SQL – Podstawy języka

Krzysztof Regulski
WIMiIP, KISiM,
regulski@agh.edu.pl
B5, pok. 408

Języki zapytań

- Interfejsy typu zapytanie przez przykład (ang. Query by Example - QBE), szblony (formularze, strony WWW)
- Structured Query Language (SQL), języki algebraiczne
- języki predykatowe (o zmiennych atrybutowych i krotkowych)
- DATALOG (język zbliżony do PROLOGu ale nieproceduralny i bez termów)
- XQuery (XML Query Language) – język zapytań do przeszukiwania dokumentów XML.
- SPARQL (SPARQL Protocol And RDF Query Language) - język zapytań i protokół dla plików RDF.

Czym jest SQL?

- *Structured Query Language* **SQL** - nieproceduralny język typu strukturalnego przeznaczony do uzyskiwania dostępu i operowania danymi (DML) oraz budowy bazy danych (DDL).
- Program składa się z poleceń, które mają określoną strukturę wewnętrzną (dyrektywy wewnętrzne).
- SQL w swoich konstrukcjach opiera się na algebrze relacji.
- Aplikacje sięgają do bazy danych za pomocą SQL w dwóch trybach:
  - Wykonane są w językach (np. C, Cobol, Pascal) rozszerzonych o możliwość łączenia z SQL (*embedded* SQL)
  - Korzystają ze specjalnego interfejsu (np. ODBC/JDBC), który pozwala wysyłać do bazy zapytania sformułowane w SQL.

SQL – historia

- 1974: Chamberlain, IBM, San Jose – Structured English Qery Language *SEQUEL*
- koniec lat 70-tych: ORACLE (Relational Software Inc.) – pierwsza implementacja komercyjna
- 1982: ANSI* – RDL (Relational Data Language)
- 1983: ISO** – definicja SQL
- 1986: ANSI – pierwszy standard SQL (SQL-86)
- 1987: ISO – pierwszy standard SQL (ISO 9075)
- 1989: ISO – następny standard SQL (ISO 9076 – SQL-89)
- 1992: ISO – wzbogacona wersja ISO 9075 (SQL-92)
- 1999: ANSI/ISO – kolejna wersja ISO 9075 (SQL-99 lub SQL 3)
- 2003: ANSI/ISO – najnowsza wersja - SQL2003

*American National Standards Committee
**International Standards Organisation

Elementy SQL:

- **DDL** *(Data Definition Language)* – tworzenie, usuwanie i modyfikacja schematów, kluczy, indeksów, widoków, warunków integralności i praw dostępu, a także fizyczną strukturę pamięci dyskowej (CREATE, DROP, ALTER)

- **DML** (*Data Manipulation Language*) – język zapytań oparty na algebrze relacji obejmujący ponadto polecenia dodające, usuwające i aktualizujące dane w bazie danych (SELECT, INSERT, DELETE, UPDATE)

- **Kontrola transakcji** – SQL obejmuje polecenia rozpoczęcia i zakończenia transakcji, a także blokowania danych dla współbieżnych operacji (START TRANSACTION, COMMIT, ROLLBACK etc.)

Osadzony SQL

- Standard SQL definiuje również zasady osadzania (*embedded*) SQL w językach programowania, takich jak Pascal, PL/1, Fortran, C i Cobol.
- Język, w którym osadzono SQL nazywany jest *host language* a struktury SQL dozwolone w tym języku tworzą osadzony SQL.
- Etykieta EXEC SQL/END EXEC jest używana do wskazywania struktur osadzonego SQL.
  - » EXEC SQL <struktura-osadzonego-SQL> END EXEC
- <struktura-osadzonego-SQL> wykorzystuje w ogólnym przypadku pełne możliwości SQL uzupełnione o pewne elementy wynikające z osadzenia.

Inne Zalety SQL

- Języki czwartej generacji - specjalne języki pomagające programistom w tworzeniu wzorców dla dialogu z użytkownikiem i w formatowaniu danych dla raportów, dostępne dla większości komercyjnych baz danych (PL/SQL).
- Sesja SQL - dostarcza abstrakcji klienta i serwera (także zdalnego):
  - klient łączy (*connect)* się z serwerem SQL nawiązując sesję;
  - wykonuje serię poleceń;
  - rozłącza się od sesji (*disconnect);*
  - może zapisać (*commit)* albo wycofać (*rollback)* pracę wykonaną w sesji.
- Środowisko SQL zawiera kilka komponentów między innymi identyfikator użytkownika i bazy danych, które pozwalają zidentyfikować którą z kilku baz danych używa dana sesja.

