



Bazy Danych

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytetu Gdańskiego

materiały dostępne elektronicznie
<http://inf.ug.edu.pl/~amb>

© Andrzej M. Borzyszkowski
Bazy Danych

Instrukcja SELECT – złączenie naturalne

SELECT opis, kod

FROM towar NATURAL JOIN kod_kreskowy K (kod,nr)

- alias dla tabeli jednocześnie wprowadził aliasy dla kolejnych atrybutów tabeli
- NATURAL JOIN nie wymaga podania warunku złączenia, tabele złączane są w/g pasujących nazw atrybutów
- nawet jeśli zbieżność jest przypadkowa

SELECT nr, nazwisko, opis, data_wysylki

FROM (klient NATURAL JOIN towar) NATURAL JOIN zamowienie

nr	nazwisko	opis	data_wysylki
1	Kuśmierek	układanka drewniana	17.03.2024
2	Chodkiewicz	układanka typu puzzle	22.01.2024
3	Szczęsna	kostka Rubika	9.02.2024
4	Łukowski	Linux CD	9.03.2024

© Andrzej M. Borzyszkowski
Bazy Danych

3

Język SQL, operowanie na danych (*data manipulation language*) (uzupełnienia)

© Andrzej M. Borzyszkowski
Bazy Danych

Instrukcja SELECT – złączenie naturalne, c.d.

- Wynik naturalnego złączenia jest prawidłowy *jedynie* przy założeniu, że odpowiadające sobie atrybuty mają identyczne nazwy w różnych tabelach
 - założenie mało prawdopodobne w dużej bazie danych

nr	nazwisko	imie	miasto
1	Kuśmierek	Małgorzata	Gdynia
2	Chodkiewicz	Jan	Gdynia
3	Szczęsna	Jadwiga	Gdynia
4	Łukowski	Bernard	Gdynia

nr	klient_nr	data_wysylki	nr	opis	cena
1	3	17.03.2024	1	układanka drewniana	21.95
2	8	22.01.2024	2	układanka typu puzzle	19.99
3	15	9.02.2024	3	kostka Rubika	11.49
4	13	9.03.2024	4	Linux CD	2.49

© Andrzej M. Borzyszkowski
Bazy Danych

4

Instrukcja SELECT – theta-złączenie, konieczność aliasów

- Czasami wygodne może być stosowanie aliasów dla nazw tabel
SELECT K.nr, nazwisko, imie, data_zlozenia
FROM klient K, zamowienie Z WHERE K.nr = klient_nr
- Ale czasami jest niezbędne:
 - podaj nazwiska par klientów z tego samego miasta
SELECT I.nazwisko, II.nazwisko, I.miasto
FROM klient I, klient II
WHERE I.miasto = II.miasto
AND I.nr < II.nr
 - konieczna zmiana nazwy tabel
 - *nie* jest to złączenie względem pary klucz obcy–klucz główny
 - użycie innego warunku nazywa się Θ (theta)-złączeniem
 - dodatkowy warunek ma trochę uporządkować wydruk

5

© Andrzej M. Borzyszkowski

Bazy Danych

Zagnieżdżenia: wydajność

- Zagnieżdżenie skorelowane wymaga wykonania innego podzapytania w każdym wierszu
 - czyli złożoność wykonania będzie radykalnie większa
 - zagnieżdżenie skorelowane

```
EXPLAIN SELECT * FROM zamowienie  
WHERE ( SELECT miasto FROM klient  
WHERE nr = klient_nr ) = 'Gdańsk'
```

```
QUERY PLAN Seq Scan on zamowienie (cost=0.00..11455.00 rows=7  
width=30)
```

```
Filter: (((SubPlan 1))::text = 'Gdańsk')::text
```

- zagnieżdżenie nieskorelowane

```
EXPLAIN SELECT * FROM zamowienie  
WHERE klient_nr IN ( SELECT nr FROM klient  
WHERE miasto = 'Gdańsk' )
```

```
QUERY PLAN Hash Join (cost=12.14..39.89 rows=8 width=30)
```

```
Hash Cond: (zamowienie.klient_nr = klient.nr)
```

© Andrzej M. Borzyszkowski

Bazy Danych

6

Zagnieżdżenia: wydajność c.d.

- Zagnieżdżenie skorelowane nominalnie wymaga wykonania innego podzapytania w każdym wierszu
 - ale optymalizator zapytań może znaleźć inny plan wykonania
 - zagnieżdżenie skorelowane:

```
EXPLAIN SELECT * FROM zamowienie  
WHERE EXISTS ( SELECT * FROM klient  
WHERE nr = klient_nr and miasto = 'Gdańsk' )
```

```
QUERY PLAN Hash Join (cost=12.14..39.89 rows=8 width=30)
```

```
Hash Cond: (zamowienie.klient_nr = klient.nr)
```

- zagnieżdżenie nieskorelowane

```
EXPLAIN SELECT * FROM zamowienie  
WHERE klient_nr IN ( SELECT nr FROM klient  
WHERE miasto = 'Gdańsk' )
```

```
QUERY PLAN Hash Join (cost=12.14..39.89 rows=8 width=30)
```

```
Hash Cond: (zamowienie.klient_nr = klient.nr)
```

7

© Andrzej M. Borzyszkowski

Bazy Danych

Zagnieżdżenia: wydajność 3.

