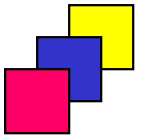
Język SQL. Rozdział 3.
Zaawansowana selekcja danych

Selekcja wg wartości elementów czasowych, ciągów znaków i liczb. Konstrukcja warunkowa.



Funkcje

- **Przekształcają dane, pobrane przez polecenie SQL, lub wyliczają nowe dane.**
- **Funkcje wierszowe – operują na wartościach atrybutów jednego rekordu, funkcja zwraca tyle wyników ile rekordów przetwarza polecenie SQL,**
- **Podział ze względu na pochodzenie:**
  - **funkcje predefiniowane,**
  - **funkcje użytkownika.**
- **Miejsce użycia:**

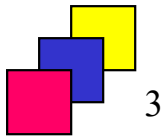
```
SELECT atrybut_1,
        funkcja_A(wyrażenie_1, wyrażenie_2) AS wynik
FROM nazwa_relacji
WHERE funkcja_B(wyrażenie_3) operator wyrażenie_4
ORDER BY funkcja_C;
```

Typy dla elementów czasowych ANSI SQL

- **Typy:**
  - **`DATE` – punkt w czasie z dokładnością do dnia,**
  - **`TIME` – punkt w czasie z dokładnością do części ułamkowych sekundy bez daty,**
  - **`TIMESTAMP` – połączenie `DATE` i `TIME`,**
  - **`INTERVAL` – okres czasu.**
- **Implementacje w różnych SZBD:**

	MS SQL Server	MySQL	OracleDB	PostgreSQL
`DATE`	+	+	+(1)	+
`TIME`	+	+	-	+
`TIMESTAMP`	-	+	+	+
`INTERVAL`	-	-	+	+

(1) zawiera dodatkowo czas z dokładnością do sekundy



Selekcja wg daty

- `DATE 'literał'` – specyfikuje literał określający datę:
  - format: `RRRR-MM-DD`,
  - brak elementów określających czas.
- Przykłady:
  - 1 lipca 2010 r. – `DATE '2010-07-01'`
  - 29 lutego 2016 r. – `DATE '2016-02-29'`
- Użycie:

```
SELECT nazwisko, zatrudniony
FROM pracownicy
WHERE zatrudniony < DATE '1970-01-01';

SELECT  nazwisko, zatrudniony
FROM pracownicy
WHERE zatrudniony BETWEEN DATE '1990-01-01' AND DATE '1999-12-31';
```

Selekcja wg czasu

- `TIME 'literał'` – specyfikuje literał określający czas:
  - format `HH:MI:SS.MMMM`,
  - brak wsparcia w OracleDB.
- Przykłady:
  - 13:46:50 – `TIME '13:46:50'`
  - 23:59:4.43 – `TIME '23:59:4.43'`
- Użycie:

```
SELECT czas_poczatku, czas_konca
FROM harmonogram
WHERE czas_poczatku BETWEEN TIME '08:00:00' AND TIME '16:00:00';
```

Selekcja wg znacznika czasowego

- `TIMESTAMP 'literał'` – literał określający znacznik czasowy:
  - format: `RRRR-MM-DD HH:MI:SS.MMMM`.
- Przykłady:
  - 1 lipca 2010 r. 13:46:50 – `TIMESTAMP '2010-07-01 13:46:50'`
  - 16 lutego 2018 8:00 – `TIMESTAMP '2018-02-16 08:00:00'`
- Użycie:

```
SELECT ip, nazwa_strony
FROM log
WHERE data_wyswietlania > TIMESTAMP '2017-01-01 00:00:00';
```

Interwał czasowy

- **Różnica dwóch elementów określających moment czasie:**

```
SELECT nazwisko, DATE '2000-01-01' - zatrudniony AS staz_w_2000_r
FROM pracownicy
ORDER BY nazwisko;
```

- **Podtypy w zależności od użytych jednostek:**
  - `YEAR TO MONTH` **– jednostki: lata i miesiące, domyślna precyzja:** `YEAR(2)`**, sposób prezentacji:**
    - `+lata-miesiące`**, np.:** `+10-6`, `+5-11`,
  - `DAY TO SECOND` **– jednostki: od dni do części ułamkowych sekundy, domyślna precyzja:** `DAY(2)`,`SECOND(6)`**, sposób prezentacji:**
    - `+dni godziny:minuty:sekundy.częściułamkowesekund`, **np.** `+15 23:33:04.45`, `+7 00:00:03.00`.

