Polecenia SQL

1. SELECT

Polecenie wybierające np. SELECT * FROM persons;

SELECT wybierz

FROM z skąd wybieramy

2. WHERE

Warunek, który musi być spełniony dla wybieranych danych np. SELECT name FROM persons WHERE name = ‘Michał’

Po słowie kluczowym WHERE podajemy warunek dla zapytania

3. AND, OR, NOT

Polecenia, które wykorzystujemy w warunkach do łączenia ich w pewnym sensie np.

SELECT * FROM persons WHERE name = ‘Michał’ AND city = ‘Warszawa’ OR city = ‘Gdańsk’

SELECT * FROM persons WHERE NOT name = ‘Michał’

4. ORDER BY

Słowo kluczowe **ORDER BY** służy do sortowania wyników zapytania w kolejności rosnącej lub malejącej na podstawie jednej lub kilku kolumn.

Do określania kolejności sortowania wykorzystywane są słowa kluczowe:

- **ASC** - rosnąco
- **DESC** – malejąco

Domyślnie rekordy są sortowane rosnąco

Np.
SELECT nazwisko, numer_telefonu
FROM ksiazka_telefoniczna
WHERE miasto = 'Warszawa'
ORDER BY nazwisko DESC, numer_telefonu ASC

5. INSERT INTO

Służy do wstawiania rekordów np.

INSERT INTO persons (id, name, surname, city)
VALUES (1, ‘Michał’, ‘Nieważne’, ‘Warszawa’)

Jeśli podamy dane do wstawienia w kolejności takiej jakie są kolumny w tabeli można pominąć podawanie kolumn np.

INSERT INTO persons
VALUES (1, ‘Michał’, ‘Nieważne’, ‘Warszawa’)

6. NULL

Jest to słowo kluczowe, które podajemy w warunku i pole może być puste lub niepuste np.

SELECT * FROM persons WHERE city IS NULL

SELECT * FROM persons WHERE city IS NOT NULL

7. UPDATE

Służy do aktualizacji danych w tabeli np.

UPDATE persons
SET name = ‘Oliwia’, surname = ‘Ładna’, city = ‘Warszawa’

WHERE id = 2

8. DELETE

Służy do usuwania rekordów z tabel np.

SELETE FROM persons WHERE id = 5

9. LIMIT

Ogranicza ilośc wyswietlonych rekordów np.

SELECT * FROM name LIMIT 5

Wyświetli 5 rekordów z tabeli name

10. MIN, MAX

Wybiera minimalna lub maksymalna wartość np.

SELECT MIN(cena)
FORM opony
WHERE rozmiar = 19

11. COUNT, AVG, SUM

COUNT zwraca liczbę wierszy do wyświetlenia np.

SELECT COUNT(id)
FROM persons

AVG zwraca średnią wartość

SELECT AVG(cena)
FROM opony

SUM zwraca sumę wartości

SELECT SUM(cena)
FORM opony

12. LIKE

LIKE używa się w WHERze aby sprawdzić czy coś w rekordzie jest jak nasz wzorzec np.

SELECT cena FROM opony WHERE nazwa LIKE ‘Miche%’

W konstruktorze LIKE są operatory

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

13. IN

Po polsku “jest w” np.

SELECT surname FROM persons WHERE city IN (‘Warszawa’, ‘Gdańsk’, ‘Bździszewo’)

Wyświetli nazwiska z tabeli osoby gdzie miasto jest w Warszawa lub Gdańsk lub Bździszewo

Można użyc NOT IN co będzie odwrotnością.

W IN można napisać SELECT’a potocznie się to nazywa SELECT w SELECT’ie np.

```
SELECT surname
 FROM persons
WHERE city IN (
            SELECT city
            FROM persons
            WHERE wojewodztwo = ‘Mazowieckie’
)
```

14. BETWEEN

Zwraca wartości, które mieszczą się w podanych wartościach np.

SELECT name, surname FROM persons WHERE old BETWEEN 18 AND 30

15. AS

Alias to przypisanie innej nazwy wyświetlanej kolumny. Robi się to po to, aby mniej pisać lub było bardziej wiadomo co wyświetlamy np.

