Relacyjne Bazy Danych

Andrzej M. Borzyszkowski
PJATK/ Gdańsk

materiały dostępne elektronicznie
http://szuflandia.pjwstk.edu.pl/~amb

Język SQL, cz. 2, operowanie na danych (*data manipulation language*)

© Andrzej M. Borzyszkowski
Relacyjne Bazy Danych

Cztery główne operacje / słowa kluczowe

- Stosowane są do tabel, nie zbiorów
  - wiersze mogą się powtarzać
  - kolejność wierszy gra rolę
- SELECT – główna operacja wyszukiwania danych,
  - realizuje zmianę nazwy, obcięcie, rzut i złączenie relacji
- INSERT – realizuje aktualizację/wstawianie danych
- UPDATE – realizuje aktualizację/zmianę wartości danych
- DELETE – realizuje aktualizację/usuwanie danych

- Notacja użyta dalej
  - [ ] oznacza element składniowy opcjonalny
  - | oznacza wybór jednego z elementów składniowych

© Andrzej M. Borzyszkowski
Relacyjne Bazy Danych

Instrukcja INSERT – składnia

- INSERT INTO *cel* [(*lista_elementów*)] *źródło;*
  - *cel* jest nazwą tabeli, do której wstawiamy dane
  - *lista_elementów* zawiera listę nazw atrybutów, którym chcemy nadać wartość, może być mniejsza niż pełna list tabeli *cel*
  - *źródło* ma jedną z dwu postaci

  VALUES (*lista_wartości*)

  albo tabela otrzymana w wyniku operacji SELECT
- Od pewnej wersji PostgreSQL dopuszcza wygodniejszą formę wstawiania wielu wierszy:

  VALUES (*lista_wartości*) [,(*lista_wartości*)]*

  tzn. wymienienie wielu wierszy pod jednym słowem VALUES

© Andrzej M. Borzyszkowski
Relacyjne Bazy Danych

Instrukcja INSERT – przykład

```
INSERT INTO kod_kreskowy
VALUES ('4892840112975', 17)
```

- wstawia jeden wiersz
- nadaje wartości atrybutom zadeklarowanym w definicji tabeli, w kolejności deklaracji
- nie można opuścić żadnego z atrybutów

```
INSERT INTO towar ( opis, koszt )
VALUES ( 'donica duża', 26.43 ),
       ('donica mała', 13.36)
```

- wstawia dwa wiersze
- atrybuty „nr” oraz „cena” nie zostały wymienione
- będą miały wartość domyślną (kolejny numer / NULL)
- kolejność atrybutów nie musi być zgodna z kolejnością deklaracji w tabeli

Instrukcja INSERT – przykład, c.d.

-
```
INSERT INTO chwilowa
  SELECT imie, nazwisko, ulica_dom
  FROM klient
  WHERE miasto = 'Gdańsk'
```
  - wstawia do utworzonej wcześniej tabeli 'chwilowa' całą tabelę otrzymaną w wyniku obliczenia operacji SELECT
- Celowe może być zdefiniowanie tabeli jako
```
CREATE TEMP TABLE chwilowa ( imię varchar(11), .....
```
  - taka tabela jest usuwana po zakończeniu sesji
-
```
INSERT INTO towar ( opis, koszt, cena )
VALUES ( 'ramka do fotografii 3''x4''', 13.36, NULL )
```
  - podwójny apostrof służy do wprowadzenia znaku apostrofu
  - można wprowadzić w jawny sposób wartość nieokreśloną

Instrukcja SELECT – składnia

- **SELECT [ALL | DISTINCT]** *lista_atrybutów_wynikowych* **[** *lista_klauzul* **];**
- *lista_atrybutów_wynikowych* realizuje m.in. rzut i zmianę nazwy kolumny, nie może być pusta
- *lista_klauzul* realizuje m.in. obcięcie i złączenie
- klauzule: **FROM WHERE ORDER BY GROUP BY HAVING**

```
SELECT DISTINCT imie, nazwisko
      -- rzut na atrybuty
FROM klient
WHERE miasto = 'Gdańsk'
      -- obcięcie do wierszy
         spełniających warunek
```

imię	*nazwisko*
Agnieszka	Kołak
Andrzej	Sosnowy
Barbara	Songin
Ewa	Hałasa
Jan	Soroczyński
Marzena	Niezabitowska-Nasiadko

