SQL: EXISTS vs IN, LIKE vs ILIKE, REGEXP, UNION, GROUP BY, HAVING, INSERT

Robert A. Kłopotek

r.klopotek@uksw.edu.pl

Wydział Matematyczno-Przyrodniczy. Szkoła Nauk Ścisłych, UKSW

Fraza EXISTS

- Jak znaleźć wszystkie rekordy z tabeli nadrzędnej, które mają (lub nie mają) pasujących rekordów?
- Przykład: Wszyscy klienci, który nie zrobili żadnych zamówień:
  - ```
    SELECT * FROM klienci k
    WHERE NOT EXISTS
    (SELECT * FROM zamówienia z WHERE z.id_zamówienia =
    k.id_klienta)
    ```
  - Alternatywnie:
    ```
    SELECT * FROM klienci k
    WHERE k.id_klienta NOT IN
    (SELECT z.id_klienta FROM zamówienia z)
    ```
    ```

IN vs EXISTS – wady i zalety

- IN – optymalizator najpierw zrobi ewaluację podzapytania - w nawiasach, a potem zapytania głównego.
- IN – lepiej użyć, gdy tabela w podzapytaniu jest mała lub używamy i kolumny, która została zaindeksowana (np. klucz główny jest zwykle indeksowaną kolumną)
- EXISTS – optymalizator najpierw zrobi ewaluację głównego zapytania a jego wynik zastosuje do porównania z podzapytaniem
- EXISTS – lepiej użyć, gdy zapytanie główne ma niewiele rekordów a podzapytanie ma dużo rekordów
- EXISTS nie wykorzystuje indeksowanych kolumn

LIKE vs ILIKE

- LIKE – służy do porównywania tekstów, wielkość liter ma znaczenie
- Znaki specjalne - znak „_” to dowolny pojedynczy znak „%” – to dowolny ciąg znaków. Aby wyszukać znaki specjalne (napisy zawierające te znaki) należy je escapować, czyli zapisać „_” lub „\%”
- LIKE używa indeksu (o ile szukana fraza nie rozpoczyna się od znaku specjalnego), co przyśpiesza porównywanie
- ILIKE – to wersja LIKE bez rozróżniania wielkości liter
- ILIKE – używa indeksu tylko wtedy, gdy szukana fraza rozpoczyna się nie od litery (nie potrzeba konwersji)
- ALE: zapytanie „`... LOWER(description) LIKE '%abcde%' ...`” zwykle jest szybsze od „`... description iLIKE '%abcde%' ...`”

REGEXP – wyrażenie regularne

- W języku SQL istnieje możliwość bardziej zaawansowanego wyszukiwania niż LIKE
- Każdy dialekt SQL ma swój własny system i składnię wyrażeń regularnych
- Baza H2 korzysta z wyrażeń regularnych języka Java
- Oznaczenia uznawane za uniwersalne:
  - | - alternatywa
  - [ ] – grupa
  - * - wielokrotne wystąpienie (może być zero dopasowań)
  - + - wielokrotne wystąpienie (musi być co najmniej jedno dopasowanie)
  - \s – znak biały
  - \w – znak alfanumeryczny
  - \d - cyfra

REGEXP - przykłady

- Dopasowanie dowolnego wyrazu:

```
SELECT imie FROM osoby
WHERE( imie REGEXP '\w')
```

- Dopasowanie wyrazu zaczynającego się na „M”:

```
SELECT imie FROM osoby
WHERE( imie REGEXP '[M*]')
```

- Dopasowanie wyrazu kończącego się na „j”:

```
SELECT imie FROM osoby
WHERE( imie REGEXP '[*j]')
```

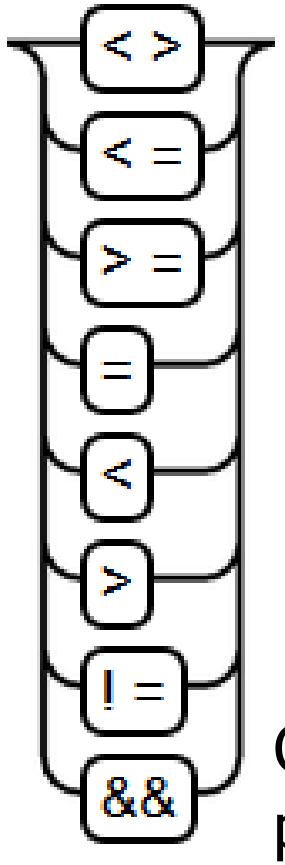
- Dopasowanie wyrazu zaczynającego się na „m” (z ignorowaniem wielkości liter):

