

# Bazy Danych i Usługi Sieciowe

## SQL dokończenie

Paweł Daniluk

Wydział Fizyki

Jesień 2012



# Data Manipulation Language

- Zapytania – klauzula `SELECT`
- Wstawianie – klauzula `INSERT`
- Aktualizacje – klauzula `UPDATE`
- Usuwanie – klauzula `DELETE`

# Klauzula INSERT

```
INSERT INTO  $R(A_1, A_2, \dots, A_n)$  VALUES ( $v_1, v_2, \dots, v_n$ );
```

- $R$  – relacja, do której wstawiane są krotki
- $A_1, A_2, \dots, A_n$  – lista atrybutów
- $v_1, v_2, \dots, v_n$  – wartości atrybutów  $A_i$  w nowej krotce

Dla pozostałych atrybutów relacji  $R$  nowa krotka będzie miała wartości domyślne (czyli zazwyczaj NULL).

# Klauzula INSERT

```
INSERT INTO  $R(A_1, A_2, \dots, A_n)$  VALUES ( $v_1, v_2, \dots, v_n$ );
```

- $R$  – relacja, do której wstawiane są krotki
- $A_1, A_2, \dots, A_n$  – lista atrybutów
- $v_1, v_2, \dots, v_n$  – wartości atrybutów  $A_i$  w nowej krotce

Dla pozostałych atrybutów relacji  $R$  nowa krotka będzie miała wartości domyślne (czyli zazwyczaj NULL).

Jeżeli lista atrybutów zostanie pominięta, należy podać wartości wszystkich atrybutów  $R$  w kolejności, w jakiej występują one w schemacie relacji.

## Klauzula INSERT – przykłady

```
INSERT INTO GraW(tytułFilmu, rokFilmu, nazwiskoGwiazdy)
VALUES ('Sokół Maltański', 1943, 'Sydney Greenstreet');
```

W tym wypadku można pominąć listę atrybutów

```
INSERT INTO GraW
VALUES ('Sokół Maltański', 1943, 'Sydney Greenstreet');
```

Można wstawić kilka krotek na raz

```
INSERT INTO GraW
VALUES ('Sokół Maltański', 1943, 'Sydney Greenstreet'),
      ('Batman', 1989, 'Michael Keaton'),
      ('Magnolia', 1999, 'Tom Cruise');
```

## Klauzula INSERT – przykłady c.d.

Nie zawsze chcemy wprowadzać dane “ręcznie”

Do relacji:

*Studia*(nazwa, adres, prezC#)

wstawić studia filmowe, które występują w relacji:

*Filmy*(tytuł, rok, długość, czyKolor, nazwaStudia, producent#)

ale nie występują w relacji *Studia*.

```
INSERT INTO Studio(nazwa)
  SELECT DISTINCT nazwaStudia
  FROM Filmy
  WHERE nazwaStudia NOT IN
    (SELECT nazwa FROM Studio);
```

# Klauzula DELETE

```
DELETE FROM R WHERE warunek;
```

## Przykład

```
DELETE FROM GraW  
  WHERE tytułFilmu = 'Sokół Maltański' AND  
        rokFilmu = 1943 AND  
        nazwiskoGwiazdy = 'Sydney Greenstreet';
```

# Klauzula DELETE

```
DELETE FROM R WHERE warunek;
```

## Przykład

```
DELETE FROM GraW
  WHERE tytułFilmu = 'Sokół Maltański' AND
        rokFilmu = 1943 AND
        nazwiskoGwiazdy = 'Sydney Greenstreet';
```

## Przykład

```
DELETE FROM Producenci
  WHERE wartość < 10000000;
```



# Klauzula UPDATE

UPDATE *R* SET przypisania WHERE warunek;

## Przykład

```
UPDATE Filmy  
    SET tytuł = 'Gwiezdne wojny'  
    WHERE tytuł = 'Gwiezdne wjony';
```

# Klauzula UPDATE

```
UPDATE R SET przypisania WHERE warunek;
```

## Przykład

```
UPDATE Filmy  
  SET tytuł = 'Gwiezdne wojny'  
  WHERE tytuł = 'Gwiezdne wjony';
```

## Przykład

```
UPDATE Producenci  
  SET nazwisko = 'Prez. ' || nazwisko  
  WHERE cert# IN (SELECT prezC# FROM Studia);
```