Instrukcja SELECT – klauzula FROM

- Klauzula FROM
  FROM *lista_tabel*
- Lista tabel nie może być pusta
- Wynikiem jest iloczyn kartezjański tabel
- **SELECT * FROM klient**
  - jedna tabela, iloczyn równy tej tabeli
- **SELECT * FROM towar, kod_kreskowy**
  - iloczyn kartezjański dwu tabel
- **SELECT * FROM klient, towar**
  - w obu tabelach występuje atrybut „nr”, czysto przypadkowa zbieżność
  - podając nazwę atrybutu, w przypadku takiej zbieżności, trzeba dodać nazwę tabeli

Instrukcja SELECT – klauzula WHERE

- Klauzula WHERE
  WHERE *wyrażenie_warunkowe*
- Występuje po klauzuli FROM
- Wynikiem jest wybór tych wierszy, które spełniają warunek
  SELECT * FROM klient WHERE miasto = 'Gdańsk'
  - obcięcie relacji w/g warunku **miasto = 'Gdańsk'**
  - σ[miasto='Gdańsk'](Klient) (sigma)
- Warunek:
  - równość, nierówność itp. na atrybutach
  - należenie atrybutu do zbioru (tabela 1 kolumnowa)
  - operacje logiczne na prostszych warunkach
- Klauzula nie musi występować, wówczas wybrane są wszystkie wiersze tabeli

Instrukcja SELECT – różne warunki WHERE

- Podaj nazwiska klientów spoza Trójmiasta:
  SELECT nazwisko FROM klient
  WHERE miasto NOT IN ('Gdańsk', 'Gdynia', 'Sopot')
  - warunek należenia do zbioru
- Podaj opis wszystkich ramek do fotografii, które moją podany wymiar w calach (tj. znak prim na końcu opisu)
  SELECT opis FROM towar
  WHERE opis LIKE E'ramka%' and opis LIKE E'%\''
  - dopasowanie wzorca tekstowego
- Wyświetl szczegóły zamówień złożonych w marcu 2020
  SELECT * FROM zamowienie
  WHERE data_zlozenia BETWEEN '2020-03-01' AND '2020-03-31'
  - warunek dla zakresu dat

Instrukcja SELECT – wyrażenia warunkowe w klauzuli WHERE

- Pojedyncze wartości: **WHERE cena > 3.14**
- Relacja pomiędzy wartością a zbiorem wartości:
  WHERE miasto NOT IN ('Gdańsk', 'Gdynia', 'Sopot')
  WHERE koszt >= ALL (SELECT koszt FROM towar)
- Istnienie elementów: **WHERE NOT EXISTS (SELECT ***
- Jednoznaczność elementów:
  SELECT * FROM zamowienie
  WHERE NOT
  klient_nr MATCH UNIQUE (SELECT nr FROM klient)

(to się nie powinno zdarzyć, jeśli nr jest kluczem w tabeli klientów)

Instrukcja SELECT – złączenie 1

- Złączenie jest wyborem pasujących wierszy w iloczynie kartezjańskim
  SELECT klient.nr, nazwisko, imie, data_zlozenia
  FROM klient, zamowienie WHERE klient.nr = klient_nr
  - bez warunku WHERE byłyby wszystkie pary wierszy
  - czyli iloczyn kartezjański
  - w obu tabelach występuje atrybut „nr”, trzeba wyjaśnić, o który chodzi
- Wygodne może być stosowanie *aliasów* dla nazw tabel
  SELECT K.nr, nazwisko, imie, data_zlozenia
  FROM klient K, zamowienie WHERE K.nr = klient_nr
  - w złączeniach wielokrotnie powtarzamy nazwę tabeli
  - ale jeśli alias jest zadeklarowany, musi być koniecznie używany

