Polecenie SELECT

- instrukcja pobierająca dane z bazy danych (z tabel, widoków)
- użytkownik posługujący się nią musi mieć uprawnienia do pobierania danych
- wynikiem zapytania jest zawsze tablica o określonych kolumnach i wierszach, spełniających nałożone warunki
- wiersze zwracane przez zapytanie mogą się powtarzać

SELECT [DISTINCT] lista kolumn, wyrażeń, funkcji

FROM lista tabel, widoków

WHERE warunek logiczny, określa wybierane wiersze

GROUP BY lista kolumn, wg których odbywać się będzie grupowanie

HAVING warunek logiczny, określa wybierane grupy

ORDER BY lista kolumn **[ASC/DESC]**

Wybór wszystkich danych:

```
SELECT * FROM employee
```

Wybór określonych kolumn lub wyrażeń:

```
SELECT first_name, last_name, salary
FROM employee
```

```
SELECT emp_no, last_name, 0.2*salary+200 as bonus
FROM employee
```

- **DISTINCT** – eliminuje powtarzające się wiersze z wyniku zapytania

```
SELECT DISTINCT job_country FROM employee
```

- **wyrażenia** składają się z kolumn, **operatorów**, stałych i funkcji
- **operatory**: **+, -, *, /**
- kolejność wykonywania działań jest standardowa, jeżeli zaistnieje potrzeba, można używać **nawiasów**

- **warunek logiczny (w klauzulach WHERE / HAVING)**

  przyjmuje wartości TRUE, FALSE, NULL (NULL oznacza wartość pustą)

  warunek może mieć postać:

    - wartość ***operator relacji*** wartość
    - ***operatory relacji*** **< , > , <= , >= , = ,<>**
    - wartość **IN** (lista wartości)
    - wartość **BETWEEN** wartość1 **AND** wartość2

- warunki można łączyć za pomocą ***operatorów logicznych:***

  AND && OR || NOT ! XOR

- porównywanie fragmentów łańcuchów
    - tekst **LIKE** '_wzorzec%'
- **SELECT** kolumna **AS** alias ... - utworzenie **aliasu** wyświetlanej kolumny (lub „alias nazwy kolumny")
- **aliasy** nazw tabel – **SELECT** * **FROM** tabela **alias_tab**
- tabela.kolumna – nazwa tabeli jako kwalifikator identyfikujący pole
- alias_tab.kolumna – alias nazwy tabeli jako kwalifikator pola

- sprawdzenie, czy wartość jest (nie jest) wartością pustą **NULL**
    - wartość **IS NULL** wartość **IS NOT NULL**

klauzula **ORDER BY**

- sortowanie danych w porządku rosnącym – **ASC** (domyślnie)
- sortowanie danych w porządku malejącym – **DESC**

- **funkcje agregacji**
    - **COUNT**(*) – zlicza wiersze
    - **SUM**(kol) - sumuje wartości w kolumnie //operuje na danych liczbowych
    - **AVG**(kol) - liczy średnią w kolumnie //operuje na danych liczbowych
    - **MIN**(kol) - minimum
    - **MAX**(kol) - maksimum
    - jeżeli jako argument funkcji agregacji podamy (**DISTINCT** nazwa kolumny), to uwzględnione zostaną tylko unikalne wartości; można jako argument funkcji podać także wyrażenie (odpowiedniego typu)
    - COUNT(kolumna) zlicza niepuste wartości
    - COUNT(DISTINCT kolumna) policzy ile jest różnych, niepustych, wartości w kolumnie.

```
SELECT COUNT(*), SUM(SALARY)
FROM employee
WHERE dept_no='600'
```

```
SELECT AVG(DISTINCT salary)
FROM employee
```

```
SELECT dept_no, COUNT(full_name), AVG(salary)
FROM employee
GROUP BY dept_no
```

```
SELECT MIN(LOWER(job_code)), dept_no
FROM employee
GROUP BY 2
```

- klauzula **GROUP BY** : **grupowanie względem podanych kolumn** (można podać kilka) lub wyrażeń – jedną grupę tworzą te wiersze, których wartości w wymienionych kolumnach są wspólne
  - zapytanie zwróci tyle wierszy, ile jest różnych wartości w kolumnach, wg których grupujemy;
  - stosowana w połączeniu z funkcjami agregacji, do wyliczenia wartości funkcji w poszczególnych grupach
  - na liście pobieranych danych w klauzuli SELECT mogą pojawić się tylko:
    - nazwy kolumn lub wyrażenia, względem których grupujemy (te, które są w klauzuli **GROUP BY)**,
    - wyrażenia skonstruowane z użyciem kolumn (wyrażeń), wzgl. których grupujemy,
    - stałe,
    - **funkcje agregujące** dotyczące innych kolumn;
  - w GROUP BY można podać pozycję kolumny na liście SELECT, zamiast jej nazwy (dla niektórych serwerów, np. MySQL, Firebird)
  - jeżeli w zapytaniu jest klauzula **ORDER BY** lub **WHERE**, to można w niej umieścić tylko kolumny (wyrażenia), wzgl. których grupujemy

```
SELECT COUNT(*), AVG(SALARY), dept_no
FROM employee
GROUP BY dept_no
```

```
SELECT COUNT(DISTINCT dept_no), SUM(SALARY), job_country
FROM employee
GROUP BY 3
ORDER BY SUM(SALARY)
```

```
SELECT SUM(SALARY), job_country, dept_no
FROM employee
GROUP BY job_country, dept_no
ORDER BY SUM(SALARY)
```

```
SELECT SUM(SALARY), job_country, dept_no
FROM employee
WHERE hire_date>='1990.10.24'
GROUP BY job_country, dept_no
ORDER BY SUM(SALARY) DESC
```

- klauzula **HAVING** służy do wybierania grup, spełniających pewien warunek logiczny, określony **tylko na kolumnach (wyrażeniach), wzgl. których grupujemy**
  - występuje tylko w połączeniu z **GROUP BY**
  - w warunku po klauzuli **HAVING**, dotyczącym wyboru grup, mogą wystąpić funkcje agregacji
  - jeżeli w zapytaniu mamy też warunek logiczny dotyczący wierszy, to powinien być w **WHERE**

```
SELECT COUNT(*), AVG(SALARY), dept_no
FROM employee
WHERE hire_date>='1999.06.12'
GROUP BY dept_no
HAVING SUM(salary)>=30000 AND COUNT(*)>=3
ORDER BY AVG(salary) DESC
```