# Data Definition Language

- Tworzenie tabel – klauzula `CREATE TABLE`
- Usuwanie tabel – klauzula `DROP TABLE`
- Modyfikacja schematu – klauzula `ALTER TABLE`

# Typy danych (MySQL)

typ	wartości
CHAR( $n$ )	ciąg znaków o długości $n$
VARCHAR( $n$ )	ciąg znaków o maks. długości $n$
INT	liczba całkowita $-2,147,483,648 \div 2,147,483,647$
FLOAT	liczba zmiennoprzecinkowa
DOUBLE	liczba zmiennoprzecinkowa podwójnej precyzji
DECIMAL( $n, p$ )	liczba stałoprzecinkowa o $n$ cyfrach znaczących (dziesiętnych) i $p$ cyfrach w części ułamkowej
DATE	data
TIME	czas
TIMESTAMP	data i godzina
BLOB	duży obiekt binarny

# Klauzula CREATE TABLE

```
CREATE TABLE  $R(p_1\ t_1, p_2\ t_2, \dots, p_n\ t_n);$ 
```

- $R$  – nazwa tworzonej tabeli
- $p_i$  – nazwa  $i$ -tego pola
- $t_i$  – typ  $i$ -tego pola

## Przykład

```
CREATE TABLE GwiazdyFilmowe(  
    nazwisko CHAR(30),  
    adres VARCHAR(255),  
    płeć CHAR(1),  
    dataUrodzenia DATE  
);
```

# Klauzula DROP TABLE

```
DROP TABLE R;
```

- *R* – nazwa tabeli do usunięcia

## Przykład

```
DROP TABLE GwiazdyFilmowe;
```

# Klauzula ALTER TABLE – dodawanie i usuwanie atrybutów

```
ALTER TABLE R ADD p t;  
ALTER TABLE R DROP p;
```

- *R* – nazwa tabeli
- *p* – nazwa atrybutu
- *t* – typ

## Przykład

```
ALTER TABLE GwiazdyFilmowe ADD telefon CHAR(16);
```

```
ALTER TABLE GwiazdyFilmowe DROP dataUrodzenia;
```

# Wartości domyślne

Jeżeli wartość atrybutu przy wstawianiu krotki nie jest określona, przyjmuje ona wartość domyślną.

Domyślną wartością domyślną jest NULL.

## Przykłady

```
płeć CHAR(1) DEFAULT '?'
```

```
nazwisko CHAR(30) DEFAULT 'Doe'
```



# Wartości domyślne

Jeżeli wartość atrybutu przy wstawianiu krotki nie jest określona, przyjmuje ona wartość domyślną.

Domyślną wartością domyślną jest NULL.

## Przykłady

```
płeć CHAR(1) DEFAULT '?'  
nazwisko CHAR(30) DEFAULT 'Doe'
```

## Określanie wartości domyślnej przy dodawaniu atrybutu

```
ALTER TABLE GwiazdyFilmowe ADD telefon CHAR(16) DEFAULT 'głuchy';
```

# Wartości domyślne

Jeżeli wartość atrybutu przy wstawianiu krotki nie jest określona, przyjmuje ona wartość domyślną.

Domyślną wartością domyślną jest NULL.

## Przykłady

```
płeć CHAR(1) DEFAULT '?'  
nazwisko CHAR(30) DEFAULT 'Doe'
```

## Określanie wartości domyślnej przy dodawaniu atrybutu

```
ALTER TABLE GwiazdyFilmowe ADD telefon CHAR(16) DEFAULT 'głuchy';
```

## Zmiana wartości domyślnej

```
ALTER TABLE GwiazdyFilmowe ALTER telefon SET DEFAULT 'nieznany';
```

# Indeksy

Wyszukanie elementu w nieuporządkowanym zbiorze wymaga przejrzenia wszystkich jego elementów.

Wyszukiwanie w zbiorze posortowanym liczący  $n$  wymaga tylko  $\log n$  porównań.

