Uniwersytet Wrocławski

Bazy danych 2022

Wykład 4

Operacje mnogościowe, samozłączenia, podzapytania, perspektywy

Bartosz Brzostowski

Wydział Fizyki i Astronomii UWr
semestr letni r. akad. 2021/22

24 marca 2022

Dygresja: operacje mnogościowe

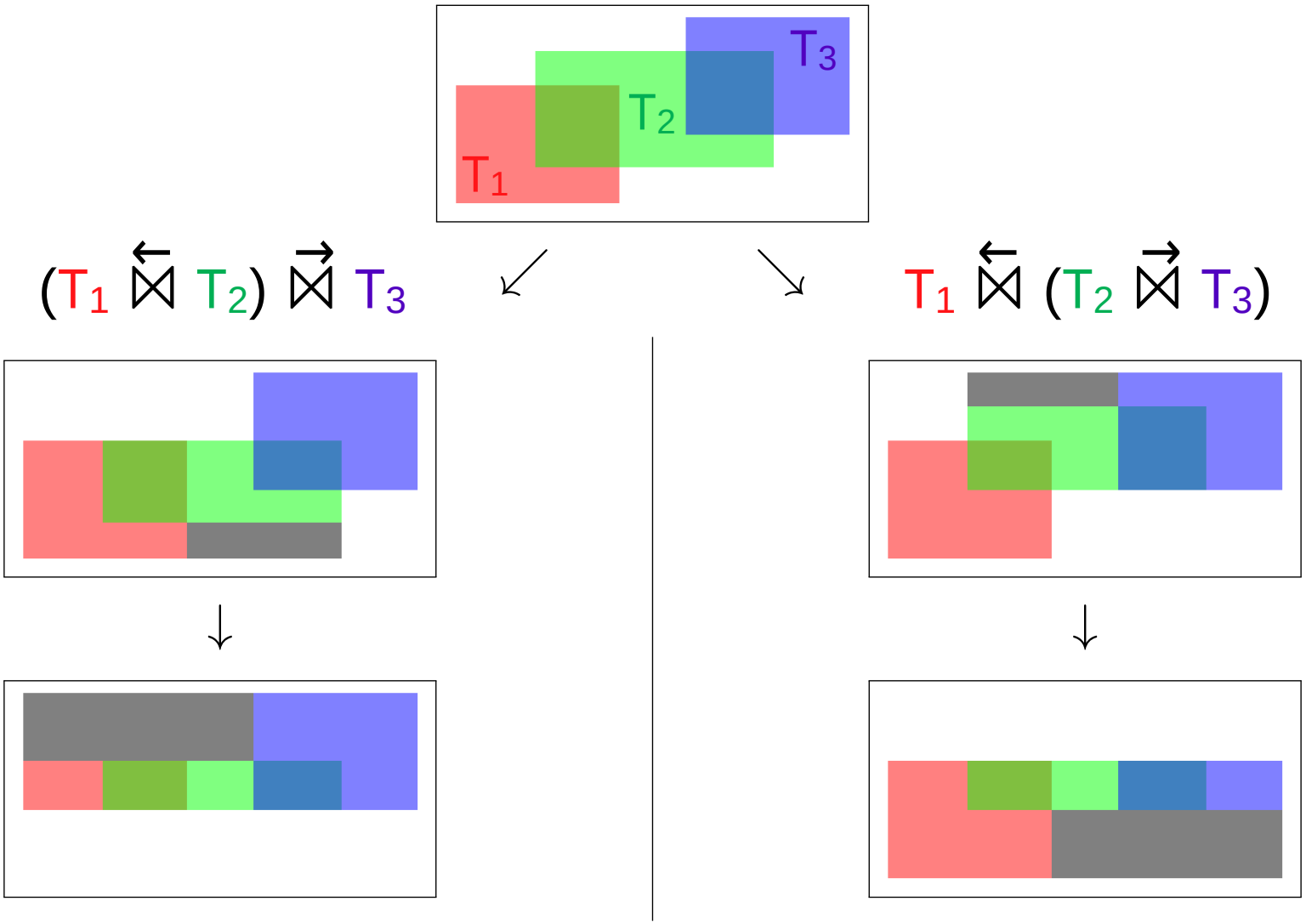
- Iloczyn kartezjański wziął się z teorii zbiorów
- Inne operacje na zbiorach: suma, przekrój, różnica
- Oczywiście możliwe również dla relacji (tabel)… o ile zgodne schematy
- Wszystkie trzy są zaimplementowane… w PostgreSQLu
- MySQL: `UNION` (oraz `UNION ALL` nie usuwająca duplikatów)
- Operacja nie na tabelach, tylko na wynikach zapytań: `SELECT * FROM [t1] UNION SELECT * FROM [t2]`
- Przekrój i różnicę można zrealizować inaczej (o czym dalej)

