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*) c.d.

Cztery główne operacje / słowa kluczowe

- **INSERT** – realizuje aktualizację/wstawianie danych
- **SELECT** – główna operacja wyszukiwania danych
- **SELECT** [**ALL** | **DISTINCT**] *lista_atrybutów_wynikowych*
  [*lista_klauzul*];
  - *lista_atrybutów_wynikowych* rzut i zmiana nazwy kolumny
  - *lista_klauzul* obcięcie, złączenie, funkcje agregujące, porządkowanie
  - klauzule: **FROM** **WHERE** **ORDER BY** **GROUP BY** **HAVING**
- **UPDATE** – realizuje aktualizację/zmianę wartości danych
- **DELETE** – realizuje aktualizację/usuwanie danych

Instrukcja SELECT – lista atrybutów

- Atrybut wynikowy jest albo gwiazdką * albo postaci
  wyrażenie_skalarne [**AS** *nazwa_kolumny*]
- * oznacza wszystkie atrybuty
  SELECT * FROM towar
  - wyświetla całą tabelę towarów
- *wyrażenie_skalarne* będzie najczęściej nazwą pojedynczego atrybutu
  SELECT imie, nazwisko FROM klient
- Realizuje rzut relacji: π[imie,nazwisko](Klient)
- **DISTINCT** usuwa powtarzające się wiersze w tabeli wynikowej, domyślnie jest **ALL**
  - cena usuwania nie jest błaha przy większych danych
  - niektóre implementacje porządkują wynik, nie jest to standard

Instrukcja SELECT – atrybuty wynikowe

- *wyrażenie_skalarne* może odwoływać się do nazw atrybutów, ale zawierać dodatkowe obliczenia
- *nazwa_kolumny* będzie nazwą kolumny w tabeli wynikowej
- SELECT *, cena – koszt AS zysk FROM towar
  - dodaje nową kolumnę w wyświetlanym wyniku
  - zawiera ona wyniki obliczeń

nr	*opis*	*koszt*	*cena*	*zysk*
1	układanka drewniana	15,23	21,95	6,72
2	układanka typu puzzle	16,43	19,99	3,56
3	kostka Rubika	7,45	11,49	4,04
4	Linux CD	1,99	2,49	0,50
5	chusteczki higieniczne	2,11	3,99	1,88
6	ramka do fotografii 4'x6'	7,54	9,95	2,41

Instrukcja SELECT – atrybuty wynikowe c.d.

- Bardziej wymyślne wyrażenie

```
SELECT *, cena – koszt AS zysk,
    case when (cena–koszt)/koszt < 0 then 'ujemny'
       when (cena–koszt)/koszt < 0.4  then 'za mało'
       when cena is NULL then 'brak danych'
       else 'ok'
    end as opinia
FROM towar
```

nr	*opis*	*koszt*	*cena*	*zysk*	*opinia*
8	ramka do fotografii 3'x4'	13,36	19,95	6,59	ok
9	szczotka do zębów	0,75	1,45	0,70	ok
10	moneta srebrna z Papieżem	20,00	20,00	0,00	za mało
11	torba plastikowa	0,01	0,00	-0,01	ujemny
12	głośniki	19,73	25,32	5,59	za mało
13	nożyczki drewniane	8,18			brak danycl
14	kompas wielofunkcyjny	22,10			brak danycl

Instrukcja SELECT – atrybuty wynikowe c.d.

- Możliwość wykonania obliczeń wykracza poza proste operacje algebry relacji (rzut uogólniony)
- Dodatkowe obliczenia w wyrażeniu skalarnym nie muszą ograniczać się do atrybutów z tabel
- SELECT 2+2
- SELECT now()
- SELECT version()

```
                                   version
-------------------------------------------------------------------------------------------------
 PostgreSQL 10.12 (Ubuntu 10.12-0ubuntu0.18.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0, 64-bit
(1 row)
```

  - tabela wynikowa w ogóle nie odwołuje się do żadnej relacji

Instrukcja SELECT – klauzula ORDER BY

- Klauzula ORDER BY

  ORDER BY *lista_kolumn* [DESC | ASC]
- Występuje po klauzulach FROM i WHERE
- Wynikiem jest tabela, w której wiersze uporządkowano według atrybutów z listy kolumn, kolejność rosnąca (ASC, domyślnie) lub malejąca (DESC)

