Bazy danych

5. Samozłączenie

SQL — podstawy

P. F. Góra

http://th-www.if.uj.edu.pl/zfs/gora/

semestr letni 2007/08

Przykład kolejowy

Tworzymy bazę danych zawierającą (uproszczony) rozkład jazdy pociągów oraz informacje o sprzedanych biletach.

Po pierwsze,

- Z każdym pociągiem stowarzyszona jest *lista* stacji, na których się on zatrzymuje.
- Z każdą stacją stowarzyszony jest *zbiór* pociągów, na których się on zatrzymuje.
- Dla każdej stacji należy podać godziny przyjazdu i odjazdu każdego pociągu, który się na niej zatrzymuje.

Pierwszy i ostatni punkt odpowiadają następującym zależnościom funkcyjnym:

$$\text{NrPociągu Stacja} \longrightarrow \text{NrKolejny} \tag{1a}$$

$$\text{NrPociągu Stacja} \longrightarrow \text{Przyjazd Odjazd} \tag{1b}$$

co prowadzi do następującej struktury tabeli

$$\text{Przystanki}\left(\underline{\text{NrPociągu}}, \underline{\text{Stacja}}, \text{NrKolejny}, \text{Przyjazd}, \text{Odjazd}\right) \tag{2}$$

Powstaje jednak pewna wątpliwość: A co się stanie jeśli jakiś pociąg nie przejeżdża przez daną stację? Czy przypadkiem zależności funkcyjne (1) nie wymagają wyspecyfikowania wszystkich możliwych par NrPociągu-Stacja? Oczywiście nie: Zależność funkcyjna ma postać implikacji: *jeżeli* podamy atrybuty poprzednika, *to* ustalamy jednoznacznie wartości atrybutów następnika. A jeżeli nie, to nie ☺.

Rola zapytań

Określmy zbioru pociągów, które zatrzymują się na danej stacji. Nie trzeba w tym celu konstruować osobnej tabeli (relacji) — *można to osiągnąć poprzez odpowiednie zapytanie*:

$$\pi_{\text{NrPociągu}}\left(\sigma_{\text{Stacja}=\text{'Koluszki'}}\left(\text{Przystanki}\right)\right) \tag{3}$$

Nie wszystkie informacje przechowuje się w postaci osobnych tabel — użyteczność baz danych polega (między innymi) na tym, że wiele rzeczy da się zrealizować poprzez zapytania.

Zadanie

Posługując się schematem tabeli Przystanki (2), znaleźć wszystkie pociągi, które przejeżdżają przez stację X oraz przez stację Y.

$$\left[\pi_{\text{NrPociągu}}\left(\sigma_{\text{Stacja='X'}}\left(\text{Przystanki}\right)\right)\right] \bowtie \left[\pi_{\text{NrPociągu}}\left(\sigma_{\text{Stacja='Y'}}\left(\text{Przystanki}\right)\right)\right] \tag{4}$$

Tabela Przystanki występuje w powyższym zapytaniu dwa razy, po obu stronach złączenia. Jest to przykład tak zwanego *samozłączenia (self-join)*.

Zadanie*

Posługując się schematem tabeli Przystanki (2), znaleźć wszystkie pociągi, które najpierw przejeżdżają przez stację X, później przez stację Y.

$$\rho_{\text{StacjaX(NrPociągu,NrX)}} \left[\pi_{\text{NrPociągu,NrKolejny}} \left(\sigma_{\text{Stacja}=\text{'X'}} (\text{Przystanki})\right)\right] \quad (5a)$$

$$\rho_{\text{StacjaY(NrPociągu,NrY)}} \left[\pi_{\text{NrPociągu,NrKolejny}} \left(\sigma_{\text{Stacja}=\text{'Y'}} (\text{Przystanki})\right)\right] \quad (5b)$$

$$\pi_{\text{NrPociągu}} \left(\sigma_{\text{NrX}<\text{NrY}} (\text{StacjaX} \bowtie \text{StacjaY})\right) \quad (5c)$$

Zapytanie (5) oczywiście także zawiera samozłączenie, tyle że w formie niejawnej.

Pytanie: Po jakim atrybucie realizowane jest złączenie w (5c)? Po atrybucie NrPociągu, bo to jest powtarzający się atrybut w tabelach StacjaX, StacjaY.

