-- DDL - Data Defination Language

--Create database Test

use Test

-- check - перевірка даних при вставці

create table Employee(

Id int primary key identity,

Name nvarchar(20) not null default('empty'),

Surname nvarchar(30) not null,

DateofBirth date check(DateofBirth < getdate()),

Age int,

Position nvarchar(30) not null,

Salary money check(Salary > 0) default(4500),

Email nvarchar(30) unique,

-- Check (Email <> '')

);

-- Додати обмеження на Email

alter table Employee

add check(Email <> '')

-- Видаляю обмеження (констреінт) Check

alter table Employee

Drop Constraint CK\_\_Employee\_\_Email\_\_3B75D760

alter table Employee

add constraint CHK\_EmployeeEmail check(Email <> '')

insert into Employee values ('ivan', 'Korobov', '1997-5-13', year(getdate()) - 1997,'Admin', 15000, 'ivan2@ukr.net'),

('Olia', 'Ivanyshyna', '2000-7-12', year(getdate())- 2000, 'Designer', 18000, 'olia2@ukr.net'),

('Olia', 'Ivanyshyna', '2000-7-12', year(getdate())- 2000, 'Designer', 18000, 'olia@gmail.com');

select \* from Employee;

delete from Employee

where Id = 2;

delete from Employee

where Age = 22 And Name = 'Olia' --Email Like '%ukr.net'

use Test

GO

select \* from Employee

Go

-- DDL (create, drop, alter)

-- DML (insert, update, delete)

-- insert into table(...columns) values (...column values) - вставляємо тільки визначений набір колонок

-- insert into table values (...all column values) - вставляємо всі колонки визначені структурою таблиці

insert into Employee(Name, Surname, Age, Position, Email) values(N'Аня', N'Богуцька', 25, 'Counter', 'ann@gmai.com'),

(N'Олександр', N'Тимейчук', 30, N'Програміст', Null)

Go

insert into Employee values (N'Сергій', N'Паламарчук', '1997-5-5', 23 ,N'Архітектор', 25000,'ssss@gmail.com')

Go

delete from Employee -- якщо деліт виконати без where - то видаляємо всі записи в таблиці, але не перезапускаємо ID

where Id = 12 Or Id = 13

Go

truncate table Employee -- очищення таблиці і перезапуск ID з 1

Go

Update Employee

Set DateofBirth = '1998-6-23'

Where DateofBirth is null

Go

-- Update - оновити таблицю Set - встанови поле в значення value set column = value

Update Employee

Set Position = N'Архітектор',

Age = 33

Where Position = 'Admin'

Go

-- Distinct - без повторень

-- Top - перших N рядочків

Select Distinct Top(2) Position

From Employee

Order by Position Desc

/\*

1) Select Position

Distinct

Order by

Top

\*/

Select Name, Age

From Employee

--Where Age >= 23 And Age <= 30

Where Age Between 24 And 30 -- діапазон (значення включаються (>= <=))

Select Name, Surname, Position

From Employee

--Where Position = 'Counter' Or Position = 'Admin'

Where Position In ('Counter', 'Admin') -- входження в множину

Select \* From Employee

Where Email is not null -- не пусті емейли

Select \* from Employee

Where Name Like N'[АаОо]%'

-- Like: % - довільна к-сть символів

-- \_ - один символ

-- [Variants (aAoO)]

-- %a a% a%o%a (angora)

-- b%l\_ (blablabla)

use master

go

drop database University

go

create database University

go

use University

Go

--create table Teachers(

-- Id int identity primary key,

-- FullName nvarchar(40) not null,

-- BirthDate date not null,

-- Department nvarchar(40) not null,

-- Phone varchar(15) not null,

-- [Group] nvarchar(10) not null,

-- Subject nvarchar(30) not null

--)

--go

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Estrellita Lamburn', '1/9/1984', 'Human Resources', '332-533-4692', N'ПУ924', 'C#');

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Penni Wildman', '1/3/1984', 'Support', '820-789-8026', N'ПУ824', 'C++');

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Calli Heindl', '8/13/1998', 'Marketing', '136-805-7593', N'ПУ924', 'DP');

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Pamella McGee', '2/9/1991', 'Product Management', '734-585-8195', N'ПУ824', 'Math');

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Ham Wedderburn', '11/20/1974', 'Training', '982-642-3563', N'ПУ924', 'C++');

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Tonnie Mowsdale', '8/25/1986', 'Legal', '810-619-5969', N'ПУ924', 'Math');

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Hamlin Bonner', '1/6/1985', 'Support', '705-439-8203', N'ПУ824', 'C#');

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Rex Lillee', '1/2/1982', 'Engineering', '869-848-8316', N'ПУ824', 'DP');