```
SELECT imie FROM osoby
WHERE( imie REGEXP '(?i)[m*]')
```

Fraza BETWEEN … AND …

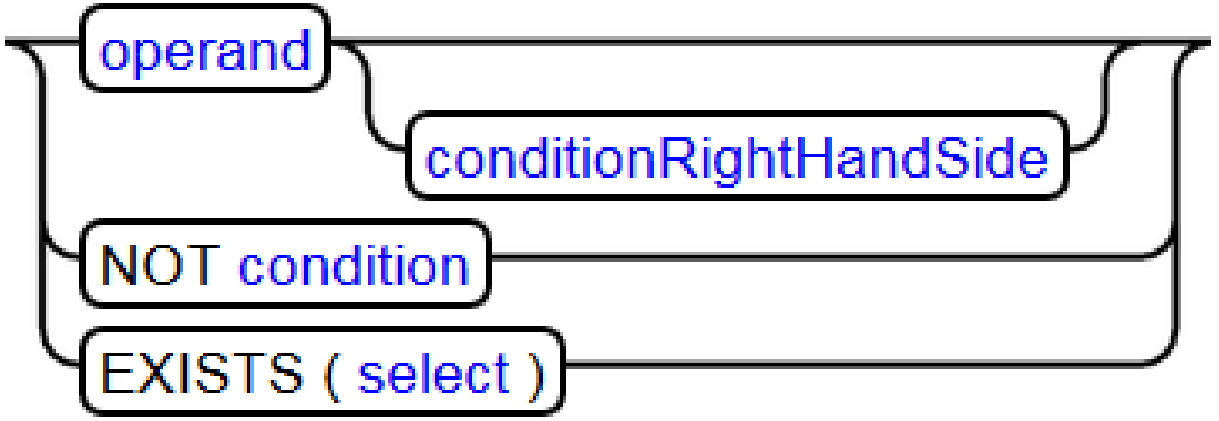
- Znajduje elementy z zadanego przedziału, np.
  - `SELECT imie FROM osoby WHERE imie BETWEEN 'Maciej' AND 'Zbyszek’`
  - `SELECT imie FROM osoby WHERE left(imie,1) BETWEEN 'M' AND 'Z’`
- Działa dla każdej kolumny, dla której możemy porównać elementy, np. cyfry, litery, napisy, daty itp.
- Niestety **działanie zależy od implementacji**:
  - Krańce przedziału wchodzą do zbioru wyszukiwanego (np. H2 SQL server)
  - Jeden z krańców przedziału wchodzi do zbioru wyszukiwanego
  - Żaden z krańców przedziału nie wchodzi do zbioru wyszukiwanego

Porównywanie i instrukcje warunkowe w H2 SQL

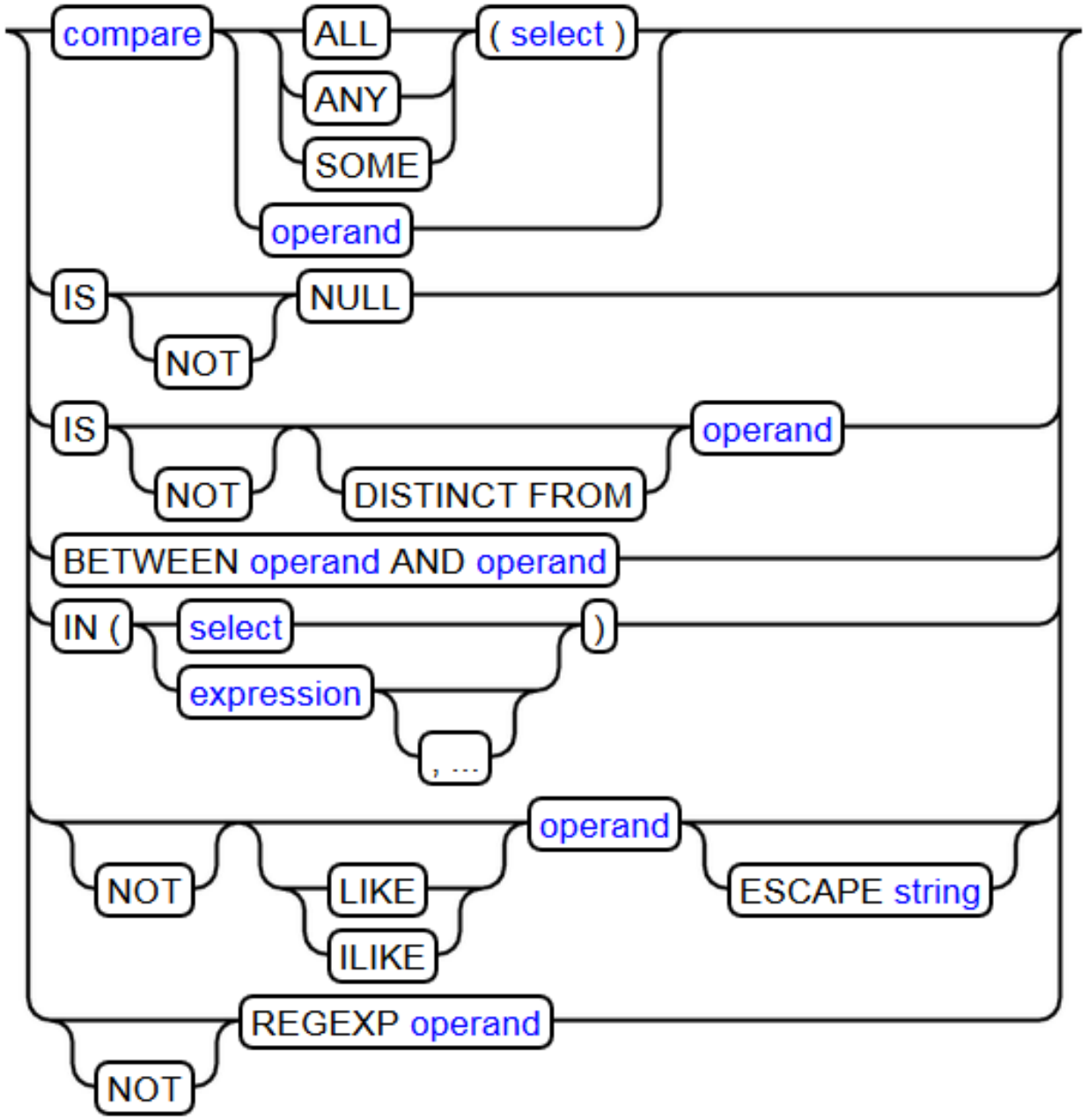


Operatory porównania

Instrukcje warunkowe