  SELECT * FROM towar ORDER BY koszt DESC
  - wyświetla tabelę towarów uporządkowaną według kosztów, zaczynając od największych

  SELECT * FROM towar ORDER BY koszt DESC LIMIT 3
  - dodatkowa opcja pozwalająca ograniczyć wyświetlanie

Instrukcja SELECT – funkcje agregujące

- *wyrażenie_skalarne* w części SELECT może być funkcją obliczaną dla wielu/wszystkich wierszy tabeli
  - jeśli nie wystąpi zmiana nazwy AS *nazwa_kolumny* to nazwa funkcji będzie nazwą w tabeli wynikowej

```
SELECT count(*) FROM klient
```

  - zwraca liczbę klientów
  - tylko jedna kolumna, o nazwie „count”, i jeden wiersz
  - wynik może być użyty jako pojedyncza liczba

```
SELECT count (DISTINCT nazwisko) FROM klient
```

  - usuwa powtórzenia przed podjęciem zliczania

```
SELECT max(koszt), min(koszt), avg(koszt) AS średni
FROM towar
```

  - wyświetla tabelę o jednym wierszu i trzech kolumnach

Instrukcja SELECT – klauzula GROUP BY

- Klauzula GROUP BY

  GROUP BY *lista_kolumn*

- Występuje po klauzulach FROM i WHERE
- Wynikiem jest tabela, w której zgrupowano wiersze o identycznych atrybutach z listy kolumn
- Elementy wyboru instrukcji SELECT mają obowiązek dawać jednoznaczna wartość dla każdej grupy:
  - albo muszą odwoływać się do atrybutów z listy kolumn, w/g których grupujemy
  - albo do funkcji agregujących

```
SELECT towar_nr, count(zamowienie_nr), sum(ilosc)
FROM pozycja
GROUP BY towar_nr
ORDER BY count(zamowienie_nr) DESC
```


Instrukcja SELECT – klauzula GROUP BY, c.d.

- Wymóg jednoznaczności dla wartości atrybutu traktowany jest w SQL formalnie
  - tzn. można odwoływać się do tylko atrybutów, w/g których następuje grupowanie
  - nie wystarczy gwarancja jednoznaczności poprzez użycie klucza kandydującego
  - w poniższym przykładzie trzeba dodać atrybut opis do grupowania, mimo że nie spowoduje to zmiany grup

```
SELECT towar.nr, opis, count(zamowienie_nr), sum(ilosc)
FROM pozycja INNER JOIN towar ON towar_nr=towar.nr
GROUP BY towar.nr,opis
ORDER BY count(zamowienie_nr) DESC
```

- W Postgresie wersji 9, można opuścić atrybut opis w powyższym przykładzie

Instrukcja SELECT – klauzula HAVING

- Klauzula HAVING

  HAVING *wyrażenie_warunkowe*

  - Występuje po innych klauzulach
  - Wynikiem jest tabela taka jak otrzymana poprzez użycie GROUP BY, ale dodatkowo z wyeliminowanymi grupami nie spełniającymi wyrażenia warunkowego
  - Brak GROUP BY oznacza, że cała tabela jest jedną grupą
  - Wyrażenie warunkowe odwołuje się do wartości, które można wyświetlić legalnie w SELECT

```
SELECT towar_nr, count(zamowienie_nr) FROM pozycja
GROUP BY towar_nr
HAVING count(zamowienie_nr) > 1
ORDER BY count(zamowienie_nr) DESC
```

Instrukcja SELECT – klauzula HAVING, użycie

- SELECT towar.nr, opis, count(zamowienie_nr)
  FROM pozycja INNER JOIN towar on towar_nr=towar.nr
  GROUP BY towar.nr, opis
  HAVING opis LIKE '%układanka%'
  - jest prawidłowe, ale nielogiczne i niesłuszne
  - HAVING jest słuszne, gdy odwołuje się do wartości zagregowanych
  - wartości pojedynczych krotek powinny być zbadane przed grupowaniem, w klauzuli WHERE

  SELECT towar.nr, opis, count(zamowienie_nr)
  FROM pozycja INNER JOIN towar on towar_nr=towar.nr
  WHERE opis LIKE '%układanka%'
  GROUP BY towar.nr, opis