Typy danych w MySQL (1)

- `CHAR(n)` – skończonej długości łańcuch znakowy, z podaną przez użytkownika długością *n*
- `VARCHAR(n)` – zmiennej długości łańcuch znakowy, z podaną przez użytkownika maksymalną długością *n*
- `TEXT` – typ znakowy różniący się od `CHAR` i `VARCHAR` długością, nie może posiadać wartości `DEFAULT`
- `INT(n)`, `INTEGER(n)` – liczba całkowita o podanej długości
- `BOOL`, `BOOLEAN` – równoważne typowi `TINYINT(1)`, mogą posiadać wartość `1` lub `0` rozumiane odpowiednio jako `TRUE` lub `FALSE`
- `DATE` - daty zawierające rok (4-cyfrowy), miesiąc i dzień (w formacie YYYY-MM-DD)
- `TIME` - czas w godzinach, minutach i sekundach
- `DATETIME` – kombinacja `DATE` i `TIME` (np.: `9999-12-31 23:59:59`)

Typy danych w MySQL (2)

- FLOAT – mała liczba rzeczywista, zmiennoprzecinkowa
- DOUBLE(length, decimal) – duża liczba rzeczywista, zmiennoprzecinkowa
- DECIMAL(length, decimal) – liczba typu DOUBLE przechowywana w postaci łańcucha co pozwala na zastosowanie stałej liczby miejsc po przecinku
- SERIAL - jest aliasem dla

  'BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE'
- SERIAL DEFAULT VALUE jako atrybut pola typu integer (od TINYINT do BIGINT) jest aliasem dla NOT NULL AUTO_INCREMENT
- Wartości NULL dozwolone są we wszystkich typach. Deklarując atrybut ze specyfikatorem NOT NULL, zabrania się wpisywania wartości NULL dla tego atrybutu.

Obsługa wartości NULL

- wartość `NULL` nie może być umieszczona w kolumnie `NOT NULL`,
- porównywanie dwóch kolumn zawierających `NULL` jest nieskuteczne (wartości `NULL` można identyfikować w klauzuli `WHERE` przy użyciu wyrażeń `IS NULL; IS NOT NULL`)
- kolumna zawierająca `NULL` jest ignorowana podczas obliczania wartości agregujących natomiast jest uwzględniana w klauzuli `GROUP BY`
- jeżeli w warunku złączenia pojawi się kolumna z wartościami `NULL` to złączenie traktowane jest jako zewnętrzne

Konstrukcja tabeli

– Relacja (tabela) w SQL jest definiowana za pomocą polecenia postaci:

```
CREATE TABLE nazwa_tabeli
(A1 D1,  A2 D2,...,  An Dn,
        [warunki-integralności]);
```

» Każde A_i jest nazwą atrybutu w schemacie relacji.

» Każde D_i określa dziedzinę atrybutu A_i przez podanie typu danych opisujących atrybut być może ze specyfikatorem NOT NULL

– [warunki-integralności] mogą przyjmować postaci:

» PRIMARY KEY $(A_1, \ldots, A_n)$ – definicja klucza zgodnie z zasadami.

» CHECK (P) gdzie P jest predykatem akceptowalnym w klauzuli WHERE zapytania SQL.

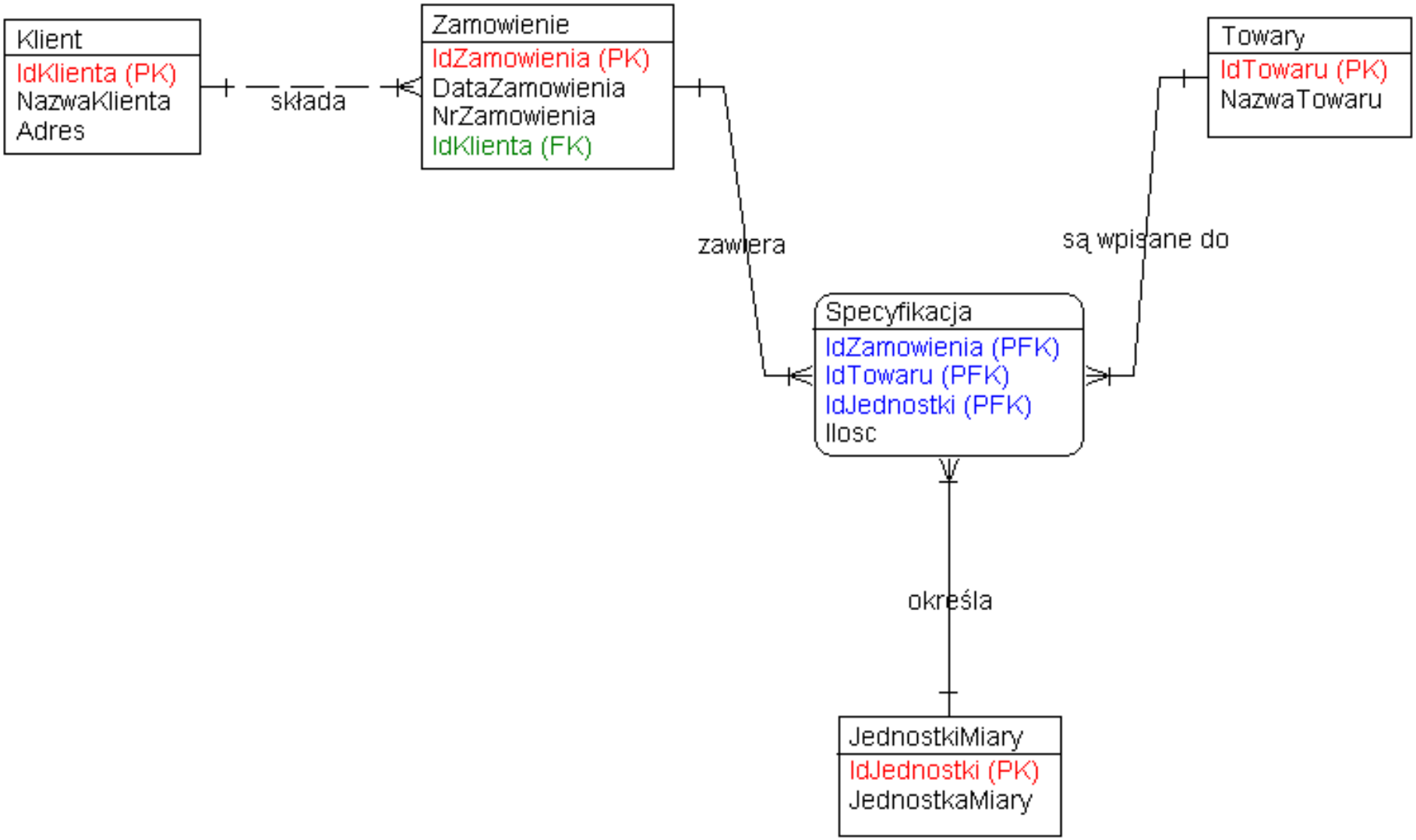
Przykład:

```
CREATE TABLE pracownicy (
        pesel CHAR(11) NOT NULL,
        imie VARCHAR(15) NOT NULL,
        nazwisko VARCHAR(40) NOT NULL,
        tytul VARCHAR(10),
    PRIMARY KEY (pesel),
    CHECK tytul IN (SELECT tytul-nazwa FROM tytuly));
```

