KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

Jan Laštovička
jan.lastovicka@upol.cz
www.inf.upol.cz/lide/jan-lastovicka
17. listopadu 12, 771 46 Olomouc

Databázové systémy

1. Relace

Představme si, že chceme vědět, které děti chodí do jisté školky. U každého dítěte chceme navíc znát jeho věk. Víme, že do školky chodí tři děti: Anna, Bert a Cyril. Anně jsou tři roky, Bertovi a Cyrilovi jsou čtyři roky. Tyto informace můžeme přehledně zapsat do následující tabulky.

name	age
Anna	3
Bert	4
Cyril	4

Tabulka má záhlaví popisující sloupce: `name` a `age`. Pod záhlavím následuje tělo tabulky obsahující tři řádky. Tabulku můžeme použít při hledání odpovědí na různé otázky. Například:

1. Chodí Cyril do školky?
2. Chodí Daniela do školky?
3. Kolik let je Anně?
4. Kterým dětem jsou čtyři roky?

Je jasné, že odpovědi na otázky se nezmění, když prohodíme řádky tabulky. Například:

name	age
Bert	4
Cyril	4
Anna	3

Pořadí řádků tedy nepřikládáme žádný význam. Podobně můžeme změnit pořadí sloupců a informace uložená v tabulce se nezmění:

age	name
3	Anna
4	Bert
4	Cyril

Tedy ani pořadí sloupců tabulky nenese žádný význam.

Konečně pokud v tabulce uvedeme nějaký řádek vícekrát ponese tabulka stejnou informaci, jako bychom jej zde uvedli jen jednou:

name	age
Anna	3
Anna	3
Bert	4
Cyril	4

Z tabulky stále vyplývá, že Anna chodí do školky a jsou jí tři roky.

Zamysleme se nad hodnotami, které můžeme do buněk tabulky vkládat. Shodneme se, že v sloupci name musí být jména (jisté řetězce znaků abecedy) a v sloupci age můžou být řekněme pouze celá čísla od tří do sedmi. Hodnota uvedená v každé buňce musí být tedy typu určeném sloupcem buňky.

Nyní budeme hledat vhodnou reprezentaci informací o školce, kterou chceme uchovávat. Tabulka je výhodný nástroj na zobrazení informace, ale jako věrná reprezentace informace selhává. Vždyť všechny výše uvedené tabulky zobrazují přesně stejnou informaci.

Začneme tím, že si ujasníme, co je to typ. *Typem* rozumíme pojmenovanou množinu hodnot. Pro každé přirozené číslo i je `varchar(`i`)` typem, který pojmenovává množinu všech řetězců délky nejvýše i a `integer` je typem, který dává jméno množině celých čísel od $-2\,147\,483\,648$ do $2\,147\,483\,647$.

Řetězec můžeme pro zvýšení čitelnosti vložit mezi jednoduché uvozovky. Například `'Anna'` je řetězec složený z těchto čtyř znaků: `A`, `n`, `n`, `a`. Pokud hodnota v patří do množiny typu T, říkáme, že hodnota v *je typu* T. Řetězec `'Anna'` je typu `varchar(4)`, ale už není typu `varchar(3)`. Pro náš příklad se školkou pomineme to, že ne všechny řetězce se hodí pro jména dětí. Hodnoty 3,4,5,6 a 7 jsou typu `integer`. Vidíme, že typ `integer` je výrazně bohatší, než potřebujeme.

Sloupce tabulky zobrazují *atributy*. Atribut má jméno, které budeme psát anglicky. Tedy `name` a `age` jsou jména atributů. Každý atribut dále určuje jméno typu. Například atribut jménem `name` určuje jméno typu `varchar(10)` a atribut jménem `age` jméno typu `integer`. Atribut je dán pouze svým jménem a jménem jeho typu.

Konečnou množinu atributů s unikátními jmény nazveme *záhlavím*. V zápisech budeme běžně vynechávat informaci o typech atributů a v záhlaví budeme uvádět pouze jména atributů. Například můžeme říci, že množina $\{$`name`, `age`$\}$ je záhlavím, ale musí být jasné, jakého typu atributy `name` a `age` jsou. Může se stát, že v různých záhlavích mají atributy se stejným jménem různý typ. Například atribut jménem `name` může v jednom záhlaví mít typ `varchar(5)` a v jiném `varchar(10)`.

