Wykład 3: Agregacja i nulle

Po co się męczyć z algebrą? Żeby dobrze zrozumieć semantykę SQLa:

```
R cap (S cup T)
select R.a from R, S, T where R.a = S.a or R.a = T.a
select R.a from R, (S union T) as U where R.a = U.a
```

Multizbiory i agregacja

Dlaczego multizbiory?

- Łatwiej: suma to konkatenacja, nie trzeba usuwać duplikatów; przy rzutowaniu wypisujemy po kolei, bez sprawdzania, czy coś już było, czy nie.
- Jak policzyć średnią zarobków? Nie można wziąć zbioru wszystkich zarobków i zastosować operatora AVG; trzeba uwzględnić krotności!

Semantyka podstawowych operacji

- suma i rzutowanie było wyżej
- różnica wiadomo: różnica liczby wystąpień
- przecięcie - minimum
- selekcja - po kolei jedziemy po krotkach i wypisujemy jak się nadaje
- produkt: jedziemy podwójną pętlą

Uwaga: nie wszystkie prawa na zbiorach działają też na multizbiorach:

- `(R cap S) - T = R cap (S - T)`
- `R cap (S cup T) = (R cap S) cup (R cap T)`
- `(R cup S) - T = (R - T) cup (S - T)`
- `select * from R where (C or D) = (select * from R where C) union all (select * from R where D)`

Operatory

- usuwanie duplikatów: `select distinct * from R`
- agregacja: `MIN, MAX, SUM, COUNT, AVG`
- grupowanie: operator grupowania w wygodny sposób umożliwia agregację.

Tak naprawde: listy krotek; sortowanie.

Nulle w SQLu

Co reprezentuje null?

1. nieznana/brakująca wartość: np. brakująca data urodzenia
2. nieistniejąca wartość: np. mąż dla kobiety niezamężnej
3. ukryta wartość: np. zastrzeżony numer telefonu (w sumie podobna do 1, ale można żądać, żeby była podobna do 2 - nie chcemy, żeby ktoś cokolwiek wnioskował z naszego numeru telefonu, ale również z tego, że posiadamy numer telefonu, albo że go nie posiadamy).

Kiedy SQL produkuje nulle?

- `AVG(A)` na pustym zbiorze krotek. Tak samo `MIN, MAX, SUM` (czemu nie 0?). Tylko `COUNT` daje 0.
- produkt zewnętrzny
  - `select * from R, S where R.A=S.A;` niesparowane wiersze się gubią,
  - `select * from R full outer join S on R.A=S.A;` parujemy z wierszem nulli (po lewej lub prawej)
  - `select * from R left outer join S on R.A=S.A;` tylko lewe, a prawe się gubią

Reguły w SQLu:

1. dowolna operacja arytmetyczna na nullu daje null
2. dowolne porównanie na nullu daje UNKNOWN (wartość logiczna ½), koniunkcja=min, alternatywa=max, negacja=1-x, bierze się krotki o wartości TRUE (wartość 1).

Ale są pewne dziwności:

- `SUM(A)` sumuje tylko nie-nulle; **niezgodność z (1)**
- `COUNT(A)` zlicza tylko nie-nulle, count(*) zlicza wszystkie krotki; **niezgodność wewnętrzna:** co jak jest tylko jedna kolumna?
- `select a, avg(b) from R group by a;` daje jeden wiersz z nullem w kolumnie a; **niezgodność z (2)**, bo przyjmujemy, że nulle są sobie równe.
- w `OUTER JOIN` wiersze z nullami nie spełniają warunku `ON`; **niezgodność z (2)**
- Można powiedzieć, że to kwestia wyboru semantyki, i że tak jest po prostu wygodniej (faktycznie zwykle chcemy mieć avg z wartości, które znamy), ale taka semantyka jest bardziej skomplikowana, trudniej się jej nauczyć.

Paradoksy:

- `int x;` jeśli `x` jest null, to `0*x` też jest null. A nie zero, mimo że dowolna wartość pomnożona przez 0 daje 0. Bez sensu?

- `select * from Movie where length < 120 or length >= 120;` nulle się nie wybiorą, bo dają unknown. (Bez sensu? Ale jak pytamy o wzrost męża, a dana kobieta jest niezamężna, to pewnie lepiej, żeby jej nie wybierało…)
- `SELECT R.a FROM R WHERE R.a NOT IN (SELECT a FROM S);` Na R.a = {1,2,3,4}, S.a = {1}, daje {2,3,4}; ale jak dodamy null do S.a, S.a = {1, null}, zapytania daje {}. Jeden element wyrzuca trzy elementy? Problem: Ludzie nie myślą w logice 3-wartościowej…
- Można powiedzieć, że taka jest semantyka i że to nie jest paradoks. Ale na pewno jest to dysonans poznawczy! Zachowanie sprzeczne z intuicją zwiększa liczbę błędów!

Niebezpieczne konsekwencje paradoksów:

```
SELECT M.#, M.target
FROM Missiles M
WHERE
     M.target IN (SELECT Name FROM USCities) AND
     M.# NOT IN (SELECT I.Missile FROM Intercept I
                 WHERE I.Status='active')
```

Przypuśćmy, że jedna anty-rakieta nie była zainicjalizowana poprawnie, ale została aktywowana...

Semantyka pewnych odpowiedzi (*)

- Prawdą jest to, co jest prawdziwe po dowolnym ustawieniu nulli na wartości.
- `select * from Movie where length >= 120 or length < 120` zwraca wszystko.
- Paradoksy dalej sa: `SELECT R.a FROM R WHERE R.a NOT IN (SELECT a FROM S)` daje takie odpowiedzi jak wcześniej. Dla `SELECT count(R.a) FROM R WHERE R.a NOT IN (SELECT a FROM S)` wcale nie wiadomo, co zwrócić, bo mamy albo 3 albo 2; chyba null :-)
- Drugi problem: to odpowiada tylko pierwszej roli nulla (brakująca/nieznana wartość). A jak sobie radzić, z wartościami nieistniejącymi? (Może oddzielne tabele? Albo wprowadzić specjalnego nulla “atrybut nie istnieje”)
- Trzeci problem: trudno obliczyć odpowiedź w tej semantyce. Intuicyjnie, jeśli nie wiemy nic szczególnego na temat zapytania, to chyba trzeba rozważać dowolne podstawienia wartości pod nulle - wykładniczo dużo…
- Formalnie: można łatwo pokazać, że ten problem jest NP-trudny, tzn. gdyby istniał algorytm rozwiązujący go w czasie wielomianowym, to byłyby też

algorytmy wielomianowe dla wielu słynnych problemów, dla których nie znamy takich algorytmów: np. problem komiwojażera, problem plecakowy, problem SAT.

- Da się dla klasy zapytań pozytywnych, czyli SPCU (algebra relacji bez różnicy):
    - każdego nulla zastępujemy specjalną wartością, inną niż wszystkie inne, odróżnialną od normalnych stałych;
    - obliczamy zapytanie traktując te nowe wartości jak zwykłe stałe;
    - wyrzucamy z odpowiedzi wszystkie krotki używające tych specjalnych wartości;
- Pewne odpowiedzi stosuje się przy wymianie danych i integracji danych. (Więcej np. na przedmiocie *Teoria baz danych*).