Bazy Danych

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytetu Gdańskiego

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

Język SQL – wartość nieokreślona NULL

Wartość NULL

- Wartość nieznana w tej chwili
  - np. klienci, których telefon jest nieznany
- Wartość nie mogąca mieć sensu w danym kontekście
  - np. tabela książek z kluczem obcym wskazującym na aktualnego czytelnika i datą wypożyczenia
  - jeśli książka nie jest wypożyczona, to klucz obcy jest NULL
  - ale wówczas data wypożyczenia nie ma sensu, też musi być NULL

Wartość NULL, c.d.

- Konieczność, gdy jedna tabela realizuje dwie encje połączone związkiem jedno-jednoznacznym, np.

```
CREATE TABLE przedmiot_termin (
   Kod       integer   PRIMARY KEY,
   Rodzaj    varchar(20)   not null,
   Nazwa     varchar(50)   not null,
   Dzien     int,
   Godzina   int,
   Sala      int,
   CONSTRAINT  UNIQUE ( dzien_tyg, godzina, sala )
)
```

  - przy odrębnych tabelach przedmiot mógł nie być adresatem klucza obcego z tabeli terminów
  - ale w jednej tabeli przedmiot występuje i termin musi być zastąpiony NULLem

Własności wartości NULL

- Klucz kandydujący
  - SQL/92: taka sama wartość jak inne, a więc może wystąpić w tabeli najwyżej jeden raz
  - mało sensowne podejście – tylko jeden klient może być bez telefonu, tylko jedna książka niewypożyczona
  - PostgreSQL, i wiele innych: wartość nieznana, a więc wiele wystąpień NULL nie narusza warunku na klucz kandydujący
- Klucz główny: wartość NULL nie jest dozwolona wcale
- Można sprawdzać tę wartość:

```
SELECT nazwisko, telefon
FROM  klient
WHERE  telefon IS NOT NULL
```


Własności wartości NULL, c.d.

- Można jawnie wprowadzać tę wartość:

```
INSERT INTO towar ( opis, koszt, cena )
VALUES ( E'ramka do fotografii 3\'x4\'', 13.36, NULL )
```

- Wartość NULL nie pasuje do żadnego wzorca
  - załóżmy, że tabela klientów ma atrybut logiczny „zaległosc”

```
SELECT * FROM klient
  WHERE zaleglosc = TRUE OR zaleglosc = FALSE
```

  - nie wykaże wszystkich klientów, jedynie tych z określoną wartością tego atrybutu

```
SELECT * FROM klient
```

  - wykaże wszystkich klientów, również z niekreśloną wartością atrybutu

```
SELECT * FROM klient
  WHERE zaleglosc != NULL
```

  - jest absolutnie błędne, działania z NULL nigdy nie zwrócą wartości

Instrukcja SELECT – złączenie zewnętrzne

- **SELECT K.nr, nazwisko, imie, data_zlozenia**
  FROM klient K INNER JOIN zamowienie
  ON K.nr = klient_nr
  - złączenie wewnętrzne, klienci, którzy złożyli zamówienia
- **SELECT K.nr, nazwisko, imie, data_zlozenia**
  FROM klient K LEFT OUTER JOIN zamowienie
  ON K.nr = klient_nr
  - złączenie zewnętrzne, klienci i ich zamówienia, nawet jeśli ich nie złożyli
  - Słowo **OUTER** jest opcjonalne
  - brakujące dane będą uzupełnione NULL
  - atrybut nr w warunku złączenia występuje w obu tabelach, musi być poprzedzony nazwą tabeli

Złączenie zewnętrzne, przykład

nr	*nazwisko*	*imie*	*data_zlozenia*
1	Kuśmierek	Małgorzata	
2	Chodkiewicz	Jan	
3	Szczęsna	Jadwiga	23.02.2025
3	Szczęsna	Jadwiga	13.02.2025
3	Szczęsna	Jadwiga	23.01.2025
4	Łukowski	Bernard	22.02.2025
4	Łukowski	Bernard	1.02.2025
5	Soroczyński	Jan	4.02.2025
6	Niezabitowska-Nasiadko	Marzena	
7	Kołak	Agnieszka	
8	Kołak	Agnieszka	12.01.2025
8	Kołak	Agnieszka	7.01.2025