Selekcja wg interwału czasowego

- `INTERVAL 'literał' podtyp_interwału` – specyfikuje literał określający interwał, podtypy interwału:
  - `YEAR TO MONTH` – domyślna precyzja: `YEAR(2)`,
  - `DAY TO SECOND` – domyślna precyzja: `DAY(2)`, `SECOND(6)`.
- Przykłady:
  - `INTERVAL '4 5:12' DAY TO MINUTE` – 4 dni, 5 godzin i 12 minut,
  - `INTERVAL '400 5' DAY(3) TO HOUR` – 400 dni i 5 godzin,
  - `INTERVAL '10' HOUR` – 10 godzin,
  - `INTERVAL '10:22' MINUTE TO SECOND` – 10 minut 22 sekundy,
  - `INTERVAL '10' MINUTE` – 10 minut,
  - `INTERVAL '30.12345' SECOND(2,5)` – 30,12345 s,
  - `INTERVAL '1-6' YEAR TO MONTH` – półtora roku,
  - `INTERVAL '100-2' YEAR(3) TO MONTH` – 100 lat i 2 miesiące.

```
SELECT nazwisko, DATE '2000-01-01' - zatrudniony AS staz_w_2000_r
FROM pracownicy
WHERE DATE '2000-01-01' - zatrudniony > INTERVAL '10-6' YEAR TO MONTH;
```

Selekcja wg interwału czasowego

- **Uwaga! W OracleDB różnica dwóch dat daje w wyniku liczbę dni. Konieczna jawna konwersja do interwału:**

```
SELECT nazwisko,
        (DATE '2000-01-01' - zatrudniony) YEAR TO MONTH  AS staz_w_2000_r
FROM pracownicy
WHERE (DATE '2000-01-01' – zatrudniony) YEAR TO MONTH >
        INTERVAL '10-6' YEAR TO MONTH;
```

Użycie funkcji dla elementów czasowych

- **Funkcja `EXTRACT` – wydobywa z el. czasowego jeden ze składników:**
  - **Składnia:**
    `EXTRACT (YEAR/MONTH/DAY/HOUR/MINUTE/SECOND/ TIMEZONE_HOUR/TIMEZONE_MINUTE/TIMEZONE_REGION/ TIMEZONE_ABBR FROM data/czas/interwał)`
  - **UWAGA! Z wartości typu `DATE` możesz wydobyć jedynie elementy `YEAR`, `MONTH` i `DAY`.**
- **Przykład:**

```
SELECT nazwisko, zatrudniony
FROM pracownicy
WHERE EXTRACT(YEAR FROM zatrudniony) BETWEEN 1990 AND 1999;
```

Użycie funkcji dla elementów czasowych

- **Funkcja `TO_DATE` – przekształca ciąg znaków do daty:**
  - **Składnia:**
    `TO_DATE(ciąg_znaków,format_konwersji)`
  - **Elementy ciągu formatującego:**

`SCC`	Stulecie	`D DD DDD`	Dzień
`YYYY`	Rok	`DAY`	Nazwa dnia
`BC AD`	Wskaźnik ery	`AM PM`	Wskaźnik pory dnia
`MM`	Miesiąc	`HH HH24`	Godziny
`MON`	Skrót nazwy m-ca	`MI`	Minuty
`MONTH`	Nazwa miesiąca	`SS`	Sekundy

- **Przykład:**

```
SELECT nazwisko, zatrudniony
FROM pracownicy
WHERE zatrudniony < TO_DATE('01/01/1970', 'DD/MM/YYYY');
```

Prezentacja daty w różnych formatach

- **Funkcja `TO_CHAR` – przekształca datę do ciągu znaków wg zadanego formatu konwersji:**
  - **Składnia:**
    `TO_CHAR(data,format_konwersji)`
  - **Elementy ciągu formatującego identyczne jak dla funkcji `TO_DATE`.**
  - **Wielkość liter w ciągu formatującym ma znaczenie, np. `MON` vs `mon` ("STY" vs "sty").**
  - **Elementy `MONTH` i `DAY` zwracają, odpowiednio, nazwę miesiąca i dnia, dopełnione spacjami do długości najdłuższej nazwy, odpowiednio, miesiąca i dnia. Elementy bez dopełnienia: `fmMONTH` i `fmDAY`.**
- **Przykład:**

```
SELECT nazwisko, TO_CHAR(zatrudniony, 'dd month yyyy') AS zatrudniony
FROM pracownicy ORDER BY nazwisko;
-- Użycie do selekcji
SELECT nazwisko FROM pracownicy
WHERE TO_CHAR(zatrudniony, 'DD fmDAY') = '13 PIĄTEK';
```