Złączenie, przykład

nr	nazwisko	imie
3	Szczęsna	Jadwiga
4	Łukowski	Bernard
5	Soroczyński	Jan
6	Niezabitowska-Nasiadko	Marzena
7	Kołak	Agnieszka
8	Kołak	Agnieszka

klient_nr	data_zlozenia
3	21.02.2019
3	23.03.2019
3	13.03.2019
5	4.05.2019
6	1.02.2019
6	22.03.2019
8	7.04.2019
8	12.01.2019

nr	nazwisko	imie	data_zlozenia
3	Szczęsna	Jadwiga	21.02.2019
3	Szczęsna	Jadwiga	23.03.2019
3	Szczęsna	Jadwiga	13.03.2019
5	Soroczyński	Jan	4.05.2019
6	Niezabitowska-Nasiadko	Marzena	1.02.2019
6	Niezabitowska-Nasiadko	Marzena	22.03.2019
8	Kołak	Agnieszka	7.04.2019
8	Kołak	Agnieszka	12.01.2019

Instrukcja SELECT – złączenie 2

- Inna składnia na złączenie

```
SELECT K.nr, nazwisko, imie, data_zlozenia
FROM klient K INNER JOIN zamowienie ON K.nr  = klient_nr
```

  - bezpośrednie odwołanie się do operacji złączenia w algebrze relacyjnej
  - deklaracja atrybutu klient_nr jako klucza obcego wskazującego na klient(nr) nie zwalnia z obowiązku napisania jawnego warunku dla złączenia
  - słowo kluczowe INNER jest domyślne (będą inne złączenia)

Instrukcja SELECT – przykłady podstawowe

- Podaj nazwiska i numery telefonów klientów z Gdyni

```
SELECT  nazwisko, telefon
FROM klient
WHERE miasto = 'Gdynia'
```

  - obcięcie i rzut w jednym
- Podaj opis i kod kreskowy wszystkich towarów

```
SELECT opis, kod
FROM towar T, kod_kreskowy
WHERE T.nr = towar_nr
```

  - złączenie

```
SELECT opis, kod
FROM towar T INNER JOIN kod_kreskowy
ON T.nr = towar_nr
```


SQL a rachunek krotek

- Zapytanie

```
SELECT atrybuty FROM tabele WHERE warunek
```

  - bezpośrednio przypomina konstrukcję

$\{<t.A1,..,t.An> \mid r(t) \text{ AND } \Phi(t)\}$ lub $\{ t \mid r(t) \text{ AND } \Phi(t) \}$

- Zmienna *t*, która przebiega krotki, nie musi wystąpić w postaci jawnej
  - ale wygodnie jest myśleć, że wykonanie zapytania polega na pętli przebiegającej wszystkie krotki

```
SELECT K.imie, K.nazwisko, T.opis
FROM klient K, zamowienie Z, pozycja P, towar T
WHERE K.nr = Z.klient_nr AND Z.nr = P.zamowienie_nr AND
   P.towar_nr = T.nr
```

  - występują jawne nazwy dla krotek z tabel
  - nazwy atrybutów poprzedzone są nazwą (aliasem) tabeli

SQL a rachunek krotek, c.d.

- Zapytanie

  SELECT DISTINCT K.nazwisko
  FROM klient K, zamowienie Z
  WHERE K.nr = Z.klient_nr

  - oznacza

    { K.nazwisko | Klient(K) AND
    ∃ Z (Zamówienie(Z) AND K.nr=Z.klient_nr) }

  - w rachunku krotek występował kwantyfikator egzystencjalny
  - w SQL jest on niejawny – rzut dotyczy istniejących par krotek, w szczególności istnieje zamówienie spełniające warunek
  - warunek Klient(K) od razu gwarantuje, że jest tylko skończona liczba krotek do rozpatrzenia