Ograniczenie można zadać poprzez zdefiniowanie warunku logicznego, w tym także takiego, które sięga do innych tabel lub poprzez standardowego ograniczenia: NOT NULL lub UNIQUE

Integralność danych w SQL

- Wprowadzenie kluczy podstawowych i obcych zapewnia automatyczną kontrolę poprawności struktury danych i operacji przetwarzania danych
- Klucz podstawowy zapewnia unikalność i możliwość identyfikacji każdego zapisu
- Klucze obce zapewniają integrację referencyjną głoszącą, że każda niepusta wartość klucza obcego musi odpowiadać jednej z istniejących wartości klucza podstawowego


Klient
IdKlienta (PK)
NazwaKlienta
Adres
składa
Zamowienie
IdZamowienia (PK)
DataZamowienia
NrZamowienia
IdKlienta (FK)
zawiera
Towary
IdTowaru (PK)
NazwaTowaru
są wpisane do
Specyfikacja
IdZamowienia (PFK)
IdTowaru (PFK)
IdJednostki (PFK)
Ilosc
określa
JednostkiMiary
IdJednostki (PK)
JednostkaMiary

Klucz podstawowy

- W tablicach rodzicach (`parent`) jest niezbędny jako łącznik z tablicami dziećmi (`child`)
- Najlepiej rolę tą spełnia klucz, który jest kolumną tożsamości (`idenity`)

Klucz podstawowy w definicji i zmianie tabeli

```
CREATE TABLE Towary (
        IdTowaru Int NOT NULL AUTO_INCREMENT,
        NazwaTowaru Char(50),
 PRIMARY KEY (IdTowaru);
```

```
ALTER TABLE Towary DROP PRIMARY KEY;

ALTER TABLE Towary ADD PRIMARY KEY (IdTowaru);
```

Klucze obce

- W związkach nieidentyfikujących pozwalają na tworzenie złączeń (nie jest do tego konieczne wymuszanie integralności)
- W związkach identyfikujących pozwalają na kontrolę unikalności krotek w relacjach dzieciach
- FOREIGN KEY znany jako klucz obcy to pewnego rodzaju odnośnik łączący tabelę w którym występuje klucz obcy z inną tabelą. Klucz obcy zapobiega wszelkim operacjom, które mogłyby zerwać taką więź między tabelami.

```
CREATE TABLE Klient (
      IdKlienta Int NOT NULL AUTO_INCREMENT,
      NazwaKlienta Char(50),
      Adres Char(50),
 PRIMARY KEY (IdKlienta));


CREATE TABLE Zamowienie (
      IdZamowienia Int NOT NULL AUTO_INCREMENT,
      DataZamowienia Date,
      NrZamowienia Char(20),
      IdKlienta Int,
 PRIMARY KEY (IdZamowienia));
```