Structured Query Language

Używane standardy:

- SQL92
- SQL99
- SQL:2003

Żaden dostawca nie jest w pełni zgodny ze standardem — prawie wszyscy wprowadzają rozszerzenia, prawie nikt nie spełnia wszystkich wymogów. W dużych systemach komercyjnych odejście od wymogów nie jest wielkie

Połączenie się z serwerem

Przed rozpoczęciem pracy, nalezy wywołać proces kliencki, który połaczy się z serwerem. W MySQL może to wyglądać na przykład tak:

```
C:\>mysql -upawel -p
Enter password: **************
```

Po zakończeniu pracy trzeba się pożegnać:

```
mysql> QUIT;
Bye
```

Znacznie częściej połączenie z serwerem nawiązuje się za pomocą jakiejś aplikacji, nie zaś bezpośrednio z shella.

Utworzenie bazy danych

```
mysql> CREATE DATABASE MoiStudenci CHARACTER SET cp1250;
Query OK, 1 row affected (0.06 sec)

mysql> USE MoiStudenci;
Database changed
mysql>
```

Po wywołaniu klienta, trzeba “wejść” do wybranej bazy za pomocą instrukcji `USE` — nie tylko po jej utworzeniu, ale zawsze.

To, jakie bazy znajdują się na serwerze, możemy zobaczyć za pomocą instrukcji `SHOW DATABASES`.

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| kaskady            |
| moistudenci        |
| mysql              |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

Utworzenie tabeli

```
mysql> CREATE TABLE Studenci
    -> (NrStudenta SMALLINT UNSIGNED NOT NULL
    ->  AUTO_INCREMENT PRIMARY KEY);
Query OK, 0 rows affected (0.14 sec)
```

Jeśli okaże się to konieczne, po utworzeniu można zmienić definicję tabeli.

```
mysql> ALTER TABLE Studenci
    -> ADD COLUMN (Nazwisko VARCHAR(20) NOT NULL);
Query OK, 1 row affected (0.17 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

Składnia polecenia `CREATE TABLE`

`CREATE TABLE` NazwaTabeli (
NazwaKolumny$_1$ TypKolumny$_1$ `NULL | NOT NULL` ...,
NazwaKolumny$_2$ TypKolumny$_2$ `NULL | NOT NULL` ...,
...
)
`PRIMARY KEY` (NazwaKolumny$_p$,NazwaKolumny$_q$,...)
;

Inne opcje poznamy później.

Typy danych (atrybutów)

TINYINT	DATE	TINYBLOB
SMALLINT	TIME	BLOB
MEDIUMINT	TIMESTAMP	MEDIUMBLOB
INT	DATETIME	LONGBLOB
INTEGER	CHAR	TINYTEXT
BIGINT	VARCHAR	TEXT
REAL	BOOLEAN	LONGTEXT
DOUBLE		ENUM
FLOAT		SET
DECIMAL		
NUMERIC		

Instrukcja `DESCRIBE` podaje definicję tabeli.

```
mysql> DESCRIBE Studenci;
+------------+----------------------+------+-----+---------+----------------+
| Field      | Type                 | Null | Key | Default | Extra          |
+------------+----------------------+------+-----+---------+----------------+
| NrStudenta | smallint(5) unsigned | NO   | PRI | NULL    | auto_increment |
| Nazwisko   | varchar(20)          | NO   |     |         |                |
+------------+----------------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

Niepotrzebną tabelę usuwamy za pomocą instrukcji `DROP TABLE`.

```
mysql> DROP TABLE Studenci;
Query OK, 0 rows affected (0.02 sec)
```

Lepiej ją utworzyć od początku dobrze niż dodawać kolumna po kolumnie. . . ☺

```
mysql> CREATE TABLE Studenci
    -> (NrStudenta SMALLINT UNSIGNED NOT NUll
    ->  AUTO_INCREMENT PRIMARY KEY,
    ->  Imie VARCHAR(20),
    ->  Nazwisko VARCHAR(20) NOT NULL,
    ->  Uwagi VARCHAR(30),
    ->  Grupa CHAR(2) NOT NULL));
Query OK, 0 rows affected (0.06 sec)
mysql> DESCRIBE Studenci;
+------------+---------------------+------+-----+---------+----------------+
| Field      | Type                | Null | Key | Default | Extra          |
+------------+---------------------+------+-----+---------+----------------+
| NrStudenta | smallint(5) unsigned | NO   | PRI | NULL    | auto_increment |
| Imie       | varchar(20)         | YES  |     | NULL    |                |
| Nazwisko   | varchar(20)         | NO   |     |         |                |
| Uwagi      | varchar(30)         | YES  |     | NULL    |                |
| Grupa      | char(2)             | NO   |     |         |                |
+------------+---------------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Wstawianie wartości do tabeli

