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

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

Databázové systémy ◇ poznámky k přednášce

8. Klauzule SELECT výrazu

verze z 19. listopadu 2024

1 Skalární výraz

Výraz, jehož hodnota je skalárního typu, se nazývá **skalární výraz**. Skalární výraz má vždy pevně určený skalární typ.

Připomeňme si, že Y označuje množinu atributů. Předpokládáme, že pro každý atribut $y \in Y$ máme skalární typ $\texttt{scalar\_type}_y$ takový, že množina hodnot typu $\texttt{scalar\_type}_y$ je rovna doméně D_y atributu y. Například pro `movie_title` je $\texttt{scalar\_type}_{\texttt{movie\_title}}$ rovno `text` tedy množině všech řetězců nad jistou abecedou.

Mějme relační schéma S. Každý atribut $y \in S$ je skalární výraz nad S typu $\texttt{scalar\_type}_y$. Například uvažujme relační schéma $S = \{\texttt{title}, \texttt{year}, \texttt{length}\}$, pak `title` je skalární výraz nad S typu `text`.

Hodnota skalárního výrazu nad S se určuje vzhledem k n-tici t nad S. Hodnota skalárního výrazu y nad R vzhledem k t je $t(y)$. Například hodnota skalárního výrazu `title` nad $S = \{\texttt{title}, \texttt{year}, \texttt{length}\}$ vzhledem k n-tici

$$t = \{\langle \texttt{title}, \texttt{The Matrix}\rangle, \langle \texttt{year}, 1999\rangle, \langle \texttt{length}, 136\rangle\}$$

snad S je $t(\texttt{title}) =$ '`The Matrix`'.

Libovolná skalární hodnota d typu ***`scalar_type`*** je skalárním výrazem nad libovolným schématem, d je typu ***`scalar_type`*** a vyhodnocuje se v každé n-tici na sebe samu. Například skalární hodnota '`The Matrix`' je skalárním výrazem, který se vždy vyhodnotí na '`The Matrix`'.

Skalární výrazy lze skládat za pomoci operátorů. Operátory mohou být **unární**, nebo **binární**. Pokud ***`operator`*** je unární operátor a ***`operand`*** skalární výraz nad S, pak

```
( operator operand )
```

je skalární výraz nad S. Typ vzniklého skalárního výrazu je závislý na typu operátoru i operandu. **Aritmetické operátory** vyžadují, aby byly operandy typu číslo, výsledný výraz je pak také vždy typu číslo (`integer`).

Představíme si aritmetický unární operátor `-`. U operátoru `-` se nepíše mezera mezi operátorem a operandem. Například `(-1)` je skalární výraz. Opět nejvíce vnější závorky můžeme vynechat a psát například, že `-1` je skalární výraz. Také

`-year` je skalární výraz nad $S = \{\texttt{title}, \texttt{year}, \texttt{length}\}$ ale `-'The Matrix'` nebo `-title` není skalární výraz nad S.

Pokud ***operator*** je binární operátor a ***operand1*** a ***operand2*** jsou skalární výrazy nad S, pak

```
( operand1 operator operand2 )
```

je skalární výraz nad S.

Představíme si několik binárních aritmetických operátorů: `+` (součet), `-` (rozdíl), `*` (součin), `/` (celočíselný podíl), `%` (zbytek po celočíselném podílu). Například `1 + 2` je skalární výraz nebo `(1 + year) * 2` je skalární výraz nad $\{\texttt{title}, \texttt{year}, \texttt{length}\}$.

Hodnota skalárního výrazu tvořeného aritmetickým operátorem je výsledek příslušné aritmetické operace na hodnoty operandů. Například pro n-tici:

$$t = \{\langle \texttt{title}, \texttt{The Matrix}\rangle, \langle \texttt{year}, 1999\rangle, \langle \texttt{length}, 136\rangle\}$$

je hodnota skalárního výraz `1 + 2` rovna 3 a hodnota `1 + year` je $2000 = 1 + t(\texttt{year})$.