Złączenie naturalne

```
mysql> select nazwaklienta, datazamowienia from klient join zamowienie using
    -> (idklienta) order by nazwaklienta;
+----------------------------------+-----------------+
| nazwaklienta                     | datazamowienia  |
+----------------------------------+-----------------+
| Fabryka Rower~w                  | 2005-03-30      |
| Fabryka Rower~w                  | 2005-03-30      |
| Fabryka Rower~w                  | 2005-03-30      |
| Fabryka Rower~w                  | 2005-03-30      |
| Fabryka Rower~w                  | 2005-03-30      |
| Fabryka w~zk~w dzieciźcych       | 2005-04-02      |
| Hurtownia Materia|~w Budowlanych | 2005-03-30      |
| Klient 2                         | 2005-04-06      |
| Klient 3                         | 2005-04-07      |
| Klient 5                         | 2005-04-02      |
| Klient 5                         | 2005-04-02      |
| Klient 5                         | 2005-04-02      |
| Sklep Og~lny                     | 2005-03-27      |
+----------------------------------+-----------------+
13 rows in set (0.39 sec)
```

Złożony klucz obcy

```
CREATE TABLE JednostkiMiary (
        IdJednostki Int NOT NULL AUTO_INCREMENT,
        JednostkaMiary Char(20),
 PRIMARY KEY (IdJednostki));

CREATE TABLE Specyfikacja (
        IdZamowienia Int NOT NULL,
        IdTowaru Int NOT NULL,
        IdJednostki Int NOT NULL,
        Ilosc Float,
 PRIMARY KEY (IdZamowienia,IdTowaru,IdJednostki));
```

```
mysql> select * from specyfikacja;
+--------------+----------+--------------+-------+
| IdZamowienia | IdTowaru | IdJednostki | Ilosc |
+--------------+----------+--------------+-------+
|            1 |        1 |            1 |   100 |
|            1 |        2 |            1 |   150 |
|            2 |        1 |            2 |   200 |
|            2 |        2 |            1 |   120 |
|           59 |        1 |            2 |   200 |
|           59 |        2 |            1 |   100 |
|           60 |        2 |            1 |    50 |
|           60 |        1 |            2 |   100 |
|           66 |        3 |            1 |   500 |
|           61 |        2 |            1 |   543 |
|           62 |        2 |            1 |   100 |
|           62 |        1 |            1 |   200 |
|           63 |        2 |            1 |   105 |
|           70 |        1 |            1 |   250 |
|           70 |        3 |            1 |   200 |
|           69 |        2 |            2 |   120 |
|           69 |        6 |            3 |   300 |
|           69 |        1 |            1 |   250 |
|           69 |        3 |            1 |   200 |
|           69 |        2 |            1 |   150 |
|           70 |        6 |            3 |   320 |
|           71 |        3 |            1 |   150 |
|           70 |        2 |            1 |   150 |
|           71 |        2 |            1 |   300 |
|           72 |        2 |            1 |   100 |
|           72 |        3 |            1 |   300 |
+--------------+----------+--------------+-------+
26 rows in set (0.41 sec)

mysql> insert into specyfikacja (idzamowienia, idtowaru, idjednostki, ilosc)
    -> values('1', '1', '1', '200');
ERROR 1062 (23000): Powt~rzone wyst?pienie '1-1-1' dla klucza 1
mysql>
```

próba wprowadzenia identycznego klucza

Wymuszanie integralności

- REFERENCES – podaje źródło klucza obcego, tj. tabelę i klucz podstawowy
- ON DELETE, ON UPDATE – określenie czynności, które należy podjąć jeśli wartość klucza podstawowego zostanie usunięta lub ulegnie zmianie:
  - » SET NULL zastąp wartość klucza obcego przez NULL,
  - » SET DEFAULT zastąp wartość klucza obcego przez wartość domyślną,
  - » CASCADE skasuj lub zmodyfikuj wszystkie wiersze zawierające zmienianą wartość klucza obcego
  - » NO ACTION (tylko modyfikacja) nie zmieniaj wartości klucza
  - » RESTRICT nie dopuść do zmiany

Zabronione usuwanie w MySQL

```
ALTER TABLE Zamowienie ADD FOREIGN KEY (IdKlienta)
REFERENCES Klient (IdKlienta) ON DELETE  RESTRICT
ON UPDATE RESTRICT;
```

```
DELETE FROM `klient` WHERE `IdKlienta` =1 LIMIT 1
```

#1217 - Cannot delete a parent row: a foreign key constraint fails

IdKlienta	NazwaKlienta	Adres
1	Klient 1	*NULL*
2	Sklep Ogólny	Zabierzów ul. Spokojna 2
3	Firma Kruk	Modlniczka 127
4	Fabryka Rowerów	Kraków ul. Bracka 1
5	Fabryka Mebli	Zabierzów, ul. Krakowska 12
6	Hurtownia Materia³ów Budowlanych	Kraków ul.Wielicka 20
7	Fabryka wózków dzieciêcych	Krzeszowice ul, Krakowska 5
10	Klient 5	*NULL*
11	Klient 2	*NULL*
12	Klient 3	*NULL*

IdZamowienia	DataZamowienia	NrZamowienia	IdKlienta
1	2005-03-27	2005/127	1
2	2005-03-27	2005/128	2
59	2005-03-30	2005/201	4
60	2005-03-30	2005/202	4
61	2005-03-30	2005/203	4
62	2005-03-30	2005/204	4
63	2005-03-30	2005/an	4
66	2005-03-30	2005/303	6
67	2005-04-02	2005/876	10
68	2005-04-02	2005/876	10
69	2005-04-02	2005/876	10
70	2005-04-02	2005/876	7
71	2005-04-06	05/12	11
72	2005-04-07	9876/05	12