Prawa strona porównania w H2 SQL



Fraza GROUP BY

- Pozwala na grupowanie rekordów z tą samą wartością w pewnych kolumnach
- Może być używana wraz z zapytaniem SELECT to jednej lub więcej tabel
- Zwykle używa się jej wraz z funkcjami agregacji

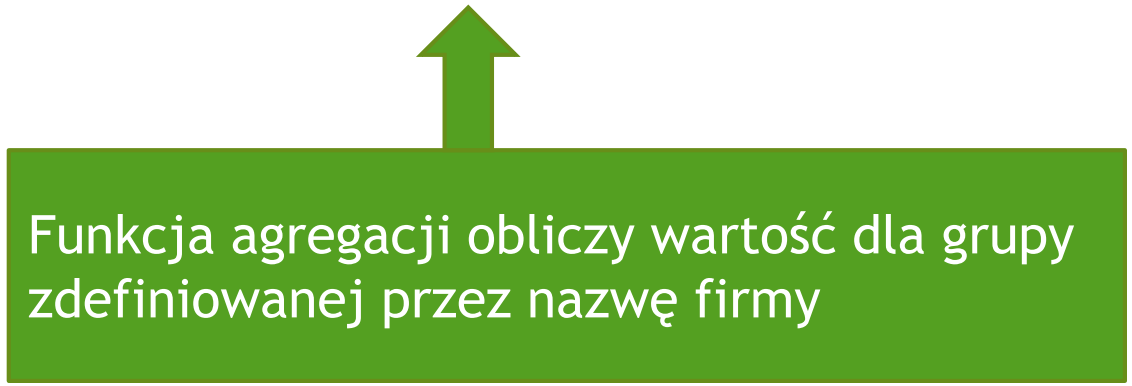
Funkcje agregujące

Kategoria	Opis
COUNT(*)	Oblicza ilość wierszy w grupie
SUM(kolumna_numeryczna)	Oblicza sumę wartości numerycznych w grupie
AVG(kolumna_numeryczna)	Oblicza średnią wartość w grupie
MIN(kolumna)	Oblicza najmniejsza wartość w grupie. Wartości są porównywane wg ich typu – w taki samym porządku jak są sortowane.
MAX(kolumna)	Oblicza największą wartość w grupie. Wartości są porównywane wg ich typu – w taki samym porządku jak są sortowane.

GROUP BY - przykład

- Obliczyć średnią pensję w każdej z firm:

```
SELECT f.nazwa, avg(e.pensja)
FROM etaty e
JOIN firmy f ON f.nazwa_skr = e.id_firmy
GROUP BY f.nazwa
```



Interpretacja kwerendy – krok 1

NAZWA	PENSJA
Uniwersytet Kardynala Stefana Wyszynskiego	600.0000
Uniwersytet Kardynala Stefana Wyszynskiego	1600.0000
Uniwersytet Kardynala Stefana Wyszynskiego	3200.0000
Uniwersytet Kardynala Stefana Wyszynskiego	2200.0000
Hewlett Packard	20000.0000
Uniwersytet Kardynala Stefana Wyszynskiego	3200.0000
Uniwersytet Kardynala Stefana Wyszynskiego	4200.0000
Hewlett Packard	50000.0000
Fabryka Lodzi Podwodnych	6200.0000
Fabryka Lodzi Podwodnych	65200.0000

Po połączeniu tabel po kluczy, aby odpowiedzieć na zapytanie GROUP BY tworzona jest lista unikalnych wartości (tu nazw firm)

Interpretacja kwerendy – krok 2

NAZWA	PENSJA
Uniwersytet Kardynala Stefana Wyszynskiego	600.0000
Uniwersytet Kardynala Stefana Wyszynskiego	1600.0000
Uniwersytet Kardynala Stefana Wyszynskiego	3200.0000
Uniwersytet Kardynala Stefana Wyszynskiego	2200.0000
Uniwersytet Kardynala Stefana Wyszynskiego	3200.0000
Uniwersytet Kardynala Stefana Wyszynskiego	4200.0000

NAZWA	AVG(PENSJA)
Uniwersytet Kardynala Stefana Wyszynskiego	2500

Tu pokazane jest jak liczona jest wartość wynikowa wiersza dla f.nazwa=„Uniwersytet Kardynala Stefana Wyszynskiego”. Podobna operacja jest wykonywana dla innych wartości kolumny f.nazwa.

Interpretacja kwerendy – krok 3

- Dla zapytania otrzymujemy wynik:

```
SELECT f.nazwa, avg(e.pensja)
FROM etaty e
JOIN firmy f ON f.nazwa_skr = e.id_firmy
GROUP BY f.nazwa
```

NAZWA	AVG(E.PENSJA)
Fabryka Lodzi Podwodnych	35700
Hewlett Packard	35000
Uniwersytet Kardynala Stefana Wyszynskiego	2500

GROUP BY - dyskusja

- Tylko kolumny używane we frazie GROUP BY mogą być użyte w liście kolumn SELECT, więc nie można zrobić zapytania:
```
SELECT f.nazwa, f.nazwa_skr, …
FROM firmy f, …
GROUP BY f.nazwa
```
- DLACZEGO? Dlatego, że może być potencjalnie wiele `f.nazwa_skr` dla tej samej `f.nazwa`. Która nazwa skrócona wtedy zastałaby wyświetlona skoro jest tylko jeden rekord dla danej nazwy firmy?
- GROUP BY pozwala na zmniejszenie transferów przez sieć wszędzie tam, gdzie potrzeba nam jedynie zagregowanych podsumowań zamiast całych szczegółowych tabel z rekordami

GROUP BY dla wielu kolumn

- Policz liczbę etatów i średnią pensję dla każdego pracownika w każdej z firm.

```
SELECT id_osoby, id_firmy,
COUNT(*) as liczba_etatow, avg(pensja)
FROM etaty
GROUP BY  id_osoby, id_firmy
ORDER BY 1
```

Funkcja agregacji obliczy wartość dla każdej z grup rekordów z tym samym id_osoby oraz id_firmy

Interpretacja zapytania

ID_OSOBY	ID_FIRMY	PENSJA
1	FLP	6200.0000
1	HP	20000.0000
1	UKSW	4200.0000
1	UKSW	600.0000
1	UKSW	2200.0000
1	UKSW	3200.0000
1	UKSW	1600.0000

ID_OSOBY	ID_FIRMY	PENSJA	LICZBA_ETATOW
1	FLP	6200	1
1	HP	20000	1
1	UKSW	2360	5

Tu pokazane jest jak liczona jest wartość wynikowa wiersza dla id_osoby =1. Podobna operacja jest wykonywana dla innych wartości kolumny id_osoby oraz id_firmy.

Fraza HAVING

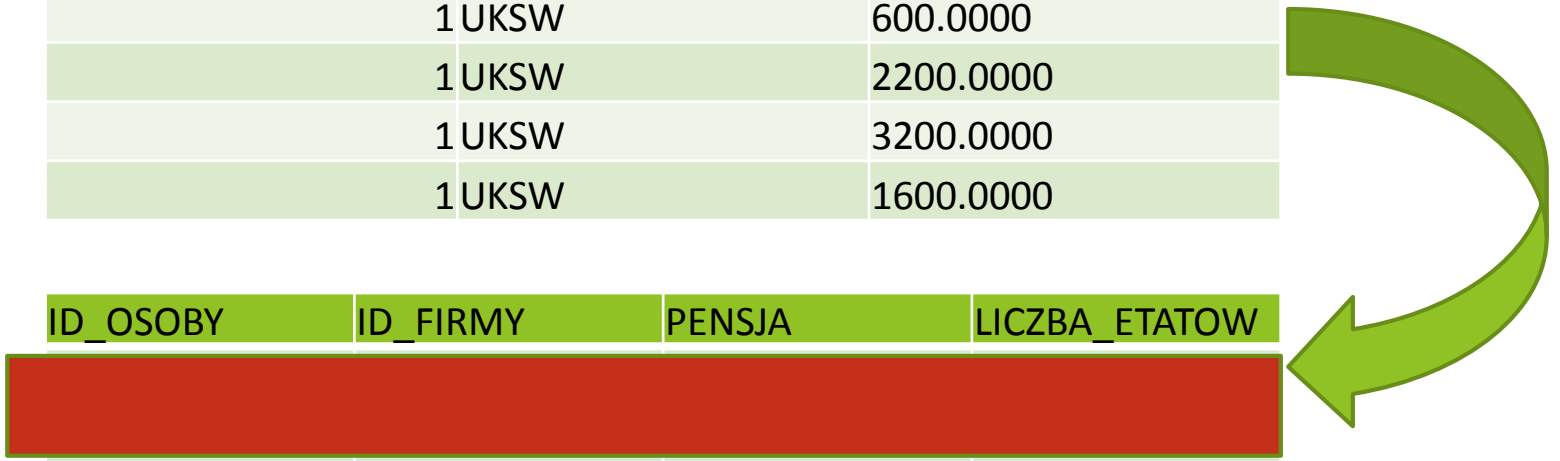
- Pozwala na wyrażenia logiczne na wynikach funkcji agregacyjnych