Indeks określony na pewnym atrybucie  $A$  relacji  $R$  jest mechanizmem, który pozwala na szybkie wyszukiwanie krotek, dla których wartość  $A$  spełnia zadany warunek.

## Przykładowe techniki realizowania indeksów

- Mapy bitowe
- Tablice haszujące
- Drzewa wyszukiwania binarnego (np. czerwono-czarne)
- B-drzewa

## Uwaga

Indeks przyspiesza wyszukiwanie według atrybutu, którego dotyczy, ale wymaga aktualizacji przy każdej modyfikacji tabeli. Może się zdarzyć, że koszt utrzymania indeksu przewyższa zysk z jego istnienia.

# Klauzula CREATE INDEX

```
CREATE INDEX  $I$  ON  $R(A_1, A_2, \dots, A_n)$ ;
```

- $I$  – nazwa tworzonego indeksu
- $R$  – nazwa tabeli
- $A_1, \dots, A_n$  – atrybuty

## Przykłady

```
CREATE INDEX IndeksRoku ON Filmy(rok);  
CREATE INDEX IndeksKlucza ON Filmy(tytuł, rok);
```

## Usuwanie indeksu

```
DROP INDEX IndeksRoku;
```

# Jak wybrać atrybuty do indeksowania?

*GraW(tytułFilmu, rokFilmu, nazwiskoGwiazdy)*

*Q<sub>1</sub>*

```
SELECT tytułFilmu, rokFilmu FROM GraW  
WHERE nazwiskoGwiazdy=s;
```

*Q<sub>2</sub>*

```
SELECT nazwiskoGwiazdy FROM GraW  
WHERE tytułFilmu=t AND rokFilmu=y;
```

*/*

```
INSERT INTO GraW VALUES (t, y, s);
```

# Jak wybrać atrybuty do indeksowania? c.d.

## Założenia

- Relacja GraW jest przechowywana w 10 blokach na dysku.
- Gwiazdy występują średnio w 3 filmach. W filmie występują średnio 3 gwiazdy.
- Skorzystanie z indeksu wymaga przeczytania 1 bloku na dysku.

## Liczba dostępów do dysku

Operacja	Brak indeksów	Indeks gwiazdy	Indeks filmu	Oba indeksy
$Q_1$	10	4	10	4
$Q_2$	10	10	4	4
$I$	2	4	4	6

# Trzy rodzaje relacji

## Tabele

Trwałe, zapisane w bazie danych (CREATE TABLE), modyfikowalne (INSERT, UPDATE, DELETE).

## Wyniki podzapytań

Nietrwałe, niemodyfikowalne, istnieją tylko podczas obliczania wyniku zapytania.



# Trzy rodzaje relacji

## Tabele

Trwałe, zapisane w bazie danych (CREATE TABLE), modyfikowalne (INSERT, UPDATE, DELETE).

## Wyniki podzapytań

Nietrwałe, niemodyfikowalne, istnieją tylko podczas obliczania wyniku zapytania.

## Widoki

Nazwane wyniki zapytań. Trwałe, niemodyfikowalne (zazwyczaj), niezapisane w bazie danych.

# Kluczula CREATE VIEW

```
CREATE VIEW R AS Q;
```

- *R* – nazwa tworzonego widoku
- *Q* – zapytanie definiujące widok

## Przykład

```
CREATE VIEW FilmyParamount AS  
  SELECT tytuł, rok  
  FROM Filmy  
  WHERE nazwaStudia='Paramount';
```

```
CREATE VIEW FilmProd AS  
  SELECT tytuł, nazwisko  
  FROM Filmy, Producenci  
  WHERE producent#=cert#;
```

## Klauzula CREATE VIEW c.d.

### Usuwanie widoku

```
DROP VIEW FilmyParamount;
```

# Widoki mogą być modyfikowalne

Modyfikacja widoku jest dopuszczalna, jeżeli da się ją jednoznacznie przetłumaczyć na modyfikacje relacji, na bazie których widok jest zdefiniowany.

