Relacyjne Bazy Danych

Andrzej M. Borzyszkowski
PJATK/ Gdańsk

materiały dostępne elektronicznie
http://szuflandia.pjwstk.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          serial      PRIMARY KEY,
   rodzaj       varchar(20)  not null,
   nazwa        varchar(50)  not null,
   dzien_tyg    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

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 *nieokreślona* 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 nie jest niezbędna
  - jeśli dopuszczamy NULL, zawsze trzeba go wykluczyć w kontekście negacji

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

- * nie ma na pewno wartości niekreślonej

```
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 jedna 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

Złączenie w SQL a wartości NULL

- 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 w SQL a wartości NULL – złączenie zewnętrzne

- 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 w wyniku

- Istnieją też wersje `RIGHT OUTER JOIN` oraz `FULL OUTER JOIN`
  - chroniona jest lewa lub prawa lub obie tabele
  - tzn. wiersze z chronionej tabeli wejdą do wyniku nawet jeśli nie będą miały pasującego wiersza z drugiej tabeli
  - brakujące atrybuty będą miały wartość NULL
  - słowo `OUTER` jest zbędne i może być pominięte

Złączenie zewnętrzne, przykład

nr	opis	koszt	cena
2	układanka typu puzzle	16.43	19.99
5	chusteczki higieniczne	2.11	3.99
10	moneta srebrna z Papieżem	20.00	20.00
19	zegarek męski	26.43	
11	torba plastikowa	0.01	0.00
17	donica duża	26.43	
12	nożyczki drewniane	8.18	

towar_nr	ilosc
2	2
5	3
10	1
19	1

nr	opis	koszt	cena	towar_nr	ilosc
2	układanka typu puzzle	16.43	19.99	2	2
5	chusteczki higieniczne	2.11	3.99	5	3
10	moneta srebrna z Papieżem	20.00	20.00	10	1
19	zegarek męski	26.43		19	1
11	torba plastikowa	0.01	0.00		
17	donica duża	26.43			
12	nożyczki drewniane	8.18			

Instrukcja SELECT – negatywne zapytanie jeszcze raz

- Podaj nazwiska klientów, którzy założyli zamówienie po 1 marca 2021:

```
SELECT nazwisko
FROM klient K INNER JOIN zamowienie
ON K.nr = klient_nr AND data_zlozenia > '2021-3-1'
```

- Podaj nazwiska klientów, którzy nie założyli zamówienia po 1 marca 2021 ?

```
SELECT nazwisko
FROM klient K LEFT OUTER JOIN zamowienie Z
ON K.nr = klient_nr AND data_zlozenia > '2021-3-1'
WHERE Z.klient_nr is NULL
```