Negatywne zapytanie raz jeszcze

- Podaj nazwiska klientów, którzy nie złożyli zamówienia wcale

```
SELECT nazwisko
  FROM klient K LEFT JOIN zamowienie Z
  ON K.nr=klient_nr WHERE Z.nr IS NULL
```

  - albo można spytać **WHERE klient_nr IS NULL**
  - ani klucz główny w tabeli nie może być NULL ani klucz obcy jeśli był zadeklarowany jako NOT NULL
  - są to sztucznie dodane wartości nieokreślone do krotek z lewej tabeli, które nie mają pary w prawej tabeli

Złączenie zewnętrzne, inny przykład

- Podaj informacje o towarach, których w magazynie jest mało *lub wcale*

```
SELECT * FROM towar INNER JOIN zapas ON nr =
  towar_nr
WHERE ilosc < 10
UNION
SELECT * FROM towar
WHERE nr NOT IN ( SELECT towar_nr FROM zapas )
```

  - ERROR: each UNION query must have the same number of columns
- **SELECT T.* FROM towar T INNER JOIN zapas ON nr = towar_nr**
  - nie wyświetli informacji o stanie zapasów, tylko o towarze

Złączenie zewnętrzne, inny przykład c.d.

- Podaj informacje o towarach, których w magazynie jest mało *lub wcale*

```
SELECT *
FROM towar LEFT OUTER JOIN zapas
ON nr = towar_nr AND ilosc < 10
```

  - towary, których nie ma w magazynie i dla których drugi warunek nie ma sensu, znajdą się jednak w wyniku
  - AND jest dalszym warunkiem złączenia zewnętrznego
- **SELECT * FROM towar LEFT OUTER JOIN zapas ON nr = towar_nr WHERE ilosc < 10**
  - towary, których nie ma w magazynie i dla których drugi warunek nie ma sensu, nie mogą spełnić warunku WHERE

Złączenie zewnętrzne, 3

- Towary i sumy ich zamówień (nawet jeśli ich brak)

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

  - zagnieżdżenie skorelowane skutkuje słabą wydajnością
-

```
SELECT nr, sum(ilosc) AS razem
FROM towar LEFT JOIN pozycja ON towar_nr=nr
GROUP BY nr;
```

  - ten sam skutek z radykalnie lepszą wydajnością

Negatywne zapytanie raz jeszcze

- Podaj dane klientów, którzy nie złożyli zamówienia po 1 lutego 2025:

```
SELECT K.nr, imie, nazwisko
FROM klient K LEFT JOIN zamowienie Z
ON K.nr=klient_nr AND data_zlozenia > '2025-2-1'
WHERE Z.nr IS NULL
```

  - dane klientów, którzy nie złożyli takiego zamówienia są uzupełnione NULL, o który można zapytać

Klucz obcy z możliwą wartością NULL

- Wypisz opisy towarów z nieokreślonym kodem kreskowym

```
SELECT opis FROM towar
   WHERE nr NOT IN (SELECT towar_nr FROM kod_kreskowy)
```

  - nie ma takich towarów ?
- Wypisz opisy towarów z określonym kodem kreskowym

```
SELECT opis FROM towar
   WHERE nr IN (SELECT towar_nr FROM kod_kreskowy)
```

- Wersja A: Operacja z użyciem NULL zawsze zwraca wartość nieokreśloną
  - wewnętrzny SELECT zawiera wartość NULL
  - a więc oba warunki IN oraz NOT IN mają wartość nieokreśloną
- Wersja B: Operacja IN w kontekście zbioru zawierającego NULL zwraca wartość *true* jeśli element występuje w zbiorze
  - ale wartość *nieokreśloną* jeśli nie występuje

Klucz obcy z możliwą wartością NULL, c.d.