## Nie da się modyfikować widoku między innymi jeżeli

- Nie można dodawać ani usuwać krotek ze złączenia
- Nie można dodawać krotek, jeżeli widok nie zawiera niezbędnych atrybutów.
- Zawiera grupowanie i/lub funkcje agregujące (COUNT, SUM, AVG, ...)
- Zawiera sumę teoriomnogościową (UNION)
- Jest oparty na niemodyfikowalnym widoku

# Rodzaje więzów

## Klucze

- klucz główny (PRIMARY KEY)
- inne klucze (UNIQUE)

# Rodzaje więzów

## Klucze

- klucz główny (PRIMARY KEY)
- inne klucze (UNIQUE)

## Więzy integralności referencyjnej

- klucze obce (FOREIGN KEY)
- strategie utrzymywania więzów (CASCADE, SET NULL)

# Rodzaje więzów

## Klucze

- klucz główny (PRIMARY KEY)
- inne klucze (UNIQUE)

## Więzy integralności referencyjnej

- klucze obce (FOREIGN KEY)
- strategie utrzymywania więzów (CASCADE, SET NULL)

## Więzy określone na atrybutach i krotkach

- więzy NOT NULL
- więzy CHECK dla atrybutów lub krotek

# Rodzaje więzów

## Klucze

- klucz główny (PRIMARY KEY)
- inne klucze (UNIQUE)

## Więzy integralności referencyjnej

- klucze obce (FOREIGN KEY)
- strategie utrzymywania więzów (CASCADE, SET NULL)

## Więzy określone na atrybutach i krotkach

- więzy NOT NULL
- więzy CHECK dla atrybutów lub krotek

## Więzy określone dla bazy danych

- asercje
- wyzwalacze



# Klucze podstawowe

## Przykład

```
CREATE TABLE GwiazdyFilmowe(  
    nazwisko CHAR(30) PRIMARY KEY,  
    adres VARCHAR(255),  
    płeć CHAR(1),  
    dataUrodzenia DATE  
);
```

```
CREATE TABLE GwiazdyFilmowe(  
    nazwisko CHAR(30),  
    adres VARCHAR(255),  
    płeć CHAR(1),  
    dataUrodzenia DATE,  
    PRIMARY KEY (nazwisko)  
);
```

PRIMARY KEY (tytuł, rok)

# Klucze

## Przykład

`nazwisko CHAR(30) UNIQUE`

`UNIQUE (nazwisko)`

## Uwaga

W przypadku kluczy zdefiniowanych klauzulą `UNIQUE` dopuszczalne są wartości `NULL`.

# Klucze obce

## Przykład

```
CREATE TABLE Studia (  
    nazwa CHAR(30) PRIMARY KEY,  
    adres VARCHAR(255),  
    prezC# INT REFERENCES Producenti(cert#)  
);
```

```
CREATE TABLE Studia (  
    nazwa CHAR(30) PRIMARY KEY,  
    adres VARCHAR(255),  
    prezC# INT,  
    FOREIGN KEY (prezC#) REFERENCES Producenti(cert#)  
);
```

## Uwaga

Każda wartość atrybutu *prezC#* musi występować w jakiejś krotce w relacji *Producenti* (nie dotyczy to wartości NULL).

# Strategie utrzymywania więzów integralności referencyjnej

Domyślnie zmiany naruszające więzy są zakazane.

## Zmiany, które mogą naruszyć więzy

- Aktualizacja – klauzula `ON UPDATE`
- Usuwanie – klauzula `ON DELETE`

## Strategie usuwania naruszenia

- Propagacja kaskadowa – `CASCADE`
- Usuwanie błędnej wartości klucza – `SET NULL`

# Strategie utrzymywania więzów integralności referencyjnej c.d.

## Przykład

```
CREATE TABLE Studia (  
    nazwa CHAR(30) PRIMARY KEY,  
    adres VARCHAR(255),  
    prezC# INT REFERENCES Producenci(cert#)  
        ON DELETE SET NULL  
        ON UPDATE CASCADE  
);
```

# Więzy określone na atrybutach lub krotkach

## Więzy NOT NULL

```
prezC# INT REFERENCES Producenti(cert#) NOT NULL
```

## Więzy dla atrybutów

```
płeć CHAR(1) CHECK (płeć IN ('K', 'M'))
```

```
prezC# INT CHECK  
(prezC# IN (SELECT cert# FROM Producenti))
```

## Uwaga

Nie jest możliwe wprowadzenie nieistniejącego identyfikatora prezesa studia, ale usunięcie producenta, który jest prezesem, nie narusza więzu.