- Optymalizator zapytań może znaleźć nawet lepszy plan wykonania
 - zagnieżdżenie skorelowane:

```
EXPLAIN SELECT imie, nazwisko, miasto FROM klient K  
WHERE EXISTS ( SELECT * FROM klient  
WHERE nazwisko=K.nazwisko AND nr < K.nr )
```

```
QUERY PLAN Hash Semi Join (cost=13.82..26.75 rows=57 width=214)
```

```
Hash Cond: ((k.nazwisko)::text = (klient.nazwisko)::text)
```

```
Join Filter: (klient.nr < k.nr)
```

- zagnieżdżenie nieskorelowane:

```
EXPLAIN SELECT imie, nazwisko, miasto FROM klient  
WHERE nazwisko IN ( SELECT nazwisko FROM klient  
GROUP BY nazwisko HAVING count (nazwisko) > 1 )
```

```
QUERY PLAN Hash Join (cost=18.07..30.23 rows=170 width=214)
```

```
Hash Cond: ((klient.nazwisko)::text = (klient_1.nazwisko)::text)
```

```
-> Seq Scan on klient (cost=0.00..11.70 rows=170 width=214)
```

© Andrzej M. Borzyszkowski

Bazy Danych

8

Zagnieżdżenie w atrybucie wynikowym

- **EXPLAIN SELECT nr, (SELECT sum(ilosc) AS razem
FROM pozycja WHERE
towar_nr=towar.nr)**

FROM towar

QUERY PLAN Seq Scan on towar (cost=0.00..13291.50 rows=390 width=12)

SubPlan 1

-> Aggregate (cost=34.04..34.05 rows=1 width=8)

-> Bitmap Heap Scan on pozycja (cost=23.45..34.01 rows=10 width=4)

Recheck Cond: (towar_nr = towar.nr)

-> Bitmap Index Scan on pozycja_pk (cost=0.00..23.45 rows=10 width=0)

© Andrzej M. Borzyszkowski

Bazy Danych

Zagnieżdżenie w atrybucie wynikowym vs funkcja agregująca

- **EXPLAIN SELECT towar_nr, sum(ilosc) AS razem
FROM pozycja
GROUP BY towar_nr**

QUERY PLAN HashAggregate (cost=40.60..42.60 rows=200 width=12)

Group Key: towar_nr

-> Seq Scan on pozycja (cost=0.00..30.40 rows=2040 width=8)

- ale pamiętamy, że wynik tego zapytania jest uboższy niż poprzedniego, wyświetlane są wyłącznie numery towarów zamawianych

© Andrzej M. Borzyszkowski

Bazy Danych

Instrukcja SELECT – brak kwantyfikatora ogólnego w języku SQL

- Podaj nazwiska klientów, którzy zamówili każdy towar w dostępny ofercie:

SELECT nr, imie, nazwisko

FROM klient K

WHERE NOT EXISTS

-- nie istnieje

(SELECT *

FROM towar T

WHERE NOT EXISTS

-- towar niezamawiany

(SELECT *

FROM zamowienie Z INNER JOIN pozycja ON

Z.nr=zamowienie_nr AND

K.nr = klient_nr AND T.nr = towar_nr

))

- SQL nie ma konstrukcji dla kwantyfikatora ogólnego FORALL, konieczne podwójne przeczenie dla EXISTS

© Andrzej M. Borzyszkowski

Bazy Danych

Instrukcja SELECT – kwantyfikator ogólny, inne rozwiązanie

- Podaj nazwiska klientów, którzy zamówili każdy towar w dostępny ofercie:

SELECT nr, imie, nazwisko

FROM klient K

WHERE NOT EXISTS

-- nie istnieje

((SELECT nr FROM towar)

-- wszystkie towary

EXCEPT

-- oprócz

(SELECT towar_nr

-- towary tego klienta

FROM zamowienie Z INNER JOIN pozycja ON

Z.nr = zamowienie_nr AND K.nr = klient_nr

))

- użycie operatora teoriomnogościowego
– nie każdy system baz danych implementuje EXCEPT

© Andrzej M. Borzyszkowski

Bazy Danych

Instrukcja SELECT – kwantyfikatory ogólny, teoretyczne rozwiązanie

- Podaj nazwiska klientów, którzy zamówili każdy towar w dostępny ofercie:

```
SELECT nr, imie, nazwisko  
FROM klient K WHERE  
(( SELECT towar_nr -- towary zamawiane  
FROM zamowienie Z INNER JOIN pozycja ON  
Z.nr=zamowienie_nr AND K.nr = klient_nr  
)  
CONTAINS  
( SELECT nr FROM towar ) -- wszystkie towary  
)
```

- SQL nie pozwala na porównanie tabel
- oryginalny system R firmy IBM posiadał implementację predykatu zawierania

13

© Andrzej M. Borzyszkowski
Bazy Danych

Instrukcja SELECT – brak kwantyfikatora ogólnego w języku SQL, c.d.

