# SQL GEEKON

## Typy danych

| | |
|---|---|
| BIGINT | reprezentuje liczby całkowite z zakresu od -9 223 372 036 854 775 808 do 9 223 372 036 854 775 807 |
| INT | reprezentuje liczby całkowite z zakresu od 2 147 483 648 do 2 147 483 647 |
| DECIMAL | reprezentuje liczby o określonej skali i precyzji -1038 + 1 do 1038 – 1 |
| DATETIME | reprezentuje datę i czas |
| DATE | reprezentuje datę z dokładnością do jednego dnia |
| TIME | reprezentuje czas i pozwala określić jego dokładność (maksymalna dokładność to 100 nanosekund) |
| CHAR | reprezentuje ciąg znaków o określonej długości |
| NCHAR | reprezentuje ciąg znaków o określonej długości zapisanej w postaci Unicode |
| VARCHAR | reprezentuje ciąg znaków o zmiennej długości |
| NVARCHAR | reprezentuje ciąg znaków o zmiennej długości zapisanej w postaci Unicode |

## Kolejność wykonywania zapytania

| Kolejność bez UNION | Kolejność z UNION |
|---|---|
| FROM, JOIN | FROM, JOIN |
| WHERE | WHERE |
| GROUP BY i funkcje agregujące | GROUP BY i funkcje agregujące |
| HAVING | HAVING |
| SELECT | TOP |
| ORDER BY | UNION i SELECT |
| TOP | ORDER BY |

## Modyfikowanie danych

Wiersze w tabeli możemy dodawać za pomocą instrukcji **INSERT**

```
INSERT INTO nazwa_tabeli [(kolumna1, kolumna2, …)
VALUES (wartość1, wartość2, …)
```

Wstawiane dane muszą uwzględniać ograniczenia nałożone na tabelę.

Instrukcja **DELETE** usuwa wybrane wiersze tabeli.

```
DELETE FROM nazwa_tabeli [FROM warunki]
```

Próba usunięcia wierszy, które są powiązane z wierszami w innych tabelach, może skończyć się niepowodzeniem. W takim przypadku należy usunąć najpierw odpowiednie wiersze z powiązanych tabel.

Za pomocą instrukcji **UPDATE** możemy aktualizować istniejące dane.

```
UPDATE nazwa_tabeli
SET kolumna1 = wartość1, kolumna2 =
wartość2, …
[WHERE warunki]
```

## Pobieranie danych

Instrukcja **SELECT** służy do pobierania danych z bazy.

```
SELECT [ALL | DISTINCT] [TOP n [PERCENT]]
```

| | |
|---|---|
| ALL | (wartość domyślna) oznacza, że zostaną zwrócone wszystkie wybrane wiersze |
| DISTINCT | oznacza, że z wyniku zostaną usunięte duplikaty wybranych wierszy |
| TOP | oznacza, że zostanie zwrócona tylko określona ilość wierszy |
| PERCENT | pozwala na względne określenie liczby zwracanych wierszy. Jeżeli użyto znacznika PERCENT, wartość n musi należeć do zakresu <0, 100> |

Klauzula **FROM** określa, z których obiektów będą pobierane dane źródłowe

Klauzula **WHERE** określa, które wiersze zostaną zwrócone przez zapytanie (filtrowanie).

## Wybieranie danych na podstawie warunków różnego rodzaju.

Język SQL nie nakłada żadnych ograniczeń na liczbę argumentów wyszukiwania wymienionych w klauzuli **WHERE**.

**Operatory porównania** – jeżeli chcemy sprawdzić, czy wartość zapisana w tabeli odpowiada wartości warunków wyszukiwania. Dostępne operatory porównania:
=, !=, <, <=, >, >=

**Operator LIKE** – jeżeli chcemy sprawdzić zgodność tekstu z podanym wzorcem. Operator LIKE dopuszcza stosowanie następujących symboli

| | |
|---|---|
| % | odpowiada dowolnemu ciągowi znaków o długości równej 0 lub większej |
| _ | odpowiada dowolnemu pojedynczemu znakowi |
| [] | odpowiada pojedynczemu znakowi z określonego zbioru lub zakresu |
| [^] | odpowiada pojedynczemu znakowi nienależącego do określnego zbioru lub zakresu |

**Operatory logicznych** – jeżeli chcemy połączyć kilka warunków wyszukiwania.

| | |
|---|---|
| AND | koniunkcja. Warunek jest spełniony jeśli wszystkie warunki są spełnione |
| OR | alternatywa. Warunek jest spełniony jeśli choć jeden z warunków jest spełniony |

| | |
|---|---|
| NOT | zaprzeczenie. Zamienia wynik warunku na odwrotny |

**Operator BETWEEN** – jeżeli chcemy sprawdzić, czy wartość zapisana w tabeli należy (lub nie) do określonego przedziału zamkniętego.