## Więzy określone na atrybutach lub krotkach c.d.

### Więzy dla krotek

```
CREATE TABLE GwiazdyFilmowe(  
    nazwisko CHAR(30) PRIMARY KEY,  
    adres VARCHAR(255),  
    płeć CHAR(1),  
    dataUrodzenia DATE,  
    CHECK (płeć = 'F' OR nazwisko NOT LIKE 'Pani%')  
);
```

# Asercje i wyzwalacze

## Asercje

- Więzy podobne do CHECK, ale sprawdzane przy każdej modyfikacji bazy danych
- Nie wspierane przez większość SZBD.

## Wyzwalacze (*Triggers*)

- Procedury wykonywane przed lub po zajściu zdarzenia (albo zamiast).
- Mogą realizować dowolne zmiany w bazie danych.
- Efektem wyzwalacza może być wycofanie zmiany, która go wywołała.



## Zastosowania wyzwaczy

- ewidencjonowanie zmian (audyt)
- automatyczne rozszerzanie zmian (np. o stempel czasowy)
- wymuszanie skomplikowanych więzów (np. zakaz zapętleń w hierarchii organizacyjnej)
- wykonywanie procedur (np. propagowanie zmiany)
- replikacja danych
- aktualizacja danych obliczanych (np. aktualizacja stanu konta po zaewidencjonowaniu operacji)

# Transakcje

## Szeregowanie

W przypadku równoczesnego dostępu do bazy danych może łatwo nastąpić naruszenie spójności danych.

## Przykład – rezerwacja biletów

- 1 Klient 1 sprawdza, czy miejsce jest wolne.
- 2 Klient 2 sprawdza, czy miejsce jest wolne.
- 3 Klient 1 wprowadza rezerwację.
- 4 Klient 2 wprowadza rezerwację.

## Transakcje c.d.

### Niepodzielność

Niektóre operacje muszą być wykonane razem. W przypadku awarii lub błędu nie powinna się wykonać żadna z nich.

### Przykład – przelew bankowy

- 1 Odejmij kwotę od salda konta A.
- 2 Dodaj kwotę do salda konta B.

# Transakcje – ACID

## ACID

- atomicity – atomowość
- consistency – spójność
- isolation – izolacja
- durability – trwałość

## Atomowość

Każda transakcja albo wykona się w całości, albo w ogóle.

## Spójność

Po wykonaniu transakcji system będzie spójny, czyli nie zostaną naruszone żadne zasady integralności.

# Transakcje – ACID c.d.

## Izolacja

Jeżeli dwie transakcje wykonują się współbieżnie, to zazwyczaj (zależnie od poziomu izolacji) nie widzą zmian przez siebie wprowadzanych. Poziom izolacji jest zazwyczaj konfigurowalny:

- read uncommitted – najniższy poziom izolacji, jedna transakcja może odczytywać wiersze, na których działają inne transakcje,
- read committed – transakcja może odczytywać tylko wiersze zapisane,
- repeatable read – transakcja nie może czytać, ani zapisywać, na wierszach odczytywanych, bądź zapisywanych w innej transakcji,
- serializable – pełna izolacja

## Trwałość

System potrafi uruchomić się i udostępnić spójne, nienaruszone i aktualne dane zapisane w ramach zatwierdzonych transakcji, na przykład po nagłej awarii zasilania.

# Wydajność systemów baz danych

## Sprzęt

- macierze dyskowe
- systemy wieloprocessorowe
- komputery mainframe

## Algorytmy

- indeksy
- optymalizatory zapytań
- pamięć podręczna

# Wydajność systemów baz danych c.d.

## Współbieżność

- każdy procesor obsługuje inne żądanie
- systemy rozproszone

## Projektowanie

- schemat bazy danych dostosowany do obsługi najczęstszych żądań
- analizy wydajności

[http://bioexploratorium.pl/wiki/  
Bazy\\_Danych\\_i\\_USlugi\\_Sieciowe\\_-\\_2012z](http://bioexploratorium.pl/wiki/Bazy_Danych_i_USlugi_Sieciowe_-_2012z)