Kaskadowa aktualizacja w MySQL

```
ALTER TABLE Zamowienie ADD FOREIGN KEY (IdKlienta)
REFERENCES Klient(IdKlienta)
ON DELETE CASCADE ON UPDATE CASCADE;
```

```
DELETE FROM `klient` WHERE `IdKlienta` =1
```

IdKlienta	NazwaKlienta	Adres
2	Sklep Ogólny	Zabierzów ul. Spokojna 2
3	Firma Kruk	Modlniczka 127
4	Fabryka Rowerów	Kraków ul. Bracka 1
5	Fabryka Mebli	Zabierzów, ul. Krakowska 12
6	Hurtownia Materia³ów Budowlanych	Kraków ul.Wielicka 20
7	Fabryka wózków dzieciêcych	Krzeszowice ul, Krakowska 5
10	Klient 5	*NULL*
11	Klient 2	*NULL*
12	Klient 3	*NULL*

IdZamowienia	DataZamowienia	NrZamowienia	IdKlienta
2	2005-03-27	2005/128	2
59	2005-03-30	2005/201	4
60	2005-03-30	2005/202	4
61	2005-03-30	2005/203	4
62	2005-03-30	2005/204	4
63	2005-03-30	2005/an	4
66	2005-03-30	2005/303	6
67	2005-04-02	2005/876	10
68	2005-04-02	2005/876	10
69	2005-04-02	2005/876	10
70	2005-04-02	2005/876	7
71	2005-04-06	05/12	11
72	2005-04-07	9876/05	12

Porządkowanie kluczy w MySQL

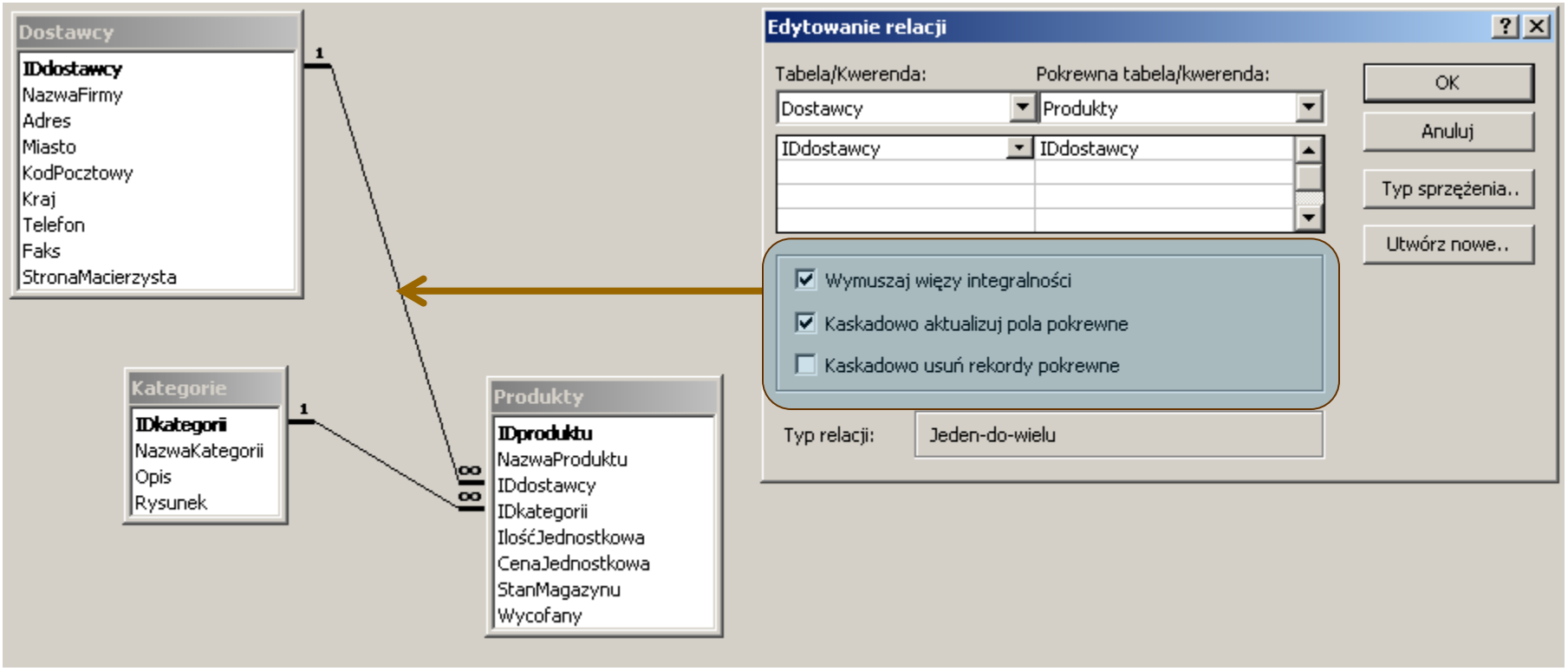
– Zmiana wartości klucza w tabeli rodzicielskiej powoduje odpowiednie zmiany w tabelach dzieciach

```
UPDATE `klient` SET `IdKlienta` = '8' WHERE
  `IdKlienta` =10;
```

