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

2. Množinové operace a restrikce

1 Relační výraz

Výrazu, jehož hodnota je relace, říkáme *relační výraz*. Pokud R je relační proměnná, pak

```
( TABLE R )
```

je relační výraz, jehož hodnota je hodnota relační proměnné R.

Pokud v je relační výraz, pak

```
v;
```

je příkaz, který vytiskne hodnotu relačního výrazu v. Například příkaz

```
( TABLE child );
```

vytiskne hodnotu relační proměnné `child`. Nejvíce vnější závorky v relačních výrazech budeme vynechávat, proto můžeme předchozí příkaz zjednodušit:

```
TABLE child;
```

Podíváme se postupně na základní operace s relacemi a jak je zapsat pomocí relačních výrazů. *Relační operace* je operace, která očekává na vstupu jednu nebo více relací a vrací na výstup jednu relaci.

Záhlaví relace budeme také nazývat *typem relace*. Můžeme například říci, že dvě relace jsou stejného typu. Tím se vyjádří, že mají stejná záhlaví.

Hodnota relačního výrazu se může měnit v závislosti na hodnotách relačních proměnných. Typ možných hodnot relačního výrazu však zůstává vždy stejný. Tento typ nazýváme *typem* relačního výrazu. Opět můžeme říci, že dva relační výrazy jsou stejného typu. To znamená, že jejich hodnoty jsou vždy stejného typu (mají stejné záhlaví).

2 Množinové operace

Nejprve se seznámíme s relačními operacemi, které vychází ze skutečnosti, že tělo relace je množina n-tic. Dává tedy smysl například uvažovat relaci, jejichž tělo vznikne sjednocením těl jiných relací. Musíme si však dát pozor na typ relací, jejichž těla sjednocujeme. Neplatí, že sjednocení těl dvou libovolných relací vznikne tělo nějaké relace. Relace musí být stejného typu.

Pro příklady si zavedeme dvě níže uvedené relace stejného typu uložené v proměnných `child1` a `child2`.

`child1`

name	age
Anna	3
Bert	4
Cyril	4

`child2`

name	age
Bert	4
Cyril	4
Daniela	5

Představíme si relační operace sjednocení, průnik a rozdíl, které mají na vstupu dvě relace stejného typu. Výstupem všech tří operací je relace stejného typu, jako vstupní relace. Výstupní relace má stejné záhlaví jako obě vstupní relace. Těla výstupních relací se u jednotlivých operací liší. Předpokládejme, že r_1 a r_2 jsou relace se stejným záhlavím h. Dále předpokládejme, že v_1 a v_2 jsou relační výrazy stejného typu.

Sjednocení. Výstupem *sjednocení* r_1 *a* r_2 je relace, kde tělo je množinové sjednocení těl r_1 a r_2.

Například sjednocením `child1` a `child2` obdržíme relaci:

name	age
Anna	3
Bert	4
Cyril	4
Daniela	5

Pro relační výrazy v_1 a v_2 stejného typu je

```
( v1 UNION v2 )
```

relační výraz, jehož hodnota je sjednocení hodnot výrazů v_1 a v_2.

Například máme:

```
# ( TABLE child1 ) UNION ( TABLE child2 );

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

Průnik. Výstupem *průniku* r_1 *a* r_2 je relace, kde tělo je rovno množinovému průniku těl r_1 a r_2.

Například průnik `child1` a `child2` je roven relaci:

name	age
Bert	4
Cyril	4

Pro relační výrazy v_1 a v_2 stejného typu je

```
(v1 INTERSECT v2)
```

relační výraz, jehož hodnota je množinovým průnikem hodnot výrazů v_1 a v_2.

Například:

```
# ( TABLE child1 ) INTERSECT ( TABLE child2 );

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

Rozdíl. A konečně výstupem *rozdílu* r_1 *a* r_2 je relace, kde tělo je rovno množinovému rozdílu těl r_1 a r_2.

Například rozdíl relací `child1` a `child2` je relace:

name	age
Anna	3

Pro relační výrazy v_1 a v_2 stejného typu je

```
(v1 EXCEPT v2)
```

relační výraz, jehož hodnota je rozdílem hodnot výrazů v_1 a v_2.