```
mysql> SET CHARACTER SET cp1250;
Query OK, 0 rows affected (0.00 sec)
```

Gdybyśmy tego nie zrobili, MySQL używałby takiego zestawu znaków, jaki obowiązywał domyślnie w momencie połączenia z serwerem. Kwestie określania zestawu znaków to osobliwość MySQLa ☹

```
mysql> INSERT INTO Studenci VALUES
    -> (1,'Milena','Zahorska','wt','uzupełniające');
Query OK, 1 row affected (0.02 sec)
```

Co teraz jest w tabeli?

```
mysql> SELECT * FROM Studenci;
+-------------+--------+----------+---------------+--------+
| NrStudenta | Imie   | Nazwisko | Uwagi         | Grupa |
+-------------+--------+----------+---------------+--------+
|           1 | Milena | Zahorska | uzupełniające | wt    |
+-------------+--------+----------+---------------+--------+
1 row in set (0.00 sec)
```

Składnia zapytania `SELECT`;

```
SELECT [DISTINCT] lista_select
FROM tabela_lub_zapytanie
WHERE warunek_logiczny
GROUP BY wyrażenie_grupujące
HAVING warunek_wyszukiwania_po_grupowaniu
ORDER BY wyrażenie_porządkujące [ASC | DESC]
;
```

W zapytaniu `SELECT` *wszystkie klauzule muszą występować w powyższej kolejności.* Niektóre klauzule można opuścić, ale to, co jest, musi być w tej kolejności.

Dodajmy więcej danych

```
mysql> INSERT INTO Studenci VALUES
    -> (2,'Sylwester','Tomiec','uzupełniające','wt'),
    -> (3,'Michał','Gajewczyk','''','wt');
Query OK, 2 rows affected (0.02 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Studenci;
+------------+-----------+-----------+---------------+-------+
| NrStudenta | Imie      | Nazwisko  | Uwagi         | Grupa |
+------------+-----------+-----------+---------------+-------+
|          1 | Milena    | Zahorska  | uzupełniające | wt    |
|          2 | Sylwester | Tomiec    | uzupełniające | wt    |
|          3 | Michał    | Gajewczyk | '             | wt    |
+------------+-----------+-----------+---------------+-------+
3 rows in set (0.00 sec)
```

Wykorzystanie opcji AUTOINCREMENT

```
mysql> INSERT INTO Studenci (Imie,Nazwisko,Grupa) VALUES
    -> ('Rafał','Świderski','wt'),
    -> ('Maciej','Matowicki','wf'),
    -> ('Janko','Muzykant','nd');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
mysql> SELECT * FROM Studenci;
+------------+-----------+-----------+---------------+-------+
| NrStudenta | Imie      | Nazwisko  | Uwagi         | Grupa |
+------------+-----------+-----------+---------------+-------+
|          1 | Milena    | Zahorska  | uzupełniające | wt    |
|          2 | Sylwester | Tomiec    | uzupełniające | wt    |
|          3 | Michał    | Gajewczyk | ’             | wt    |
|          4 | Rafał     | Świderski | NULL          | wt    |
|          5 | Maciej    | Matowicki | NULL          | wf    |
|          6 | Janko     | Muzykant  | NULL          | nd    |
+------------+-----------+-----------+---------------+-------+
6 rows in set (0.00 sec)
```

Zapytanie `UPDATE`

```
mysql> UPDATE Studenci
    -> SET Uwagi=NULL
    -> WHERE NrStudenta=3;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Studenci
    -> SET Grupa='wt' WHERE Grupa='wf';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM Studenci;
+------------+-----------+-----------+---------------+-------+
| NrStudenta | Imie      | Nazwisko  | Uwagi         | Grupa |
+------------+-----------+-----------+---------------+-------+
|          1 | Milena    | Zahorska  | uzupełniające | wt    |
|          2 | Sylwester | Tomiec    | uzupełniające | wt    |
|          3 | Michał    | Gajewczyk | NULL          | wt    |
|          4 | Rafał     | Świderski | NULL          | wt    |
|          5 | Maciej    | Matowicki | NULL          | wt    |
|          6 | Janko     | Muzykant  | NULL          | nd    |
+------------+-----------+-----------+---------------+-------+
6 rows in set (0.00 sec)
```

Zapytanie `DELETE`

Usuwanie wierszy spełniających podane kryterium:

```
mysql> DELETE FROM Studenci
    -> WHERE Imie='Janko';
Query OK, 1 row affected (0.03 sec)
```

Dodajmy coś jeszcze:

```
mysql> INSERT INTO Studenci (Imie,Nazwisko,Grupa) VALUES
    -> ('Michał','Czubek','pt'),('Marcin','Pyra','pt'),
    -> ('Marcin','Baranowski','pt');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM Studenci;
+------------+-----------+------------+---------------+-------+
| NrStudenta | Imie      | Nazwisko   | Uwagi         | Grupa |
+------------+-----------+------------+---------------+-------+
|          1 | Milena    | Zahorska   | uzupełniające | wt    |
|          2 | Sylwester | Tomiec     | uzupełniające | wt    |
|          3 | Michał    | Gajewczyk  | NULL          | wt    |
|          4 | Rafał     | Świderski  | NULL          | wt    |
|          5 | Maciej    | Matowicki  | NULL          | wt    |
|          7 | Michał    | Czubek     | NULL          | pt    |
|          8 | Marcin    | Pyra       | NULL          | pt    |
|          9 | Marcin    | Baranowski | NULL          | pt    |
+------------+-----------+------------+---------------+-------+
8 rows in set (0.01 sec)
```

Brakuje numeru ’6’ — `AUTOINCREMENT` “zapamiętał”, że on tam był.

`AUTOINCREMENT` zawsze startuje od *high water mark*.

Jawnie wyspecyfikowana *lista_select*

```
mysql> SELECT Imie, Nazwisko FROM Studenci;
+-----------+------------+
| Imie      | Nazwisko   |
+-----------+------------+
| Milena    | Zahorska   |
| Sylwester | Tomiec     |
| Michał    | Gajewczyk  |
| Rafał     | Świderski  |
| Maciej    | Matowicki  |
| Michał    | Czubek     |
| Marcin    | Pyra       |
| Marcin    | Baranowski |
+-----------+------------+
8 rows in set (0.01 sec)
```

Gwiazdka w zapytaniu `SELECT` (`SELECT * FROM...`) oznacza “wszystkie kolumny”.

Problem duplikatów

```
mysql> SELECT Imie
    -> FROM Studenci;
+-----------+
| Imie      |
+-----------+
| Milena    |
| Sylwester |
| Michał    |
| Rafał     |
| Maciej    |
| Michał    |
| Marcin    |
| Marcin    |
+-----------+
8 rows in set (0.00 sec)
```

```
mysql> SELECT DISTINCT Imie
    -> FROM Studenci;
+-----------+
| Imie      |
+-----------+
| Milena    |
| Sylwester |
| Michał    |
| Rafał     |
| Maciej    |
| Marcin    |
+-----------+
6 rows in set (0.00 sec)
```

Operator `LIKE`

```
mysql> SELECT Imie, Nazwisko
    -> FROM Studenci
    -> WHERE Imie LIKE 'Mi%';
+--------+-----------+
| Imie   | Nazwisko  |
+--------+-----------+
| Milena | Zahorska  |
| Michał | Gajewczyk |
| Michał | Czubek    |
+--------+-----------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT Imie, Nazwisko
    -> FROM Studenci
    -> WHERE Imie LIKE '%ł';
+--------+-----------+
| Imie   | Nazwisko  |
+--------+-----------+
| Michał | Gajewczyk |
| Rafał  | Świderski |
| Michał | Czubek    |
+--------+-----------+
3 rows in set (0.00 sec)
```

Wykorzystanie `ORDER BY`

```
mysql> SELECT Imie, Nazwisko
    -> FROM Studenci
    -> WHERE Grupa='wt'
    -> ORDER BY Nazwisko ASC;
+-----------+-----------+
| Imie      | Nazwisko  |
+-----------+-----------+
| Michał    | Gajewczyk |
| Maciej    | Matowicki |
| Sylwester | Tomiec    |
| Milena    | Zahorska  |
| Rafał     | Świderski |
+-----------+-----------+
5 rows in set (0.02 sec)
```

Problemy z poprawnym sortowaniem polskich znaków diakrytycznych ☹