IdKlienta	NazwaKlienta	Adres
2	Sklep Ogólny	Zabierzów ul. Spokojna 2
3	Firma Kruk	Modlniczka 127
4	Fabryka Rowerów	Kraków ul. Bracka 1
5	Fabryka Mebli	Zabierzów, ul. Krakowska 12
6	Hurtownia Materia³ów Budowlanych	Kraków ul.Wielicka 20
7	Fabryka wózków dzieciêcych	Krzeszowice ul, Krakowska 5
8	Klient 5	*NULL*
11	Klient 2	*NULL*
12	Klient 3	*NULL*

IdZamowienia	DataZamowienia	NrZamowienia	IdKlienta
2	2005-03-27	2005/128	2
59	2005-03-30	2005/201	4
60	2005-03-30	2005/202	4
61	2005-03-30	2005/203	4
62	2005-03-30	2005/204	4
63	2005-03-30	2005/an	4
66	2005-03-30	2005/303	6
67	2005-04-02	2005/876	8
68	2005-04-02	2005/876	8
69	2005-04-02	2005/876	8
70	2005-04-02	2005/876	7
71	2005-04-06	05/12	11
72	2005-04-07	9876/05	12

Kaskadowa aktualizacja w MS Access



Modyfikacja schematu bazy (1)

- Polecenie ALTER wykonuje następujące operacje:
  - » dodaje kolumny i warunki
  - » modyfikuje definicje kolumn jak typy i warunki
  - » usuwa warunki
- ALTER TABLE nazwa_tabeli ADD A_i D_i
  jest używane do dodawania atrybutów do istniejącej relacji (tabeli). A_i jest nazwą atrybutu dodawanego do relacji nazwa_tabeli, a D_i jest specyfikacją typu A_i. Wszystkie krotki relacji uzyskują dla nowego atrybutu wartość NULL.
- ALTER TABLE nazwa_tabeli DROP A_i
  może być użyta do usunięcia atrybutu A_i relacji (tablicy) nazwa_tabeli
- Przykłady:

```
ALTER TABLE pracownicy ADD (placa DECIMAL(7, 2))
ALTER TABLE pracownicy MODIFY (placa DECIMAL(9, 2))
```

Modyfikacja schematu bazy (2)

- `DROP TABLE tabela1, tabela2`

  Usuwa relacje (tabele) `tabela1, tabela2` z bazy danych

- `RENAME TABLE tabela1 TO tabela1`

  Polecenie służące do zmiany nazwy jednej lub więcej tabel

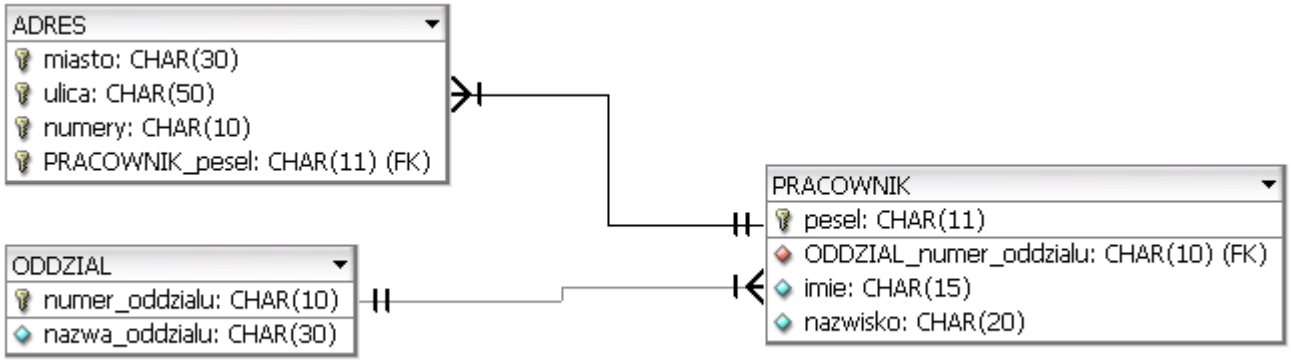
- Przykłady:

```
CREATE TABLE new_table (...);
RENAME TABLE old_table TO backup_table,
             new_table TO old_table;

RENAME TABLE current_database.table_name
             TO other_database.table_name;
```

Modyfikacja schematu relacji

– Utwórz tabelę wg schematu:




```
CREATE TABLE ODDZIAL (
numer_oddzialu CHAR(10) NOT NULL,
nazwa_oddzialu CHAR(30) NULL,
PRIMARY KEY(numer_oddzialu)
);

CREATE TABLE ADRES (
miasto CHAR(30) NOT NULL,
ulica CHAR(50) NOT NULL,
numery CHAR(10) NOT NULL,
PRACOWNIK_pesel CHAR(11) NOT NULL,
PRIMARY KEY(miasto, ulica, numery, PRACOWNIK_pesel),
);
```

```
CREATE TABLE PRACOWNIK (
pesel CHAR(11) NOT NULL,
ODDZIAL_numer_oddzialu CHAR(10) NOT
NULL,
imie CHAR(15) NULL,
nazwisko CHAR(20) NULL,
PRIMARY KEY(pesel),
);
```