Například máme:

```
# ( TABLE child1 ) EXCEPT ( TABLE child2 );

 name | age
------+-----
 Anna |   3
(1 row)
```

Velkou výhodou relačních operací je, že výstup jedné operace můžeme použít na vstup jiné. Například hodnota následujícího relačního výrazu je relace obsahující děti, které jsou právě v jedné z relací `child1` a `child2`:

```
# ( ( TABLE child1 ) EXCEPT ( TABLE child2 ) )
      UNION
  ( ( TABLE child2 ) EXCEPT ( TABLE child1 ) );

  name   | age
---------+-----
 Anna    |   3
 Daniela |   5
(2 rows)
```

3 Restrikce

Začneme pozorováním, že libovolná podmnožina těla relace r je opět tělo jisté relace, která je stejného typu jako r. Podmnožinu b' těla b relace r můžeme vybrat tak, že zadáme podmínku na n-tice množiny b. Množina b' pak bude obsahovat právě ty n-tice z b, které splňují zadanou podmínku.

Začneme velmi jednoduchým typem podmínek. Předpokládejme, že máme záhlaví $\{A_1, \ldots, A_n\}$, kde atribut A_i je typu T_i pro každé $1 \leq i \leq n$. Vezmeme libovolné $1 \leq j \leq n$ a libovolnou hodnotu v typu T_j, pak výraz

```
( A_j = v )
```

je *podmínkou nad* $A_1, \ldots, A_n$. Například uvažujme záhlaví $\{\texttt{name}, \texttt{age}\}$, kde jak jsme zvyklí, je `name` typu `varchar(10)` a `age` typu `integer`. Výrazy

```
( name = 'Anna' )
( age = 3 )
```

jsou podmínky nad `name` a `age`. Opět nejvíce vnější závorky budeme vynechávat. Můžeme tedy jednoduše psát:

```
name = 'Anna'
age = 3
```

Máme dánu n-tici t nad $A_1, \ldots, A_n$. Podmínka `(` A_j `=` v `)` *je* v *t splněna*, pokud t atributu A_j přiřazuje v.

Například uvažujme n-tici

name	age
Anna	3

Podmínky `name = 'Anna'` a `age = 3` jsou v n-tici splněné, ale podmínky `name = 'Bert'` a `age = 4` v n-tici splněné nejsou.

Je dána relace r nad $A_1, \ldots, A_n$ a podmínka c nad $A_1, \ldots, A_n$, pak *restrikcí* r *vzhledem k* c rozumíme relaci r', která má stejné záhlaví jako relace r a jejíž tělo obsahuje právě ty n-tice z těla r, které splňují podmínku c.

Například uvažujme relaci r:

name	age
Anna	3
Bert	4
Cyril	4

Restrikce r vzhledem k podmínce `age = 4` je relace:

name	age
Bert	4
Cyril	4

Uvažujme relační výraz v, jehož hodnota je relace r nad $A_1, \ldots, A_n$, R je jméno relační proměnné a c je podmínka nad $A_1, \ldots, A_n$, pak

```
( SELECT *
  FROM   v AS R
  WHERE  c )
```

je relační výraz, jehož hodnota je restrikce relace r vzhledem k c. Relační proměnná R pojmenovává hodnotu výrazu v a zatím nehraje žádnou roli.

Například uvažujme relační proměnnou `child`:

```
# TABLE child;

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

Pak restrikci relace `child` vzhledem k `age = 4` získáme takto:

```
# SELECT *
  FROM ( TABLE child ) AS t
  WHERE  age = 4;

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

Uvažujme nyní relační proměnnou `child`:

```
# TABLE child;

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

Otázka je, jak zjistit, které děti jsou ve věku čtyř let a bydlí v ulici Kosinova. Odpověď můžeme získat průnikem dvou restrikcí relace child:

```
# ( SELECT *
    FROM ( TABLE child ) AS t
    WHERE  age = 4 )
    INTERSECT
  ( SELECT *
    FROM ( TABLE child ) AS t
    WHERE  street = 'Kosinova' );

 name  | age |  street
-------+-----+----------
 Cyril |   4 | Kosinova
(1 row)
```

Podobně odpověď na otázku, které děti jsou ve věku tří let nebo bydlí v ulici Mahlerova poskytne příkaz:

```
# ( SELECT *
    FROM ( TABLE child ) AS t
    WHERE  age = 3 )
    UNION
  ( SELECT *
    FROM ( TABLE child ) AS t
    WHERE  street = 'Mahlerova' );

 name | age |  street
------+-----+-----------
 Bert |   4 | Mahlerova
 Anna |   3 | Kosinova
(2 rows)
```

A konečně následuje příkaz, který zjistí, kterým dětem nejsou tři roky:

```
# ( TABLE child )
    EXCEPT
  ( SELECT *
    FROM ( TABLE child ) AS t
    WHERE  age = 3 );

 name  | age |  street
-------+-----+-----------
 Bert  |   4 | Mahlerova
 Cyril |   4 | Kosinova
(2 rows)
```

Vidíme, že průnik, sjednocení a rozdíl relací můžeme použít pro vyjádření složených podmínek (konjunkce, disjunkce a negace). Pohodlnější však bude si přímo zavést složené podmínky. Proto pokud c_1 a c_2 jsou podmínky nad $A_1, \ldots, A_n$, pak i

```
( c1 AND c2 )
( c1 OR c2 )
( NOT c1 )
```

jsou podmínky nad $A_1, \ldots, A_n$. Vezměme n-tici t nad $A_1, \ldots, A_n$. Pak podmínka `( c1 AND c2 )` *je splněna pro* t pokud obě podmínky v_1 a v_2 jsou pro t splněny, podmínka `( c1 OR c2 )` *je splněna pro* t pokud aspoň jedna z podmínek v_1 a v_2 je splněna pro t a konečně podmínka `(NOT c1 )` *je splněna pro* t, pokud c_1 není splněna pro t.

Předchozí otázky můžou být tedy jednodušeji zodpovězeny příkazy:

```
# SELECT  *
  FROM  ( TABLE child ) AS t
  WHERE ( age = 4 )
  AND   ( street = 'Kosinova' );

 name  | age |  street
-------+-----+----------
 Cyril |   4 | Kosinova
(1 row)

# SELECT  *
  FROM  ( TABLE child ) AS t
  WHERE ( age = 3 )
  OR    ( street = 'Mahlerova' );

 name | age |  street
------+-----+-----------
 Anna |   3 | Kosinova
 Bert |   4 | Mahlerova
(2 rows)

# SELECT *
  FROM ( TABLE child ) AS t
  WHERE  NOT ( age = 3 );

 name  | age |  street
-------+-----+-----------
 Bert  |   4 | Mahlerova
 Cyril |   4 | Kosinova
(2 rows)
```

Uvažujme relační proměnnou uchovávající relaci, která vyjadřuje, jakou známku dostali studenti z matematiky a informatiky:

student

name	mathematics	informatics
Anna	2	2
Bert	1	3
Cyril	3	3

Můžeme chtít získat studenty, kteří mají stejnou známku jak z matematiky tak informatiky. Za předpokladu, že studenti dostávají známky od jedné do pěti, můžeme odpověď získat kostrbatým dotazem:

```
SELECT        *
FROM        ( TABLE student ) AS t
WHERE ( ( ( ( mathematics = 1 AND informatics = 1 )
OR          ( mathematics = 2 AND informatics = 2 ) )
OR          ( mathematics = 3 AND informatics = 3 ) )
OR          ( mathematics = 4 AND informatics = 4 ) )
OR          ( mathematics = 5 AND informatics = 5 )
```

Bylo by rozhodně užitečnější mít podmínku, která je splněna, pokud n-tice dvěma zadaným atributům přiřazuje stejnou hodnotu.

Opět uvažujme záhlaví $\{A_1, \ldots, A_n\}$, kde atribut A_i je typu T_i pro každé $1 \leq i \leq n$. Vezmeme libovolné $1 \leq j, k \leq n$ takové že T_j se rovná T_k. Tedy atributy A_j a A_k mají stejný typ. Poté

(A_j = A_k)

je také *podmínka nad* $A_1, \ldots, A_n$.

Vezměme n-tici t nad $A_1, \ldots, A_n$. Pak podmínka (A_j = A_k) je v n-tici t *splněna*, pokud t přiřazuje stejnou hodnotu atributu A_j i atributu A_k.

Například uvažujme n-tici:

name	mathematics	informatics	physics
Anna	2	2	1

nad `name`, `mathematics`, `informatics` a `physics`. Podmínka (`mathematics = informatics`) je v n-tici splněna, ale podmínka (`mathematics = physics`) v n-tici splněna není.

Nyní již hravě získáme studenty, kteří získaly z obou předmětů stejné známky:

```
# SELECT * FROM ( TABLE student ) AS t
  WHERE  mathematics = informatics;

 name  | mathematics | informatics
-------+-------------+-------------
 Anna  |           2 |           2
 Cyril |           3 |           3
(2 rows)
```