Relacyjne Bazy Danych

Andrzej M. Borzyszkowski
PJATK/ Gdańsk

materiały dostępne elektronicznie
http://szuflandia.pjwstk.edu.pl/~amb

Język SQL, cz.1, definiowanie danych (*data definition language*)

© Andrzej M. Borzyszkowski
Relacyjne Bazy Danych

Standard SQL

- Standard nieformalnie nazywany SQL/92
  - pełna nazwa: Międzynarodowy Standardowy Język Baz Danych SQL (1992)
  - skrót od *Structured Query Language"*
- Istniejące implementacje nie implementują w pełni powyższego standardu
  - ale rozszerzają niektóre aspekty standardu, czyli *nadzbiór podzbioru*
- Język deklaratywny – użytkownik deklaruje swoje potrzeby, optymalizator przekształca zapytanie na ciąg instrukcji
- Zawiera w sobie język definiowania danych i język manipulowania danymi
  - i dodatkowo język zarządzania użytkownikami, określania zabezpieczeń, sterowania transakcjami, ...

© Andrzej M. Borzyszkowski
Relacyjne Bazy Danych

SQL, kilka uwag ogólnych

- SQL realizuje raczej rachunek krotek niż algebrę relacji
- Relacja nazywana jest tabelą (*table*), może zawierać powtórzenia i oczywiście ma ustaloną kolejność
- Projekt bazy danych składa się głównie z zestawu tabel, są one zgrupowane w *schemacie*
- Standard wymaga by wielkość liter w nazwach nie grała roli
  - zasadniczo wszystkie słowa są konwertowane na duże litery
  - PostgreSQL zamienia wszystko na małe litery
  - gra to rolę jedynie gdy występują napisy w cudzysłowach
- Standard nie określa sposobu kończenia zapytania,
  - w PostgreSQL jest to średnik
- Każdy element musi mieć nazwę, nawet gdy nie mamy zamiaru odwoływać się do niego

© Andrzej M. Borzyszkowski
Relacyjne Bazy Danych

Typy wbudowane

- Jest wiele typów wbudowanych, najważniejsze z nich:
  - CHAR(_), VARCHAR(_)
  - INTEGER, SMALLINT
  - DATE, TIME, TIMESTAMP, obsługa czasu i daty
  - BOOLEAN, wartości np. 't', TRUE, '1','y', 'yes', (SQL/99)
  - NUMERIC(_,_), np. NUMERIC (7,2), 7 cyfr, w tym 2 po przecinku
  - FLOAT(_), np. FLOAT(15), 15 cyfr znaczących
  - BIT, VARBIT, wartości np. B'10011101'
  - MONEY, to samo co NUMERIC (9,2)

Definiowanie tabeli

- CREATE TABLE *nazwa_tabeli*
  lista-(*definicja_kolumny* [*wartość_domyślna*]
  | [*definicja_klucza_kandydującego*]
  | [*definicja_klucza_obcego*]
  | [*definicja_warunku_poprawności*]) ;
- *definicja_kolumny* ::= *nazwa_kolumny nazwa_dziedziny*
- *nazwa_dziedziny* ::= *typ_wbudowany* | *nazwa_zdefiniowana*
  ::= "to jest"
  | "albo"
  [ ] "alternatywnie"
- Wartość domyślna może zmienić wartość podaną w definicji dziedziny, brak definicji wartości domyślnej oznacza NULL
- Można żądać, by atrybut był zawsze określony: NOT NULL

Definiowanie tabel, klucze

- *definicja_klucza_kandydującego* ::=
  UNIQUE (*lista_kolumn*) |
  PRIMARY KEY (*lista_kolumn*)
  - lista kolumn w obu przypadkach jest niepusta
  - najwyżej jeden klucz może być określony jako główny (PRIMARY KEY)
  - jeśli występuje klucz główny, to wszystkie atrybuty tego klucza zyskują warunek poprawności NOT NULL
  - klucz alternatywny dopuszcza wartości NULL,
    - PostgreSQL – nie naruszają one warunku
    - SQL92 (np.. MS SQL Server) – naruszają warunek
- Warunki poprawności można opcjonalnie nazwać:
  CONSTRAINT *nazwa definicja_klucza_kand.*

Definiowanie tabel, klucze obce

- *definicja_klucza_obcego* ::=
  FOREIGN KEY (*lista_kolumn*)
  REFERENCES *tabela_bazowa* [(*lista_kolumn*)]
  [ON DELETE *opcja*]
  [ON UPDATE *opcja*]
  - nie jest wymagane podanie listy kolumn, jeśli klucz obcy odwołuje się do klucza o tej samej nazwie

  opcja ::= NO ACTION | CASCADE | SET DEFAULT | SET NULL
- Warunki poprawności można opcjonalnie nazwać:
  CONSTRAINT *nazwa definicja_klucza_obcego*

Definiowanie tabel, warunki poprawności

- *definicja_warunku_poprawności* ::=
  CHECK (*wyrażenie_warunkowe*)
  - *wyrażenie_warunkowe* może być dowolnie skomplikowane, nie musi ograniczać się do danej tabeli, musi być określone dla każdego wiersza tabeli
  - więzy poprawności są spełnione, jeśli powyższe *wyrażenie_warunkowe* ma wartość "true" dla każdego wiersza tabeli
  - system zarządzania bazą danych nie zezwoli na wprowadzenie danych czy aktualizację danych takie, że więzy poprawności nie są spełnione
  - kolejność sprawdzania warunków jest nieokreślona
- Warunki poprawności można opcjonalnie nazwać:
  CONSTRAINT *nazwa_definicja_warunku_poprawności*

Zmiana definicji tabeli podstawowej

- ALTER TABLE *nazwa_tabeli operacja*;
- Przykład

```
ALTER TABLE klient
ADD COLUMN rabat INT
DEFAULT 0;
```

- *operacja* może oznaczać
  - dodanie/usunięcie/zmiana nazwy kolumny,
  - zmiana dotychczasowej wartości domyślnej w kolumnie,
  - dodanie/usunięcie warunku poprawności

```
ALTER TABLE klient
ALTER COLUMN telefon DROP NOT NULL
```


Usuwanie tabeli podstawowej

- DROP TABLE *nazwa_tabeli* [RESTRICT | CASCADE];
  - jeżeli wybrano RESTRICT i tabela podstawowa występuje w jakiejkolwiek definicji perspektywy, to instrukcja DROP TABLE nie powiedzie się
  - jeżeli wybrano CASCADE, to instrukcja DROP TABLE powiedzie się i usunie daną tabelę wraz ze wszystkimi perspektywami bazującymi na tej tabeli oraz więzami poprawności