Záhlaví přirozeně zobrazíme jako záhlaví tabulky:

name varchar(10)	age integer

Pokud jsou typy atributů známé, můžeme jména typů vynechat:

name	age

Tím, že je záhlaví množina atributů, nezáleží na pořadí v jakém je v záhlaví tabulky uvedeme. Tedy předchozí záhlaví tabulky a níže uvedené záhlaví tabulky zobrazuje stejné záhlaví.

age	name

Informace, že tříletá Anna chodí do školky, je zachycena jedním řádkem tabulky. Tuto informaci reprezentujeme n-ticí. Dříve, než si definujeme n-tici uvažujme libovolný atribut A. *Komponenta* přiřazuje atributu A hodnotu, která je stejného typu, jako je typ atributu A. Například máme komponentu, která atributu `name` typu `varchar(10)` přiřadí hodnotu `'Anna'`. Uvažujme množinu komponent t s jedinečnými atributy. Množinu t nazýváme *n-ticí nad* $A_1, \ldots, A_n$, kde $A_1, \ldots, A_n$ jsou všechny atributy komponent v t. Množina $\{A_1, \ldots, A_n\}$ se nazývá *záhlaví n-tice t*. Například množina dvou komponent, kde atributu `name` je přiřazena hodnota `'Anna'` a atributu `age` hodnota `3` je n-tice nad `name` a `age`. Přirozeně můžeme n-tici zobrazit jako tabulku s jedním řádkem:

name	age
Anna	3

Tabulka

name	age
Anna	

žádnou n-tici nezobrazuje, protože atributu `age` není přiřazena žádná hodnota. Také tabulka

name	age
Anna	3, 4

nezobrazuje n-tici, protože atributu `age` jsou přiřazeny dvě hodnoty.

Konečně všechny informace z úvodního příkladu o školce reprezentujeme relací. Je dáno záhlaví $\{A_1, \ldots, A_n\}$. *Relace r nad atributy* $A_1, \ldots, A_n$ se skládá ze záhlaví $\{A_1, \ldots, A_n\}$ a těla. *Tělo relace* je konečná množina n-tic nad $A_1, \ldots, A_n$. Uvažujme

například relaci r dětí ve školce nad `name` a `age`, která obsahuje tři n-tice t_1, t_2, t_3. Každá z tří n-tic t_1, t_2, t_3 má dvě komponenty: jednu s atributem `name` a druhou s atributem `age`. První n-tice t_1 přiřazuje atributu `name` hodnotu `'Anna'` a atributu `age` hodnotu `3`, t_2 přiřazuje atributům hodnoty `'Bert'` a `4` a konečně t_3 přiřazuje hodnoty `'Cyril'` a `4`. Relaci r můžeme přehledně zobrazit tabulkou:

name varchar(10)	age integer
Anna	3
Bert	4
Cyril	4

Pokud jsou známé typy atributů name a age můžeme je z tabulky vynechat:

name	age
Anna	3
Bert	4
Cyril	4

Protože tělo relace je množina n-tic, tak přirozeně nezáleží na pořadí v jakém řádky zobrazující n-tice v tabulce uvedeme.

Tabulka

name	age
Anna	
Bert	3, 4
Cyril	4

nezobrazuje relaci, protože ani první ani druhý řádek nezobrazuje n-tici.

Tělo relace je množina n-tic a tato množina může být prázdná. Relaci, jejíž tělo je prázdná množina, říkáme *prázdná relace*.

Relace nemají jméno. Chceme-li relaci pojmenovat musíme ji dát do proměnné. Proměnným, které uchovávají relace, říkáme přirozeně *relační proměnné*. Relační proměnná má jméno a je jistého typu, který je dán záhlavím (konečnou množinou atributů). Relační proměnná může uchovávat pouze relace s odpovídajícím záhlavím. Například definujeme relační proměnnou `child` typu {`name varchar(10)`, `age integer`} a nastavíme její hodnotu na relaci dětí školky. Situaci můžeme zobrazit pojmenovanou tabulkou:

`child`

name	age
Anna	3
Bert	4
Cyril	4

Nainstalujte si databázový systém PostgreSQL (`https://www.postgresql.org`), spusťte konzolový nástroj psql (SQL shell) a přihlaste se do databáze. Za znakem mřížka `#` databáze očekává příkaz. Začneme příkazem, který deklaruje relační proměnnou. Pokud R je jméno relační proměnné, $n > 0$, $A_1, \ldots, A_n$ jsou jména atributů a $T_1, \ldots, T_n$ jsou jména typů, pak

```
CREATE TABLE R (
        A1 T1 NOT NULL,
        ⋮
        An Tn NOT NULL,
        UNIQUE (A1, ..., An)
);
```

je *příkaz deklarující relační proměnnou*. Příkaz se vykoná tak, že deklaruje relační proměnnou R nad $A_1, \ldots, A_n$, kde A_i je typu T_i pro každé $1 \leq i \leq n$. Hodnota proměnné R bude prázdná relace nad $A_1, \ldots, A_n$.

Nyní zadejte následující příkaz:

```
# CREATE TABLE child (
    name varchar(10) NOT NULL,
    age integer NOT NULL,
    UNIQUE (name, age)
);
```

Deklarovali jste relační proměnnou `child` nad `name` a `age`, kde `name` je typu `varchar(10)` a `age` typu `integer`. Hodnota relační proměnné `child` je prázdná relace nad `name` a `age`.

Následujícím příkazem zobrazíte hodnotu relační proměnné. Pokud R je relační proměnná, pak

```
TABLE R;
```

je *příkaz zobrazující hodnotu relační proměnné*.

Například po zadání:

```
# TABLE child;
```

obdržíme:

```
 name | age
------+-----
(0 rows)
```

Chceme-li, aby relační proměnná `child` obsahovala neprázdnou relaci, musíme změnit její hodnotu. Relace však z principu měnit nelze. Můžeme ale s použitím hodnoty proměnné `child` vytvořit novou relaci a tu vložit do proměnné `child`. K tomu použijeme následující příkaz.

Předpokládejme, že R je relační proměnná nad $A_1, \ldots, A_n$, kde A_i je typu T_i pro každé $1 \leq i \leq n$ a $\{t_1, \ldots, t_m\}$ je konečná neprázdná množina n-tic nad $A_1, \ldots, A_n$, která je disjunktní s tělem relace R, dále předpokládejme, že pro každé $1 \leq i \leq n$ a $1 \leq j \leq m$ je v_{ij} hodnota, kterou přiřazuje n-tice t_j atributu A_i, pak

```
INSERT INTO R (A_1, ..., A_n) VALUES
     ( v_11, ..., v_1n ),
             ⋮
     ( v_m1, ..., v_mn );
```

je příkaz *přidání n-tic do relace*.

Relaci proměnné R označíme r. Uvažujme relaci r', která má stejné záhlaví jako relace r a jejíž tělo vznikne sjednocením těla r a množiny $\{t_1, \ldots, t_m\}$. Příkaz nastaví hodnotu proměnné R na r'.

Například po vykonání příkazu:

```
# INSERT INTO child (name, age) VALUES
       ( 'Anna', 3 ),
       ( 'Bert', 4 ),
       ( 'Cyril', 4 );
```

má relace proměnné `child` tělo obsahující tři n-tice:

```
# TABLE child;

 name  | age
-------+-----
 Bert  |   4
 Cyril |   4
 Anna  |   3
(3 rows)
```

Všimněte si, že pořadí řádků zobrazené tabulky neodpovídá pořadí n-tic v příkazu přidání do relace. U vás může být dokonce pořadí řádků jiné.

Pokud do školky začne chodit nové dítě, můžeme jej snadno do relace `child` přidat příkazem:

```
INSERT INTO child (name, age) VALUES ('Daniela', 5);
```

Přesvědčíme se, že došlo k přidání:

```
# TABLE child;

  name   | age
---------+-----
 Bert    |   4
 Cyril   |   4
 Anna    |   3
 Daniela |   5
(4 rows)
```

Tuto část zakončíme příkazem rušící proměnné. Pokud R je relační proměnná, pak

```
DROP TABLE R;
```

je *příkaz zrušení relační proměnné*.

Tedy relační proměnnou `child` zrušíme příkazem

```
DROP TABLE child;
```