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Bucky Dwyer', '8/27/1982', 'Services', '315-447-4737', N'ПУ924', 'TD');

--insert into Teachers (FullName, BirthDate, Department, Phone, [Group], Subject) values ('Elie Halden', '3/17/2000', 'Support', '597-190-9859', N'ПУ724', 'TD');

--go

--select \* from Teachers

create table Department(

Id int identity primary key,

Name nvarchar(30) not null unique,

Phone nvarchar(15) not null unique

);

go

create table Groups(

Id int identity primary key,

Name nvarchar(10) not null unique,

Id\_Department int foreign key references Department(Id)

);

go

create table Teacher(

Id int identity primary key,

Name nvarchar(20) not null,

Surname nvarchar(40) not null,

BirthDate date not null check (BirthDate < getdate()),

Id\_Department int foreign key references Department(Id)

);

go

create table Subject(

Id int identity primary key,

Name nvarchar(30) not null unique,

Id\_Department int foreign key references Department(Id)

);

go

create table TeachersGroups(

Id int identity primary key,

IdTeacher int foreign key references Teacher(Id),

IdGroup int foreign key references Groups(Id),

IdSubject int foreign key references Subject(Id)

);

go

insert into Department values('Software Development', '123-456'),('Design', '789-012'),('Admin', '345-678')

go

insert into Groups values ('Pr123', 1),('Pr456', 1), ('Pr980', 1), ('D123', 2),('D456', 2), ('A123', 3)

go

insert into Teacher values ('Iryna', 'Mertsalova', '1992-11-14', 1),('Andrii', 'Trofimchuk', '1984-10-21', 3),

('Valentyna', 'Benkalovych', '1991-10-12', 1), ('Igor', 'Pylypchuk', '1989-6-12', 2)

go

insert into Subject values ('C++', 1), ('C#', 1), ('DP', 1), ('Photoshop', 2), ('Illustrator', 2), ('security', 3), ('Network', 3)

go

insert into TeachersGroups values (1, 1, 1), (1, 2, 1), (2, 6, 7), (2, 1, 2), (4, 5, 4)

go

select\* from Department

select \* from Groups

select \* from Teacher

select \* from Subject

select \* from TeachersGroups

-- Вибрати всі групи і назви кафедр, на яких вони розміщені

select G.Name as 'Group Name', D.Name as 'Department', Phone

from Groups G, Department D

where Id\_Department = D.Id

-- Вибрати всіх викладачів, які працюють на кафедрі Розробки Програмного забезп.

select T.Name + ' ' + T.Surname as FullName, D.Name as Department

From Teacher T, Department D

Where D.Name = 'Software Development' and T.Id\_Department = D.Id

-- Вибрати всі предмети, які читають на кафедрі РПЗ і Адмінки

select S.Name as SubjectName, D.Name as Department

from Subject S, Department D

Where S.Id\_Department = D.Id And D.Id in (1, 3)

-- Вивести всі дані, про вчителів, предмети та групи в яких вони викладають

select T.Name + ' ' + T.Surname as FullName, G.Name as Groups, S.Name as Subject

From Teacher T, Subject S, Groups G, TeachersGroups TG

where TG.IdGroup = G.Id And TG.IdSubject = S.Id and TG.IdTeacher = T.Id

-- Вивести всі назви предметів і назви кафедр на яких викладає Андрій Трофімчук :)

Select T.Name + ' ' + T.Surname as FullName, S.Name as Subject, D.Name as Department

From Teacher T, Department D, Subject S, TeachersGroups TG

where TG.IdSubject = S.Id and TG.IdTeacher = T.Id and T.Id = 2 and S.Id\_Department = D.Id

/\*

-- якщо поле ID\_Department існує, але не є зовнішнім ключем

alter table Teacher

add Foreign key(Id\_Department) References Department(Id)

-- у видку, коли треба створити поле і зробити зовнішнім ключем

alter table Teacher

add Id\_Subject int foreign key references Subject(Id)

\*/

select \* from TeachersGroups

select \* from Teacher

update TeachersGroups

set IdTeacher=3

where IdTeacher is Null

delete from Teacher

where Id = 3

create table Student(

Id int identity primary key,

Name nvarchar(20),

Surname nvarchar(30),

IdGroup int foreign key references Groups(Id)

);

Go

create table Achievement (

Id int identity primary key,

IdStudent int foreign key references Student(Id),

IdSubject int foreign key references Subject(Id),

Mark int check (Mark between 1 and 12)

)