– Jakie tabele znajdują się w bazie?

```
SHOW TABLES;
```

– Jakie kolumny znajdują się w tabeli PRACOWNIK?

```
SHOW COLUMNS FROM PRACOWNIK;
```

lub

```
DESCRIBE PRACOWNIK;
```

– Usuń tabelę PRACOWNIK

```
DROP TABLE PRACOWNIK;
```

– Usuń i dodaj ponownie klucz podstawowy:

```
ALTER TABLE PRACOWNIK DROP PRIMARY KEY;
ALTER TABLE PRACOWNIK ADD PRIMARY KEY (pesel);
```

– Dodaj kolumnę *placa* do tabeli pracownicy:

```
ALTER TABLE PRACOWNIK
ADD placa DOUBLE NULL DEFAULT 0
AFTER nazwisko;
```

Modyfikacja danych (1)

- Modyfikacja bazy danych przez ich usunięcie może być zrealizowana w SQL za pomocą polecenia postaci:

  DELETE FROM <relacja> WHERE <warunek>

- Można zatem usuwać jedynie całe krotki, dla których prawdziwy jest <warunek>, oraz jednokrotnie tylko z jednej <relacja> (tabeli).
- Opuszczenie klauzuli WHERE skutkuje usunięciem danych z całej tabeli.
- Wykonanie DELETE przebiega w dwóch fazach: najpierw realizowany jest wybór wszystkich krotek, a następnie ich usuwanie.

  Usuń pracowników z wynagrodzeniem równym zero.

```
DELETE FROM pracownicy
WHERE placa = 0;
```

- <warunek> może mieć formę dowolną akceptowaną przez klauzulę WHERE w zapytaniach, w tym może zawierać podzapytania.

Modyfikacja danych (2)

- Modyfikacja bazy danych przez ich dodanie może być zrealizowana w SQL za pomocą poleceń postaci:

  INSERT INTO <relacja> VALUES <krotka>

  INSERT INTO <relacja> VALUES <relacja-wstawiana>

- Można zatem dodawać jedynie całe krotki oraz jednokrotnie tylko do jednej relacji (tabeli). Zachowany być musi schemat relacji i domeny atrybutów.
- O ile to jest dopuszczone <krotka> może zawierać wartości *NULL*.

Dodaj nową krotkę do relacji *pracownicy* z wynagrodzeniem ustawionym na *NULL*

```
INSERT INTO pracownicy (pesel, nazwisko, imie, id_oddzialu, id_adresu)
VALUES ('99999900000', 'Regulski', 'Krzysztof','104BT',NULL)

INSERT INTO pracownicy VALUES ('99999900000', 'Regulski', 'Krzysztof','104BT',NULL)
```

- Formę z **values** stosować można również do widoków.

Wstawianie wierszy

```
INSERT INTO Klient
(NazwaKlienta, Telefon, KodPocztowy, Miejscowosc, Ulica,
NrDomuMieszkania, Email)
VALUES('Nowy klient', '48 12 1234567', '30-333', 'Bolechowice',
'Jurajska', '20','ala@tlen.pl')
```

```
+-----------+--------------------+---------------+
| IdKlienta | NazwaKlienta       | Telefon       |
+-----------+--------------------+---------------+
|         3 | STALHANDEL         | 48 32 7865748 |
|         2 | Firma Krok Sp zoo  | 48 12 6374532 |
|         5 | PHPU OSA           | 48 12 6372312 |
|         4 | Rower Polska SA    | 48 12 2853364 |
|         1 | FH Klin SA         | 48 12 1273210 |
|         6 | Nowy klient        | 48 12 1234567 |
+-----------+--------------------+---------------+
```

Modyfikacja danych (3)

- Modyfikacja bazy danych przez zmianę wartości atrybutów może być zrealizowana w SQL za pomocą polecenia postaci:

  UPDATE <relacja> SET <modyfikacja> WHERE <warunek>

- Można modyfikować jeden atrybut krotek, dla których prawdziwy jest <warunek> oraz jednokrotnie tylko z jednej <relacja> (tabeli).
- <modyfikacja> jest **wyrażeniem arytmetycznym** (akceptowalnym w klauzuli SELECT), przypisującym (=) nową wartość atrybutowi.
- Opuszczenie klauzuli WHERE skutkuje modyfikacją całej tabeli.