SELECT surname AS nazwisko FROM persons WHERE id = 5

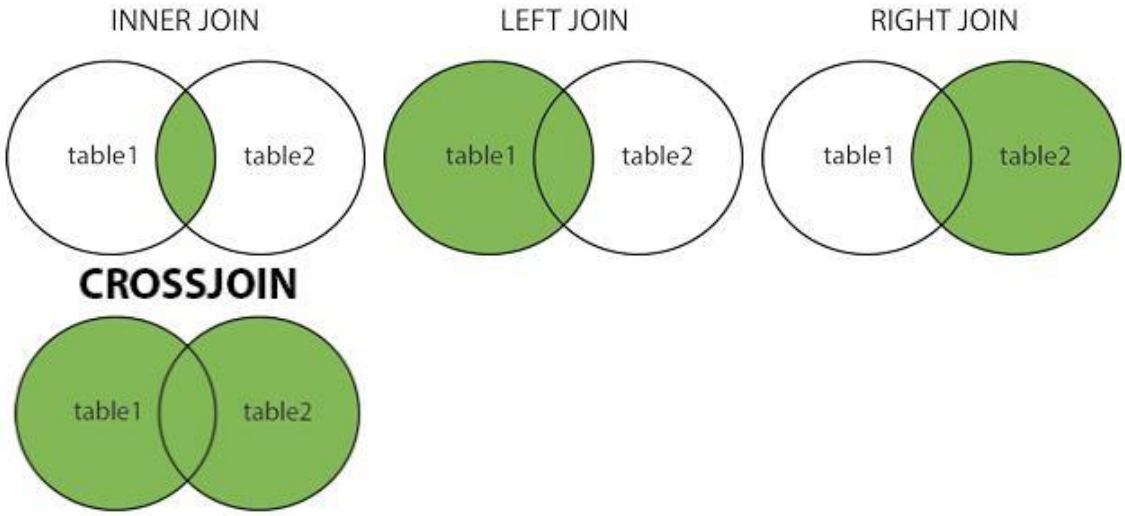
Mało kto wie alias można nadać bez słowa kluczowego AS np.

SELECT surname nazwisko FROM persons WHERE id = 5

Baza nadal będzie wiedziała, że ma wyświetlić kolumnę surname jako nazwisko.

16. JOIN

Służy do połączenia wartości tabel. Wyróżniamy kilka rodzajów



17. INNER JOIN

Łączy rekordy występujące w obu tabelach po kluczu łączenia, czyli łączy wspólne elementy np.

	Id	Imie	Nazwisko	DataUrodzenia	Pesel	Plec
1	1	Marcin	Brzeczyszczykiewicz	1980-05-05	80050517439	0
2	2	Krystyna	Mogila	1977-02-24	77022411943	1
3	3	Zenon	Cwaniak	1956-11-12	56111214599	0

	Id	IdKlienta	Ulica	NrDomu	NrLokalu	Miasto	KodPocztowy	RodzajAdresu
1	3	1	Miazgowa	14	5	Czeburkowa	00-800	0
2	4	1	Miazgowa	14	5	Czeburkowa	00-800	1
3	7	2	Cisowa	7A	NULL	Jajcowa	00-100	0
4	8	2	Brzozowa	141	9	Misiowa	00-400	1

```
SELECT * FROM dbo.Klienci AS K
INNER JOIN dbo.Adresy AS A ON K.Id = A.IdKlienta
```

Wynik:

Results Messages

	Id	Imie	Nazwisko	DataUrodzenia	Pesel	Plec	Id	IdKlienta	Ulica	NrDomu	NrLokalu	Miasto	KodPocztowy	RodzajAdresu
1	1	Marcin	Brzeczyszczykiewicz	1980-05-05	80050517439	0	3	1	Miazgowa	14	5	Czeburkowa	00-800	0
2	1	Marcin	Brzeczyszczykiewicz	1980-05-05	80050517439	0	4	1	Miazgowa	14	5	Czeburkowa	00-800	1
3	2	Krystyna	Mogila	1977-02-24	77022411943	1	7	2	Cisowa	7A	NULL	Jajcowa	00-100	0
4	2	Krystyna	Mogila	1977-02-24	77022411943	1	8	2	Brzozowa	141	9	Misiowa	00-400	1

● Jest to złączenie wewnętrzne i jako wynik otrzymujemy tylko dopasowane wiersze.
● Stosując samego JOIN'a SZBD domyślnie stosuje INNER, czyli typ złączenia wewnętrznego.
● W złączeniu tego typu nie ma znaczenia kolejność tabel, w każdym przypadku wynik będzie tożsamy.
● Kolejność warunków umieszczonych w ON także nie ma znaczenia.