```
SELECT id_osoby, id_firmy,
COUNT(*) as liczba_etatow, avg(pensja)
FROM etaty
GROUP BY  id_osoby, id_firmy
HAVING  liczba_etatow > 1
```

Tylko te wiersze, które będą spełniały warunek zawarty w HAVING będą w zbiorze wynikowym

Interpretacja zapytania - HAVING

ID_OSOBY	ID_FIRMY	PENSJA
1	FLP	6200.0000
1	HP	20000.0000
1	UKSW	4200.0000
1	UKSW	600.0000
1	UKSW	2200.0000
1	UKSW	3200.0000
1	UKSW	1600.0000



ID_OSOBY	ID_FIRMY	PENSJA	LICZBA_ETATOW
1	UKSW	2360	5

Tylko te rekordy, które spełniają wyrażenie HAVING zostaną w zbiorze wynikowym.

Wartość NULL oraz porównania

- Jak porównywać z wartością NULL?
- `WHERE … nazwa_kolumny IS [NOT] NULL …`
- Należ być ostrożnym w funkcjami MIN(), MAX(), SUM() – mogą zwracać niespodziewane wyniki jesli używamy wartości NULL

Fraza UNION

- Pozwala na dodawanie (łączenie wierszowe) wyników wielu zapytań
- `SELECT … FROM …`
  `UNION [ALL]`
  `SELECT … FROM …`
- Jeśli pominiemy słowo ALL w wynikowym zbiorze nie dostaniemy duplikatów wierszy

UNION - przykład

```
SELECT imie
FROM osoby
UNION
SELECT nazwa
FROM firmy
```

IMIE
Jacek
Fabryka Lodzi Podwodnych
Krol
Uniwersytet Kardynala Stefana Wyszynskiego
Mis
Hewlett Packard
Juz
Maciej

UNION - wymagania

- Ilość kolumn wynikowych łącznych zapytań musi być taka sama
- Typy wszystkich łącznych kolumn muszą być takie same
- Jeśli wartość kolumny jest określana przez wyrażenie, to może być konieczne, aby zadeklarować jego typ, aby uniknąć dwuznaczności w typach kolumn

Widoki

- Widoki mogą być traktowane jako zapisane zapytanie SQL
- Przykład:

```
CREATE VIEW WszystkieEtaty AS
SELECT id_osoby,
COUNT(*) as liczba_etatow, avg(pensja)
FROM etaty
GROUP BY  id_osoby
```

-
```
SELECT * FROM WszystkieEtaty
WHERE liczba_etatow = 1
```

Po zdefiniowaniu widoku może być on użyty jako zwykła tabela, jednak wydajność może ulec pogorszeniu.

Widoki - wskazówki

- Może być stworzony aby ograniczyć dostęp, np. użytkownik może mieć dostęp do liczy etatów, ale nie ma już dostępu do bardziej szczegółowych informacji dotyczących etatu
- Widoki nie powinny być tworzone tam, gdzie istniej potrzeba wykonywania wielokrotnych zapytań (lepiej użyć oddzielnej tabeli)
- Widoki zapewniają środki do integracji systemów, np. aplikacje zdalne maja dostęp do widoków a nie martwią się jaka jest wewnętrzna struktura bazy danych

Instrukcja INSERT

- Pozwala na wstawienie rekordu lub kilku rekordów do tabeli lub widoku
- `INSERT INTO TableName [(Column_1,Column_2,…)] VALUES (Value_1,Value_2,…)`
- `INSERT INTO TableName [(Column_1,Column_2,…)] SELECT … FROM …`

INSERT - dyskusja

- Listę kolumn można pominąć. W takim przypadku należy podać wartości dla wszystkich kolumn w porządku w jakim były kolumny podczas tworzenia tabeli.
- Zamiast frazy VALUES możemy napisać instrukcję SELECT. Jeśli jej użyjemy to:
  - Liczba i typ kolumn wynikowych muszą odpowiadać typom w definicji tabeli oraz użyta jest lista nazw kolumn
  - Wiele rekordów (liczba odpowiada ilości zwróconych wierszy przez zapytanie SELECT) może być wstawione z użyciem pojedynczej instrukcji INSERT

Pytania?