Złączenia zewnętrzne — komplikacje

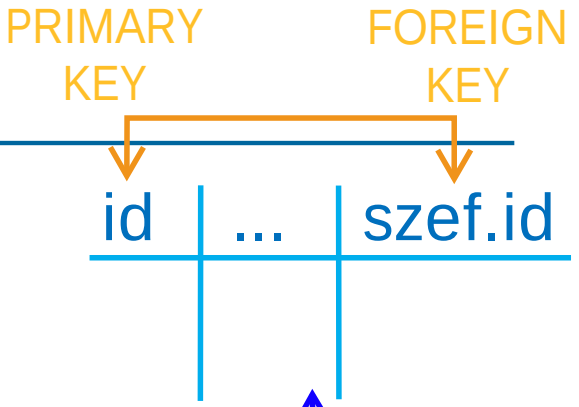
- Czy możliwe jest „obustronne” złączenie zewnętrzne? Tak!
- `FULL OUTER JOIN...` w PostgreSQLu
- MySQL: `UNION` wyników `SELECT`ów z `LEFT` oraz `RIGHT JOIN`
- Złączenia są działaniami łącznymi, o ile rozpatrujemy tylko jeden rodzaj
- „Mieszane” złączenia mogą dawać różne wyniki w zależności od wstawienia nawiasów:

  [t1] `LEFT JOIN` *[t2]* `RIGHT JOIN` *[t3]*

[t1] LEFT JOIN [t2] RIGHT JOIN [t3]



Samozłączenia



- Tabele można łączyć same ze sobą
- Szukamy par „pracownik–podwładny” (klucz obcy wskazujący na kolumnę tej samej tabeli)
- Szukamy (znowu) najdroższego produktu, rozważając pary cen: jeśli od danej ceny jest tylko jedna większa lub równa (tj. ona sama), to znaczy, że jest maksymalna
- Szukamy klienta, który zamówił dwa określone produkty (zwłaszcza bez możliwości policzenia przekroju zbiorów...)
- Zazwyczaj konieczne aliasowanie: `SELECT t1.k1, t2.k2 FROM tab AS t1 JOIN tab AS t2 ON t1.k3 = t2.k3`

Podzapytania

- Zagnieżdżone wyrażenia `SELECT`, w nawiasach
- Mogą występować w każdej klauzuli (zewnętrznego) zapytania (w standardzie — poza `GROUP BY` i `ORDER BY`, w MySQL również tam, ale to dosyć wydumane)
- W klauzuli `FROM`: złączane jak zwykłe tabele...

```
SELECT nazwa, sztuk
  FROM produkty NATURAL LEFT JOIN (
    SELECT p_id AS idp, sztuk
      FROM detal_zamow
      WHERE sztuk > 1
    ) AS foo
  WHERE cena < 2000;
```

- ... ale wymagany alias
- Ułatwiają pisanie zapytań
- Ograniczają rozmiar pośrednich wyników obliczeń
- Zwiększają siłę wyrazu języka: kilkustopniowa agregacja

Nierówności na krotkach:
→ porządek leksykograficzny
(p.id, sztuk) < (P, S) oznacza:
- p.id < P lub
- p.id = p oraz sztuk < S

i analogicznie dla dłuższych

Podzapytania w klauzulach `WHERE` i `HAVING`

- Dzielą się na *wierszowe* i *tablicowe*
- Wierszowe zwracają (najwyżej) jeden wiersz
- Skąd to wiadomo? Agregacja, klucze
- Wyniki wchodzą do warunków filtrowania

```
SELECT nazwa FROM produkty
  WHERE cena > 4 * (SELECT MIN(cena) FROM produkty);
```

- Niekoniecznie muszą zwracać jedną kolumnę

```
SELECT idd FROM detal_zamow
  WHERE (p_id, sztuk) = (
    SELECT idp, ilosc
      FROM produkty
      WHERE idp = 5
    );
```