18. LEFT JOIN

● Jest to złączenie zewnętrzne lewostronne.
● W wyniku zwracane są wszystkie wiersze występujące w tabeli po lewej stronie.
● Do powyższego wyniku dopasowywane są wiersze z prawej tabeli.
● W przypadku braku połączenia otrzymujemy wartości puste, czyli NULL'e.

19. RIGHT JOIN

● Jest to złączenie zewnętrzne prawostronne.
● W wyniku zwracane są wszystkie wiersze występujące w tabeli po prawej stronie.
● Do powyższego wyniku dopasowywane są wiersze z lewej tabeli.
● W przypadku braku połączenia otrzymujemy wartości puste, czyli NULL'e.

20. FULL JOIN

● Jest to złączenie zewnętrzne pełne.
● W wyniku zwracane są wszystkie wiersze wewnętrznie dopasowane.
● Do powyższego wyniku dopisywane są wszystkie niepołączone wiersze z obydwu tabel uzupełnione o wartości puste, czyli NULL'e.

21. UNION

Ten operator łączy wynik dwóch zapytań do jednej tabeli np.

```
SELECT * FROM sprzedaz2019
UNION
SELECT * FROM sprzedaz2021
```

22. GROUP BY

Służy do grupowania wyniku zapytania po wartościach np.

```
SELECT COUNT(id), city
FROM persons
GROUP BY city
```

Wyświetli liczbę rekordów do miast które są wpisane do tabeli np. że w warszawie jest 5 osób.

23. HAVING

Służy do filtrowania grup najlepiej będzie to pokazane na przykładzie

```
select CIty, COUNT(CustomerID) as CustQty
from dbo.Customers
WHERE Country = 'Brazil'
GROUP BY City
```

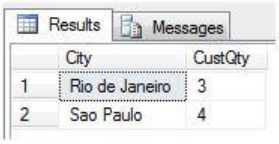
Results | Messages

	CIty	CustQty
1	Campinas	1
2	Resende	1
3	Rio de Janeiro	3
4	Sao Paulo	4

Filtrowanie w HAVING, polega na filtrowaniu całych grup rekordów. Zgodnie z zasadą opisaną na początku artykułu, możemy filtrować po kolumnach grupujących lub pozostałych, za pośrednictwem funkcji agregujących. W tym momencie, chcemy właśnie filtrować grupy rekordów, ze względu na ilość elementów (liczby klientów) w ich ramach.

Cel ten zrealizuje filtrowanie za pomocą HAVING. Filtrem będzie wynik funkcją agregującej COUNT(), wybierający tylko te grupy, dla których ilość wierszy (Klientów) będzie większa od 1.

```
select City, COUNT(CustomerID) as CustQty
from dbo.Customers
WHERE Country = 'Brazil'
GROUP BY City
HAVING COUNT(CustomerID)>1
```



Results | Messages

	City	CustQty
1	Rio de Janeiro	3
2	Sao Paulo	4

24. EXIST

Operator ustala, czy pożądane przez nas dane są w wyniku podzapytania np.

SELECT NazwaUżytkownika as 'Nazwa użytkownika'
FROM Użytkownicy
WHERE EXISTS
(SELECT *
FROM Transakcje
WHERE Użytkownicy.IDUżytkownika = Transakcje.IDUżytkownika)

25. ANY, ALL

ANY – sprawdza wartość dowolnego wiersza zwróconego przez podzapytanie

ALL – sprawdza wartości wszystkich wierszy zwróconych przez podzapytanie.

26. CASE

CASE działa na takiej zasadzie, że dla każdego wiersza zwracanego w wyniku zapytania sprawdza warunek logiczny i w zależności od wyniku wypisuje komunikat podany po słowie kluczowym THEN np.

```
SELECT imie, nazwisko, wiek,
CASE
WHEN wiek < 30 THEN 'Młodszy developer'
WHEN wiek > 30 THEN 'Starszy developer'
ELSE 'Pracownik ma 30 lat'
END
FROM pracownicy
```

Po THEN zawsze następuje działanie jeśli prawda

27. CREATE

Za pomocą tego polecenia stworzymy bazę danych np.

```
CREATE DATABASE przykladowabaza
```

28. DROP

Za pomocą tego polecenia usuwamy bazę danych np.

```
DROP DATABASE przykladowabaza
```

29. CREATE TABLE

Za pomocą tego polecenia tworzymy tabele w bazie danych np.

```
CREATE TABLE persons (
id int,
name varchar(30),
surname varchar(30)
)
```

30. DROP TABLE

Za pomocą tego polecenia usuwamy tabelę wraz z danymi np.

```
DROP TABLE persons
```

31. TRUNCATE

To polecenie usuwa dane z tabeli, ale sama tabela i jej struktura zostaje np.

```
TRUNCATETABLE persons
```

32. ALTER TABLE

Służy do modyfikacji struktury tabeli i wyróżniamy:

- ADD np.