Pozostałe funkcje operujące na datach

- **`CURRENT_DATE` – bieżąca data systemowa.**
- **`CURRENT_TIME` – bieżący czas systemowy.**
- **`CURRENT_TIMESTAMP` – bieżący znacznik czasowy.**
- **`MONTHS_BETWEEN(data1,data2)` – liczba miesięcy jakie upłynęły między datami.**
- **`ADD_MONTHS(data,n)` – data plus n miesięcy kalendarzowych.**
- **`NEXT_DAY(data,dzień)` – następna data po podanej, przypadająca na podany dzień.**
- **`LAST_DAY(data)` – data ostatniego dnia w miesiącu podanej daty.**
- **Przykład:**

```
SELECT nazwisko, CURRENT_DATE - zatrudniony AS staz
FROM pracownicy ORDER BY staz DESC;

SELECT LAST_DAY(DATE '2018-02-01') FROM dual;

SELECT nazwisko FROM pracownicy
WHERE MONTHS_BETWEEN(CURRENT_DATE, zatrudniony) <= 12;
```

Selekcja wg ciągów znaków

- **Warunek niezależny od wielkości liter – funkcje `UPPER(ciąg_znaków)` i `LOWER(ciąg_znaków)`**

```
SELECT nazwisko, zatrudniony FROM pracownicy
WHERE UPPER(nazwisko) = 'KONOPKA';

SELECT nazwisko, zatrudniony FROM pracownicy
WHERE LOWER(nazwisko) = 'konopka';
```

- **Selekcja wg podciągu – funkcja `SUBSTR(ciąg,od_pozycji, [długość])`**

```
SELECT nazwa, adres FROM zespoly
WHERE SUBSTR(nazwa, 1, 7) = 'SYSTEMY';

SELECT nazwisko FROM pracownicy
WHERE SUBSTR(nazwisko, -3) = 'ICZ';
```

Selekcja wg ciągów znaków

- **Wyszukanie podciągu – funkcja**
  `INSTR(ciąg,ciąg_szukany,od_pozycji,wystąpienie)`

```
SELECT nazwa FROM zespoly
WHERE INSTR(nazwa, ' ', 1, 1) > 0;

SELECT SUBSTR(nazwa, 1, INSTR(nazwa, ' ', 1, 1)-1) FROM zespoly
WHERE INSTR(nazwa, ' ', 1, 1) > 0;
```

- **Selekcja wg długości ciągu – funkcja `LENGTH(ciąg)`**

```
SELECT nazwisko FROM pracownicy
WHERE LENGTH(nazwisko) > 10;
```

Pozostałe funkcje operujące na ciągach

- **`TRIM([LEADING|TRAILING|BOTH] znak FROM ciąg)` – usuwa podany znak z początku (`LEADING`), końca (`TRAILING`) bądź obu końców (`BOTH`) ciągu znaków.**

```
SELECT TRIM(LEADING 'A' FROM nazwa), TRIM(TRAILING 'E' FROM nazwa)
FROM zespoly;
```

- **`LPAD(ciąg,długość,znaki)` – dopełnia ciąg z lewej strony znakami do danej długości.**
- **`RPAD(ciąg,długość,znaki)` – dopełnia ciąg z prawej strony znakami do danej długości.**

```
SELECT LPAD(nazwisko, 15, '*') FROM pracownicy;
```

Pozostałe funkcje operujące na ciągach

- **`REPLACE(ciąg,ciąg_szukany,ciąg_nowy)` – zamiana podciągów w ciągu.**

```
SELECT REPLACE(nazwisko, 'ICZ', 'SKI') FROM pracownicy
WHERE nazwisko LIKE '%ICZ';
```

- **`TRANSLATE(ciąg,szukane_znaki,znaki_zastępujące)` – zamienia w ciągu wszystkie szukane znaki na znaki zastępujące.**

```
SELECT TRANSLATE(nazwisko, 'ĄĆĘŁŃÓŚŹŻ', 'ACELNOSZZ')
FROM pracownicy;

SELECT TRANSLATE(nazwa, 'AEO', 'EA')
FROM zespoly;
```