**Operator IN** – jeżeli chcemy sprawdzić czy wartość zapisana w tabeli należy do określonego zbioru.

**Operator IS NULL** – jeżeli chcemy sprawdzić, czy wartość zapisana w tabeli jest wartością nieokreśloną

### Łączenie zbiorów danych

W klauzuli **FROM** możemy odwołać się do dowolnej liczby obiektów – tabel, widoków, funkcji tabelarycznych czy wyników podzapytań. Z reguły łączy się obiekty na podstawie wartości wspólnego atrybutu, na przykład wartość pary klucz podstawowy – klucz obcy.

| | |
|---|---|
| CROSS JOIN | określa, że wynik będzie obliczony jako iloczyn kartezjański |
| INNER JOIN | określa, że wynik będzie obliczony jako złączenie naturalne złączonych tabel. Wynikiem złączenia naturalnego są tylko te wiersze, dla których w użytych do powiązania kolumnach zapisane były te same wartości |
| LEFT OUTER JOIN | określa, że wynik będzie zawierał wszystkie wiersze z lewej tabeli, a z prawej tabeli wyłącznie te wiersze, dla których w użytych do powiązania kolumnach zapisane były te same wartości |
| RIGHT OUTER JOIN | określa, że wynik będzie zawierał wszystkie wiersze z prawej tabeli, a z lewej tabeli wyłącznie te wiersze, dla których w użytych do powiązania kolumnach zapisane były te same wartości |
| FULL OUTER JOIN | określa, że wynik będzie zawierał wszystkie wiersze z obu złączonych tabel |

### Klucze

**Primary key** definiuje klucz główny tabeli. Tabela może mieć tylko jeden klucz główny, ale w jego skład może wchodzić wiele kolumn.

**Własności klucza głównego**
**1 obowiązkowy** – kolumny nie mogą zawierać wartości NULL
**2 unikalny** – wartość nie może się powtarzać
**3 minimalny** – będzie składał się z minimalnej ilości kolumn

**Foreign key** w wyniku nadania klucza obcego ograniczamy listę dopuszczalnych dla kolumny wartości do wartości przechowywanych w powiązanej z nią kolumnie w innej tabeli. Dodatkowo w tabeli tej musi być utworzona odpowiednia kolumna (lub kolumny) tego samego rodzaju oraz musi być nałożone jedno z dwóch ograniczeń: PRIMARY KEY lub UNIQUE.

### Klauzula GROUP BY

określa grupy wierszy, dla których będą obliczane wyniki funkcji grupujących. Na liście argumentów klauzuli GROUP BY muszą znaleźć się wszystkie wyrażenia niebędące wynikiem funkcji grupującej wymienione w poleceniu SELECT.

### Klauzula HAVING

określa warunki filtrowania dla grup lub funkcji grupujących. W klauzuli HAVING można wykorzystać wszystkie typy warunków z klauzuli WHERE.

### Funkcje agregujące

| | |
|---|---|
| COUNT() | funkcja zwraca liczbę wystąpień wartości wyrażeń różnych od NULL |
| SUM() | funkcja zwraca sumę wszystkich argumentów wywołania. W przeciwieństwie do funkcji COUNT, która działa dla wszystkich typów danych, funkcja SUM działa tylko dla argumentów liczbowych |
| AVG() | funkcja zwraca wartość średnią przekazanych argumentów. Argumentami funkcji AVG muszą być dane liczbowe |
| MIN() i MAX() | funkcja MIN służy do znajdowania wartości najmniejszej w zbiorze wartości, a funkcja MAX największej. Obie funkcje, podobnie jak funkcja COUNT, mogą być użyte do różnych typów danych |

### Widoki

Widok to wirtualna tabela pobierająca dane z określonych tabel lub innych źródeł danych. Tabele bazowe muszą być wcześniej utworzone niż widok. Widok umożliwia jedynie odczyt danych, oraz nie działa w nim ORDER BY (sortowanie danych).

```
CREATE VIEW nazwa_widoku
AS
Instrukcje
```

Za pomocą instrukcji **ALTER VIEW** możemy zmienić definicję widoku. Niepotrzebny widok usuwamy za pomocą instrukcji
**DROP VIEW**.