Rozdział 2
Język bazy danych - SQL

Polecenie SELECT, klauzula WHERE, operatory SQL, funkcje znakowe, liczbowe, daty, konwersji, polimorficzne, grupowe, podział relacji na grupy, klauzule GROUP BY i HAVING,

Wprowadzenie do języka SQL

- Język dostępu do bazy danych
- Język deklaratywny, zorientowany na przetwarzanie zbiorów
- Grupy poleceń języka:
  - DML (ang. Data Manipulation Language)
  - DDL (ang. Data Definition Language)
  - DCL (ang. Data Control Language)
- Polecenie SQL może być zapisane:
  - w jednym bądź wielu wierszach
  - dużymi lub małymi literami
- Polecenie SQL zawsze kończymy średnikiem

```
SELECT *
FROM pracownicy;
```

```
select * from
PRACOWNICY;
```


Projekcja

Wybór wartości określonych atrybutów relacji

```
SELECT nazwisko, etat
FROM pracownicy;
```

```
SELECT id_prac, placa_pod,
zatrudniony, id_szefa
FROM pracownicy;
```

Wyrażenia arytmetyczne

Operatory arytmetyczne

– +, -, *, /

```
SELECT nazwisko, placa_pod*12,
placa_dod+200
FROM pracownicy;
```

```
SELECT etat, placa_pod/30,
(placa_pod+100)*12
FROM pracownicy;
```


Aliasy nazw atrybutów relacji

- Alias to alternatywna nazwa atrybutu, z aliasów można korzystać podczas sortowania i prezentacji wyników

```
SELECT nazwisko,
        placa_pod*12 AS roczna_płaca,
        placa_pod/20 AS dniówka
FROM pracownicy;
```

Operator konkatenacji - ||

- Umożliwia łączenie (sklejanie) wartości wyświetlanych atrybutów tekstowych i literałów

```
SELECT 'Pracownik ' || nazwisko || ' zarabia ' || placa_pod
FROM pracownicy;
```

Obsługa wartości pustych

NULL: wartość niedostępna, nieprzypisana, nieznana lub nieistotna

Funkcja NVL o następującej specyfikacji NVL(*wyrażenie*, *wartość*)

```
SELECT nazwisko,
       placa_pod*12 + placa_dod
FROM pracownicy;
```

```
SELECT nazwisko,
       placa_pod*12 + NVL(placa_dod, 0)
FROM pracownicy;
```

Eliminowanie duplikatów

```
SELECT etat FROM pracownicy;
```

- słowo kluczowe DISTINCT

```
SELECT DISTINCT etat FROM pracownicy;
```

```
SELECT DISTINCT etat, id_zesp FROM pracownicy;
```


Porządkowanie wyników zapytania

klauzula ORDER BY

- kolejność sortowania - słowo kluczowe ASC (*ascending*- rosnąco) lub DESC (*descending*- malejąco)
- ORDER BY występuje zawsze jako ostatnia klauzula zapytania, można w niej korzystać z aliasów i numerów kolumn (użycie numerów kolumn jest niezgodne ze standardem SQL3)
- porządek sortowania:
  - liczby – od mniejszych do większych
  - daty – od wcześniejszych do późniejszych
  - łańcuchy znaków – alfabetycznie
  - wartości puste – w zależności od RDBMS

Jeżeli klauzula ORDER BY nie zostanie użyta to wiersze zostaną zwrócone w całkowicie losowej kolejności

```
SELECT nazwisko, etat, placa_pod * 12 AS roczne_zarobki
FROM pracownicy
ORDER BY etat DESC, roczne_zarobki ASC, 2;
```


Selekcja krotek relacji

Klauzula WHERE
Składnia polecenia

```
SELECT atrybut1, atrybut2, ...
FROM relacja
WHERE atrybutm operator wartość
```

Operatory

- operatory logiczne

  =, !=, <>, >, >=, <, <=

```
SELECT nazwisko, placa_pod, etat
FROM pracownicy
WHERE placa_pod > 400;
```

```
SELECT nazwisko, id_zesp
FROM pracownicy
WHERE placa_dod > (placa_pod/10);
```

```
SELECT id_prac, nazwisko, etat
FROM pracownicy
WHERE etat != 'ASYSTENT';
```


Operatory cd.