Funkcje operujące na liczbach

- `ROUND(liczba,[n])` – zaokrągla liczbę do n miejsc ułamkowych.

```
SELECT nazwisko, ROUND(placa_pod/30,2) AS dniówka
FROM pracownicy;

SELECT nazwisko, placa_pod, ROUND(placa_pod,-3) AS najbliższy_1000
FROM pracownicy;
```

- `TRUNC(liczba,[n])` – obcina liczbę do n miejsc ułamkowych.
- `CEIL(liczba)` – najmniejsza liczba całkowita większa lub równa od podanej liczby
- `FLOOR(liczba)` – największa liczba całkowita mniejsza lub równa od podanej liczby.

Funkcje operujące na liczbach

- `POWER(wartość,n)` – wartość do n-tej potęgi.
- `SQRT(wartość)` – pierwiastek kwadratowy z wartości.
- `ABS(wartość)` – wartość bezwzględna wyrażenia.
- `MOD(wartość1, wartość2)` – reszta z dzielenia.
- `SIGN(wartość)` – zwraca -1 jeśli wartość<0, 0 dla wartości=0 i 1 jeśli wartość>0.
- `GREATEST(w1,w2,...)` – zwraca największą wartość z listy wartości.
- `LEAST(w1,w2,...)` – zwraca najmniejszą wartość z listy wartości.

```
SELECT nazwisko, LEAST(placa_pod, NVL(placa_dod, 0)) AS większe
FROM pracownicy;
```

Konwersja wartości

- **`CAST(wartość AS typ)` – konwertuje wartość do zadanego typu.**

```
SELECT 'Pracownik ' || nazwisko || ' zarabia ' ||
        CAST(placa_pod AS VARCHAR(10)) AS zarobki
FROM pracownicy;
```

- **`TO_CHAR(data,format)` – konwertuje datę do ciągu znaków wg zadanego formatu.**
- **`TO_DATE(ciąg,format)` – konwertuje ciąg znaków do daty wg zadanego formatu.**

Konwersja wartości

- **`TO_CHAR(liczba,format)` – konwertuje liczbę do ciągu znaków wg zadanego formatu.**
- **`TO_NUMBER(ciąg,format)` – konwertuje ciąg znaków do liczby wg zadanego formatu.**

D	separator części całkowitej od ułamkowej
G	separator elementów liczby, np. tysięcy od milionów
9	określa pozycję w liczbie, zera z lewej strony są pomijane
0	określa drukowanie zera z lewej lub prawej strony liczby

- **Przykład:**

```
SELECT nazwisko, TO_CHAR(placa_pod, '9G999D00') AS zarobki
FROM pracownicy;
```

Instrukcja warunkowa CASE-WHEN

- **Wersja z selektorem i wersja z warunkami logicznymi:**

```
CASE wyrażenie
 WHEN wartość1 THEN wartość1
 WHEN wartość2 THEN wartość2
… [ELSE wartośćn] END
```

```
CASE
 WHEN warunek1 THEN wartość1
 WHEN warunek2 THEN wartość2
… [ELSE wartośćn] END
```

- **Przykład:**

```
SELECT nazwisko,
    CASE etat
        WHEN 'DYREKTOR' THEN '***'
        WHEN  'PROFESOR' THEN '***'
        ELSE CAST(placa_pod AS VARCHAR(10))
    END AS placa_pod FROM pracownicy;
```

```
SELECT nazwisko,
    CASE
        WHEN etat IN ('DYREKTOR','PROFESOR') THEN '***'
        ELSE CAST(placa_pod AS VARCHAR(10))
    END AS placa_pod FROM pracownicy;
```

Wiadomości uzupełniające

Funkcja DECODE

- `DECODE(wyrażenie,S1,W1,[S2,W2,...] domyślne)`
- **Pokrywa część funkcjonalności wyrażenia CASE-WHEN.**
- **Funkcja specyficzna dla OracleDB.**
- **Przykład:**

```
SELECT nazwisko,
   DECODE(etat,
        'PROFESOR', '***',
        'DYREKTOR', '***',
        CAST(placa_pod AS VARCHAR(10))) AS placa_pod
FROM pracownicy;
```