- NULL dla pustego wyniku podzapytania

Podzapytania tablicowe

- Zwracają (potencjalnie) wiele wierszy
- Nowe operatory: `[NOT] IN`, `ANY`, `ALL`, `EXISTS`
- `IN` sprawdza, czy coś należy do wyniku podzapytania

```
SELECT nazwa FROM produkty
  WHERE (LEFT(nazwa, 1), cena) IN (
    SELECT ANY_VALUE(LEFT(nazwa, 1)), MAX(cena)
      FROM produkty
      GROUP BY LEFT(nazwa, 1)
    );
```

Operatory ANY i ALL

- Składnia:
  [wyr] *[porównanie]* [ANY | ALL] (*[podzapytanie]*)
- $x \sim \texttt{ANY}\ S \quad \Leftrightarrow \quad \exists y \in S \quad x \sim y$
  $x \sim \texttt{ALL}\ S \quad \Leftrightarrow \quad \forall y \in S \quad x \sim y$
  gdzie ~ oznacza: =, <, <=, >=, >, <=>
- ```
  SELECT * FROM detal_zamow
    WHERE sztuk >= ALL (SELECT sztuk FROM detal_zamow);
  ```
- IN to = ANY, NOT IN to <> ALL
- ALL jest bardziej „restrykcyjny” niż ANY
- . . . chyba, że wynik podzapytania pusty: ANY zwraca fałsz, ALL — prawdę

$\forall\, y{\in}S\ \varphi$, $\exists\, y{\in}S\ \varphi$ } to skrót dla { $\forall_y (y{\in}S \Rightarrow \varphi)$, $\exists_y (y{\in}S \wedge \varphi)$ } a gdy $S = \emptyset$ to $y \in S$ jest Fałszem, zaś { $F \Rightarrow \varphi$ prawdą, $F \wedge \varphi$ fałszem } niezależnie od φ
  ```

Operator `EXISTS`

- Jednoargumentowy, sprawdza, czy wynik jest pusty

```
SELECT nazwa FROM produkty
  WHERE NOT EXISTS (SELECT * FROM detal_zamow WHERE idp = p_id);
```

- Nie ma znaczenia, co `SELECT`ujemy w podzapytaniu
- Zazwyczaj podzapytanie zależy od wartości z zapytania zewnętrznego (inaczej jego wynik byłby stały)
- Podzapytania *skorelowane* (nie tylko przy `EXISTS`): obliczane potencjalnie raz dla każdego wiersza zapytania zewnętrznego (nieskorelowane: raz na zapytanie)
- Potencjalnie nieefektywne! (choć może czasem nie do uniknięcia)

Podzapytania cd.

- Można je zagnieżdżać wielokrotnie (podzapytania w podzapytaniach)
- Konieczność precyzyjnych odwołań do (aliasowanych) tabel, zwłaszcza dla skorelowanych:

najbliższa 'foo.cena' jest poziom wyżej

```
SELECT nazwa FROM produkty AS foo
  WHERE NOT EXISTS (SELECT * FROM produkty WHERE cena > foo.cena);
```

Udało się znaleźć 'cena' w tym samym nawiasie

- Mogą występować w klauzuli `SELECT` — oczywiście wierszowe, w dodatku jednokolumnowe, zazwyczaj skorelowane

```
mysql> SELECT nazwa,
    ->   (SELECT MAX(data) FROM zamow WHERE k_id = idk)
    ->   FROM klienci WHERE LENGTH(nazwa) < 6;
+------+-----------------------------------------+
| nazwa | (select max(data) from zamow where k_id = idk) |
+------+-----------------------------------------+
...
```

- Z podobnymi zastrzeżeniami także w `GROUP BY`, `ORDER BY`

Perspektywy

- Element DDL, ale upraszcza tworzenie zapytań
- „Tabele wirtualne”
- `CREATE [OR REPLACE] VIEW` *`[nazwa]`* `AS` *`[zapytanie]`*
- Nazwy kolumn: opcjonalna lista `(`*`[kol1]`*`,` *`[kol2]`*`, ...)` albo wynikające z zapytania (aliasy)
- ```
  mysql> CREATE VIEW kli_zam_popoludn AS
      ->   SELECT * FROM klienci JOIN zamow ON idk = k_id
      ->     WHERE hour(data) >= 12;
  Query OK, 0 rows affected (0,02 sec)
  ```
- Intuicja: często używane podzapytania możemy „zapakować” w perspektywę
- Widoczne w `SHOW TABLES` choć tabelami nie są
- Usuwanie: `DROP VIEW [IF EXISTS]` *`[nazwa]`*`, ...`
  ```