- operatory SQL

  BETWEEN ... AND ...

  do przedziału wartości zalicza się wartości graniczne, granica dolna musi poprzedzać granicę górną

```
SELECT nazwisko, placa_pod, etat
FROM pracownicy
WHERE placa_pod BETWEEN 208 AND 1070;
```

IN

jeżeli w zbiorze znajdują się dane znakowe bądź daty to należy je ująć w apostrofy

```
SELECT nazwisko, placa_pod, id_zesp
FROM pracownicy
WHERE etat IN ('PROFESOR', 'DYREKTOR');
```

Operatory cd.

LIKE

do tworzenia wzorca wykorzystujemy znaki specjalne % (dowolny ciąg znaków) i _ (pojedynczy znak)

```
SELECT nazwisko, placa_pod, id_zesp
FROM pracownicy
WHERE nazwisko LIKE 'M%';
```

```
SELECT nazwisko, zatrudniony
FROM pracownicy
WHERE zatrudniony LIKE '%93%';
```

IS NULL

```
SELECT nazwisko, placa_pod
FROM pracownicy
WHERE placa_dod <> NULL;
```

```
SELECT nazwisko, placa_pod
FROM pracownicy
WHERE placa_dod IS NULL;
```


Operatory cd.

negacje operatorów SQL

- **NOT BETWEEN ... AND ...**
- **NOT IN**
- **NOT LIKE**
- **IS NOT NULL**

```
SELECT nazwisko, placa_pod, id_zesp
FROM pracownicy
WHERE etat NOT IN ('PROFESOR', 'DYREKTOR');
```

```
SELECT nazwisko, etat, placa_pod+NVL(placa_dod)
FROM pracownicy
WHERE nazwisko NOT LIKE '%SKI';
```


Warunki złożone klauzuli WHERE

operatory logiczne w klauzuli WHERE

- **AND**
- **OR**

Tabele wartości logicznych

	TRUE	FLASE	UNKNOWN
NOT	FALSE	TRUE	UNKNOWN

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

Warunki złożone klauzuli WHERE cd.

Operatory logiczne mogą być stosowane jednocześnie w tej samej klauzuli WHERE, przy czym AND posiada wyższy priorytet niż OR, zmiana priorytetu jest możliwa za pomocą nawiasów

```
SELECT nazwisko, etat
FROM pracownicy
WHERE placa_pod > 500 AND etat = 'ADIUNKT'
OR etat = 'ASYSTENT';
```

```
SELECT nazwisko, etat
FROM pracownicy
WHERE placa_pod > 500 AND
      (etat = 'ADIUNKT' OR etat = 'ASYSTENT');
```

Podsumowanie polecenia SELECT

```
SELECT [DISTINCT] { * , kolumna [AS alias], ... }
FROM relacja
WHERE warunek [ AND | OR warunek ... ]
ORDER BY { kolumna, wyrażenie } [ASC | DESC];
```

SELECT	Wybiera listę kolumn
alias	Można stosować tylko do kolumn i wyrażeń (nie do *)
*	Oznacza wszystkie kolumny
DISTINCT	Eliminuje duplikaty ze zbioru wynikowego
FROM t	Określa relację, z której odczytujemy dane
WHERE	Określa warunki wyboru wierszy, zawiera wartości kolumn, wyrażenia i literały
AND/OR	Łączy warunki w klauzuli WHERE
()	Pozwala na zmianę priorytetu operatorów
ORDER BY	Służy do określenia kryterium sortowania
ASC	Rosnący porządek sortowania
DESC	Malejący porządek sortowania