- Podaj nazwiska klientów, którzy zamówili każdy towar w dostępny ofercie:

```
SELECT nr, imie, nazwisko  
FROM klient K WHERE  
(SELECT count (DISTINCT towar_nr) -- towary zamawiane  
FROM zamowienie Z INNER JOIN pozycja ON  
Z.nr=zamowienie_nr AND K.nr = klient_nr  
)  
=  
( SELECT count(nr) FROM towar -- wszystkie towary  
)
```

- korzystamy z tego, że każdy towar z tabeli pozycji musi występować w tabeli towarów (integralność referencyjna)
- stąd porównanie liczebności gwarantuje zawieranie

14

© Andrzej M. Borzyszkowski
Bazy Danych

Przykład „brak kwantyfikatora ogólnego w języku SQL” – wydajność

- Podwójne przeczenie dla EXISTS:

```
QUERY PLAN Nested Loop Anti Join (cost=0.00..2051005.92 rows=85  
width=136)
```

```
Join Filter: (NOT (SubPlan 1))
```

```
-> Seq Scan on klient k (cost=0.00..11.70 rows=170 width=136)
```

- Zagnieżdżenie skorelowane wersja 1 z EXCEPT:

```
QUERY PLAN Seq Scan on klient k (cost=0.00..47.55 rows=85  
width=136)
```

```
Filter: (NOT (SubPlan 1))
```

```
SubPlan 1
```

```
-> HashSetOp Except (cost=0.00..82.25 rows=390 width=8)
```

- Zagnieżdżenie skorelowane wersja 2 z równością:

```
QUERY PLAN Seq Scan on klient k (cost=14.88..10803.08 rows=1  
width=136)
```

```
Filter: ((SubPlan 1) = $1)
```

```
InitPlan 2 (returns $1)
```

```
-> Aggregate (cost=14.88..14.88 rows=1 width=8)
```

15

© Andrzej M. Borzyszkowski
Bazy Danych

Instrukcja SELECT – zagnieżdżenia, warunek IN

- Podaj dane klientów, których imiona i nazwiska się powtarzają:

```
SELECT imie, nazwisko, miasto  
FROM klient  
WHERE ( imie, nazwisko ) IN (  
SELECT imie, nazwisko  
FROM klient  
GROUP BY imie,nazwisko  
HAVING count (nazwisko) > 1  
)
```

- warunek należenia do zbioru stosowany jest do par wartości
- tabela zwracana w zapytaniu podrzędnym ma tyle kolumn, ile wartości w klauzuli WHERE

16

© Andrzej M. Borzyszkowski
Bazy Danych

Instrukcja SELECT – zagnieżdżenia, porównania z wartościami zagregowanymi

- Podaj dane o towarach o koszcie powyżej przeciętnej:
SELECT *
FROM towar
WHERE koszt > (
 SELECT avg(koszt) FROM towar
)
 - brak korelacji, nie ma konieczności zmiany nazwy
 - tabela wynikowa z zapytaniu podrzędnym (1x1) jest traktowana jak pojedyncza wartość

© Andrzej M. Borzyszkowski
Bazy Danych

17

Instrukcja SELECT – porównania z wartościami zagregowanymi c.d.

- Podaj dane o towarach o koszcie maksymalnym:
SELECT * FROM towar
WHERE koszt = (
 SELECT max(koszt) FROM towar
)
 - tabela 1x1, czyli pojedyncza wartość i porównanie z tą wartością
- Inne rozwiązanie
SELECT * FROM towar
WHERE koszt >= ALL (
 SELECT koszt FROM towar
)
 - tabela o jednej kolumnie, czyli zbiór i porównanie z całym zbiorem

© Andrzej M. Borzyszkowski
Bazy Danych

18

Porównania z wartościami zagregowanymi – wydajność

- Porównanie z pojedynczą wartością
EXPLAIN SELECT * FROM towar
WHERE koszt = (
 SELECT max(koszt) FROM towar
)
QUERY PLAN Seq Scan on towar (cost=14.88..29.76 rows=2 width=178)
Filter: (koszt = \$0)
- Porównanie z całym zbiorem
EXPLAIN SELECT * FROM towar
WHERE koszt >= ALL (
 SELECT koszt FROM towar
)
QUERY PLAN Seq Scan on towar (cost=0.00..3295.75 rows=195 width=178)
Filter: (SubPlan 1)

© Andrzej M. Borzyszkowski
Bazy Danych

19