ALTER TABLE model
ADD nazwa varchar(30)

-DROP np.

ALTER TABLE model
DROP nazwa

-MODIFY np.

ALTER TABLE model
MODIFY nazwa varchar(255)

33. COINSTAINT

Inaczej są to reguły I ograniczenia. Wyróżniamy następujące:

- NOT NULL (pole nie może być puste)

- UNIQUE (wartości w kolumnie muszą być różne, inaczej mówiąc nie mogą się powtarzać)

- PRIMARY KEY (klucz podstawowy, kombinacja dwóch poprzednich ograniczeń)

- FOREIGN KEY (klucz obcy)

- CHECK (sprawdza czy wartości w kolumnie są poprawne)

-DEFAULT (ustawia wartość domyślną jeśli nic nie jest podane)

- CREATE INDEX (tworzy index. Dzięki indexowaniu baza może działać szybciej)

Np.

```
CREATE TABLE opony (
Id int NOT NULL,
Nazwa varchar(255) NOT NULL,
Marka int,
Rozmiar int,
Opinia varchar (255) DEFAULT ‘brak opinii’,
PRIMARY KEY (Id)
FOREIGN KEY (marka) REFERENCES marka(id),
CHECK (Rozmiar<13)
)
```

34. AUTO INCREMENT

To polecenie automatycznie może nadać numer. Stosujemy to np. przy kluczu podstawowym, gdzie jest to id i po prostu każdy następny rekors ma następną liczbę np.

```
CREATE TABLE Persons(
ID int NOT NULL AUTO_INCREMENT,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (ID)
)
```

35. CREATE USER

Możemy utworzyć użytkownika np.

CREATE USER ‘nowy’@’localhost’ IDENTIFYED BY ‘Admin123’

Tworzymy użytkownika nowy na localhoscie który ma hasło Admin123

36. DROP USER

Usuwa użytkownika np.

DROP USER ‘nowy’

37. GRANT

Służy do nadawania uprawnień np.

GRANT ALL PRIVILEGES ON *.* TO ‘nowy’

Nadajemy wszystkie uprawnienia na wszystkich bazach oraz tabelach (*.*).

!!!Pierwsza * jest przeznaczona dla baz danych druga natomiast dla tabel!!!

Permisje:

- ALL PRIVILEGES (wszystkie uprawnienia)

- CREATE (zezwala na tworzenie nowych baz i table)

- DROP (zezwala na usuwanie baz i table)

- DELETE (zezwala na usuwanie rekordów)

- INSERT (zezwala na wprowadzanie rekordów)

- UPDATE (zezwala na modyfikacje rekordów)

- GRANT OPTION (zezwala na dodawanie i usuwanie uprawnień innym użytkownikom)

38. LOGOWANIE

Aby się zalogować na użytkownika wpisujemy następujące polecenie w konsoli

mysql –u username –p

gdzie username to nazwa użytkownika np.

mysql –u nowy –p

!!!domyślnie root nie ma hasła, więc nie używamy parametru –p podczas logowania na roota!!!

39. REVOKE

Służy do odbierania uprawnień np.

REVOKE SELECT ON *.* FROM ‘nowy’

40. ROLE

CREATE ROLE nazwa_roli (dodaje role)

DROP ROLE nazwa_roli (usuwa role)

!!!Na rolach można również nadawać uprawnienia!!!

GRANT ROLE rola ON *.* TO ‘nowy’@’localhost’ (nadanie roli użytkownikowi)

REVOKE ROLE rola FROM ‘nowy’@’localhost’ (usunięcie roli z użytkownika)

Wykorzystane źródła:

http://sqlszkolenia.pl/aktualnosci/18-jak-dziala-join-w-sql-server
https://www.sqlpedia.pl/having-filtrowanie-grup/
https://www.online-project.pl/typy-zlaczen-tabel-join-inner-outer-left-right-full-cross-self-apply/