Podnieś wynagrodzenie Regulskiemu o 30%

```
UPDATE pracownicy
SET placa=placa*1.3
WHERE nazwisko LIKE 'Regulski';
```

– Popraw miejscowość w adresie Kosmateusza Bulera na ‘Ozimek’:

```
UPDATE PRACOWNIK
SET adres = ”Plac Maly 56, Ozimek”
WHERE imie = ”Kosmateusz”
AND nazwisko = ”Buler”;
```

– Podnieś wynagrodzenie Regulskiemu o 30%

```
UPDATE PRACOWNIK
SET placa=placa*1.3
WHERE nazwisko LIKE ‘Regulski’;
```

– Zmień w bazie danych id_oddzialu dla oddziału Betatrex na B1 (pamiętaj o numerach u pracowników!)

```
UPDATE ODDZIAL
SET id_oddzialu='B1'
WHERE id_oddzialu='Betatrex';

UPDATE PRACOWNIK
SET id_oddzialu='B1'
WHERE id_oddzialu='Betatrex';
```

– żeby nie popełnić błędu w przyszłości, można założyć więzy integralności w postaci klucza obcego:

```
ALTER TABLE PRACOWNIK ADD FOREIGN KEY (id_oddzialu)
REFERENCES ODDZIAL(id_oddzialu)
ON DELETE CASCADE ON UPDATE CASCADE;
```

- Usuń pracownika o peselu 56123099087

```
DELETE FROM PRACOWNIK
WHERE pesel='56123099087';
```

- Usuń wszystkie dane z tabeli PRACOWNIK
(uważaj, żeby nie usunąć danych z innych tabel!)

```
DELETE FROM PRACOWNIK;
```

- Usuń tabelę PRACOWNIK

```
DROP PRACOWNIK;
```

Przyznawanie praw dostępu

- Serwer baz danych może obsługiwać wielu użytkowników identyfikowanych przez nazwę i hasło
- Nie każdy z użytkowników musi mieć równe prawa
- W momencie utworzenia nowego elementu bazy danych aktualny użytkownik staje się jego właścicielem
- Właściciel może nadawać i odbierać prawa innym użytkownikom

Przyznawanie praw dostępu składnia SQL99

```
GRANT  {ALL [PRIVILEGES] }
| SELECT
| INSERT [ nazwa_kolumny [,...n]) ]
| DELETE
| UPDATE [ nazwa_kolumny [,...n]) ]
| REFERENCES [ nazwa_kolumny [,...n]) ]
| USAGE } [,...n]
ON { [TABLE] nazwa_tabeli
| DOMAIN nazwa_domeny
| COLLATION nazwa_zestawienia
CHARACTER SET nazwa_zestawu_znaków
| TRANSLATION nazwa_translacji }
TO {nazwa_podmiotu | PUBLIC}
[WITH GRANT OPTION]
```

Przykład MySQL

- Aktualny użytkownik, posiadający wszelkie uprawnienia przekazuje część uprawnień użytkownikowi ***andrzej*** na serwerze ***localhost***
- Następnie odbiera mu wszelkie uprawnienia i przekazuje inne


```
GRANT CREATE , DROP , INDEX , ALTER , CREATE
TEMPORARY TABLES ON * . * TO
'andrzej'@'localhost' IDENTIFIED BY '*****' WITH
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0 ;



GRANT ALL PRIVILEGES ON `sprzedaz` . * TO
'andrzej'@'localhost' WITH GRANT OPTION ;
```

Użytkownik ma prawa do edycji danych, nie może jednak zmieniać struktur bazy danych

```
REVOKE ALL PRIVILEGES ON * . * FROM 'andrzej'@'localhost';

GRANT SELECT ,
INSERT ,
UPDATE ,
DELETE ,
RELOAD ,
SHUTDOWN ,
PROCESS ,
FILE ,
REFERENCES ,
SHOW DATABASES ,
SUPER ,
LOCK TABLES ,
EXECUTE ,
REPLICATION SLAVE ,
REPLICATION CLIENT ON * . * TO 'andrzej'@ 'localhost' WITH GRANT OPTION
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 ;
```

```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 155 to server version: 4.1.9-max

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use planowanie;
Database changed
mysql> create table test (
    -> id char(10));
ERROR 1044 (42000): Access denied for user 'andrzej'@'localhost' to database 'pl
anowanie'
mysql> select * from towary;
+----------+--------------------+------+
| IdTowaru | NazwaTowaru        | cena |
+----------+--------------------+------+
|        1 | Rura stalowa fi 10 | 0.00 |
|        2 | Rura b/s fi 12     | 1.30 |
|        3 | Rura b/s fi 15     | 1.20 |
|       10 | rura b/s fi 32     | 1.10 |
|        6 | Z|▒czka fi 10      | 2.00 |
+----------+--------------------+------+
5 rows in set (0.41 sec)

mysql> update towary set cena=1.5 where idtowaru=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from towary;
+----------+--------------------+------+
| IdTowaru | NazwaTowaru        | cena |
+----------+--------------------+------+
|        1 | Rura stalowa fi 10 | 1.50 |
|        2 | Rura b/s fi 12     | 1.30 |
|        3 | Rura b/s fi 15     | 1.20 |
|       10 | rura b/s fi 32     | 1.10 |
|        6 | Z|▒czka fi 10      | 2.00 |
+----------+--------------------+------+
5 rows in set (0.00 sec)

mysql>
```

brak uprawnień do tworzenia tabeli

są uprawnienia do aktualizacji tabeli