Go

select \* from Groups

select \* from Subject

insert into Student values('Katia', 'Marchuk', 1),

('Ann', 'Zabuzhko', 1),

('Roma', 'Zaluzhnyi', 2),

('Serhii', 'Denyshchuk', 3),

('Anatoliy', 'Popchuk', 3),

('Viktor', 'Palamarchuk', 6),

('Valia', 'Paniuk', 6)

insert into Achievement values (1, 1, Null),

(1, 2, 6),

(1, 3, 10),

(2, 1, 10),

(2, 2, Null),

(2, 3, 8),

(3, 1, 6),

(3, 2, 12),

(3, 3, Null)

select \* from Achievement

select \* from Student

insert into Student (Name, Surname, IdGroup)

select Surname, Name, IdGroup from Student

where Name Like '[Vv]%'

-- Агрегуючі функції: Count(column\_name), Avg(column\_name), Min(column\_name), Max(column\_name), Sum(column\_name)

-- Знайти к-сть студентів, в яких імена розпочинаються на літеру V

select count(\*) 'Count Students'

from Student

where Name like '[Vv]%'

-- Знайти к-сть студ. які навчаються в групі Pr123

select count(S.Id) [Stud in Pr123]

from Student S, Groups G

where S.IdGroup = G.Id And G.Name='Pr123'

insert into Achievement values (1, 4, Null)

--

-- Порахувати к-сть студентів, в яких є нездані предмети (NUll у таблиці Achievement)

select count(distinct A.IdStudent)

from Achievement A, Student S

where A.IdStudent = S.Id And A.Mark is Null

--

-- avg ignore null

select avg(Mark) as avg\_mark

from Achievement

-- знайти середню оцінку студента 1 (Катя Марчук)

select avg(Mark) Avg\_Mark

From Achievement A

where A.IdStudent = 1

-- середня оцінка з предмету С#

select avg(Mark) 'Avg C#'

From Achievement A, Subject S

Where A.IdSubject=S.Id And S.Name='C#'

-- Максимальна оцінка з предмету DP

select max(Mark) 'Max DP'

From Achievement A, Subject S

Where A.IdSubject=S.Id And S.Name='DP'

-- min mark in Group A123

select min(Mark) as MinMark

from Achievement A, Groups G, Student S

where S.IdGroup = G.Id And A.IdStudent = S.Id And G.Name = 'Pr123'

-- Наймолодший вчитель та найстарший

select max(DATEDIFF(year, BirthDate, getdate())) as 'Найстарший', min(DATEDIFF(year, BirthDate, getdate())) as 'Наймолодший'

From Teacher

select Sum(Mark) as SumaMark

From Achievement

-- Вибрати к-сть студентів в кожній групі

select G.Name, G.Id\_Department, count(S.Id)

From Student S, Groups G

where S.IdGroup = G.Id

Group by G.Name, G.Id\_Department

select \* from Achievement

select \* from Groups

select \* from Student

insert into Achievement values (6, 2, 4)

select S.Name, G.Name as Groups, Sb.Name Subject, A.Mark

from Student S, Groups G, Achievement A, Subject Sb

where S.Id = A.IdStudent and G.Id = S.IdGroup and Sb.Id = A.IdSubject

-- вивести назви груп і мінімальну оцінку в групі

select G.Name, min(A.Mark) as Mark

from Groups G, Achievement A, Student S

where G.Id = S.IdGroup and S.Id = A.IdStudent and Mark = (select min(Mark) from Achievement)

group by G.Name

select G.Name, min(A.Mark) as Mark

from Groups G, Achievement A, Student S

where G.Id = S.IdGroup and S.Id = A.IdStudent

group by G.Name

Having min(A.Mark) > 5

select \* from Teacher

-- Вибрати групи місяців, в яких народилось найбільше вчителів

select BirthMonth, Max(CountTeacher) as MaxCountTeacher

From

(

--1) Вибираємо місяць та кількість викладачів, що народились в кожному місяці

select Month(BirthDate) as BirthMonth, Count(Id) as CountTeacher

From Teacher

Group by Month(BirthDate)) as ResultTable

-- обмеження на рядки, щоб к-сть викладачів дорівнювала максимальній к-сті в групах (у внутрішній таблиці)

where CountTeacher = (

-- 3) Вибираємо максимальну кількість викладачів по внутрішній таблиці Тemp

select Max(CountTeacher)

from

--2) Групуємо місяць та кількість викладачів, що народились в кожному місяці

(select Month(BirthDate) as BirthMonth, Count(Id) as CountTeacher

From Teacher

Group by Month(BirthDate)) as Temp)

group by BirthMonth

select \* from Student

-- Вибрати всіх студентів, про які не існує запис у таблиці Achievement

select S.Name, S.Surname

From Student S

Where Not Exists (Select \*

From Achievement

Where Achievement.IdStudent = S.Id)