- Postgres dawne wersje, standard SQL literalny – wersja A
  - Postgres 9.3 – wersja B
  - inne systemy – ???
- Zawsze dobrym rozwiązaniem jest wykluczyć wartość NULL w porównaniach

```
SELECT opis
FROM towar WHERE nr NOT IN (
        SELECT towar_nr FROM kod_kreskowy
        WHERE towar_nr IS NOT NULL )
```

  - klucz obcy najczęściej nie może być NULLem, więc ta ostrożność najczęściej nie ma znaczenia
  - jeśli dopuszczamy NULL, zawsze trzeba go wykluczyć w kontekście NOT IN

Klucz obcy z możliwą wartością NULL, c.d.

- Nie ma powodu do ostrożności w zapytaniu

```
SELECT opis FROM towar T
   WHERE NOT EXISTS (
    SELECT * FROM kod_kreskowy
    WHERE towar_nr = T.nr )
```

  - jeśli klucz obcy ma jakąkolwiek wartość to taki klucz obcy nie będzie miał wartości NULL
  - nie ma potrzeby osobno jej wykluczać

Złączenie zewnętrzne, c.d.

- Jeśli klucz obcy dopuszcza NULL, wówczas brak dopasowania może wystąpić też po stronie tabeli zawierającej taki klucz obcy
- Oprócz **LEFT OUTER JOIN** istnieją również wersje **RIGHT OUTER JOIN** oraz **FULL OUTER JOIN**
  - chroniona jest lewa lub prawa lub obie tabele
  - wiersze z chronionej tabeli wejdą do wyniku nawet jeśli nie będą miały pasującego wiersza z drugiej tabeli
  - brakujące atrybuty będą miały wartość NULL

Różne złączenia zewnętrzne

- **SELECT T.*, kod FROM towar T INNER JOIN kod_kreskowy ON nr = towar_nr;**
  - złączenie wewnętrzne, towary, które mają kody kreskowe
- **SELECT T.*, kod FROM towar T LEFT JOIN kod_kreskowy ON nr = towar_nr;**
  - złączenie zewnętrzne lewe, wszystkie towary i ich kody, jeśli je posiadają
- **SELECT T.*, kod FROM towar T RIGHT JOIN kod_kreskowy ON nr = towar_nr;**
  - złączenie zewnętrzne prawe, wszystkie kody i odpowiadające im towary, o ile klucz obcy na nie wskazuje
- **SELECT T.*, kod FROM towar T FULL JOIN kod_kreskowy ON nr = towar_nr;**
  - złączenie zewnętrzne pełne, wszystkie i towary i kody

Własności wartości NULL dla funkcji agregujących

- Funkcje agregujące pomijają wiersze z wartością NULL

```
SELECT count ( telefon )
FROM klient
```

obliczy liczbę klientów z określoną wartością numeru telefonu

- zestaw wszystkich atrybutów na pewno ma określoną wartość

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

obliczy liczbę wszystkich klientów

  - na pewno istnieje atrybut ukryty OID
  - w praktyce nie ma różnicy, jakiś atrybut jest zadeklarowany jako klucz główny i jest on określony

Własności wartości NULL dla funkcji agregujących, c.d.

- Łączna cena trzech podanych towarów z tabeli:

```
SELECT sum(cena)
FROM towar
WHERE nr IN (10,11,12)
```

  - sumowane będą tylko ceny określone
  - jeśli któraś z nich będzie NULL, to efektywnie będzie zero
- **SELECT (SELECT cena FROM towar WHERE nr=10)**
  + (SELECT cena FROM towar WHERE nr=11)
  + (SELECT cena FROM towar WHERE nr=12)
  - będzie nieokreślona, jeśli choć jedna z nich będzie NULL