2 Vypočítané atributy

Již víme, že obecný SELECT výraz

```
( SELECT DISTINCT y1 AS z1, ..., yn AS zn
  FROM   expr1 AS relation1, ..., exprm AS relationm
  WHERE  condition )
```

se skládá z klauzule SELECT:

```
SELECT DISTINCT y1 AS z1, ..., yn AS zn
```

klauzule FROM:

```
FROM expr1 AS relation1, ..., exprm AS relationm
```

a klauzule WHERE:

```
WHERE condition
```

Vyhodnocení SELECT výrazu můžeme rozbít do fází příslušícím klauzulím:

1. Klauzule FROM získá hodnoty výrazů ***expr1***, ..., ***expr***m, provede přejmenování, které přidá prefixy ***relation1***, ..., ***relation***m, a spočítá kartézský součin.
2. Klauzule WHERE provede dále restrikci podle podmínky ***condition***.
3. Klauzule SELECT spočítá projekci na $\{y_1, \ldots, y_n\}$ a provede přejmenování každého atributu y_i na z_i.

Rozšíříme si SELECT klauzuli o možnost spočítat hodnoty atributů skalárními výrazy:

```
SELECT DISTINCT scalar_expr1 AS z1, ..., scalar_exprn AS zn
```

kde ***scalar_expr***1, ..., ***scalar_expr***n jsou skalární výrazy nad S, kde S je sjednocením typů relačních výrazů ***expr1***, ..., ***expr***m s prefixy.

Označme si $S' = \{z_1, \ldots, z_n\}$. Řekneme, že n-tice t' nad S' je **vyhodnocením SELECT klauzule** v n-tici t nad S,

- jestliže $t'(z_i)$ je hodnota skalárního výrazu ***scalar_expr***i v n-tici t pro každé $1 \leq i \leq n$.

Například n-tice

$$\{\langle \texttt{title}, \texttt{'The Matrix'}\rangle, \langle \texttt{next\_year}, 2000\rangle\}$$

je výsledkem vyhodnocení SELECT klauzule

```
SELECT DISTINCT title AS title, year + 1 AS next_year
```

v n-tici

$$\{\langle \texttt{title}, \texttt{'The Matrix'}\rangle, \langle \texttt{year}, 1999\rangle, \langle \texttt{length}, 136\rangle\}.$$

SELECT klauzule očekává relaci R nad S a vrátí relaci R' nad S' splňující následující podmínku. Pro každou n-tici t' nad S' platí, že

- $t' \in R'$, právě když existuje $t \in R$ taková, že t' je vyhodnocením SELECT klauzule v t.

Například klauzule `SELECT DISTINCT title AS title, year + 1 AS next_year` pro relaci:

```
      title       | year | length
------------------+------+--------
 The Matrix       | 1999 |    136
 The Avengers     | 2012 |    143
 The Avengers     | 1998 |     89
 A Space Odyssey | 1968 |    149
```

vrátí:

```
      title      | next_year
-----------------+-----------
 The Matrix      |      2000
 The Avengers    |      2013
 The Avengers    |      1999
 A Space Odyssey |      1969
```

3 Podmínky

Podmínky můžeme chápat jako jisté skalární výrazy. Skalární typ `boolean` dává jméno dvěma (pravdivostním) hodnotám `t` (reprezentuje pravdu) a `f` (reprezentuje nepravdu). Skalární výraz, jehož hodnota je typu `boolean`, se nazývá **podmínka**. Podmínka `TRUE` má vždy hodnotu `t` a podmínka `FALSE` vždy hodnotu `f`.

Podmínky lze vytvořit pomocí **komparátorů**. To jsou binární operátory, které očekávají dvě hodnoty stejného typu a vrací pravdivostní hodnotu. Mezi komparátory patří: `<` (menší než), `>` (větší než), `<=` (menší nebo rovno než), `>=` (větší nebo rovno než), `=` (rovno), `<>` (nerovno). Například hodnota podmínky `year = 1999` nad $\{\texttt{title}, \texttt{year}, \texttt{length}\}$ vzhledem k n-tici:

$$\{\langle \texttt{title}, \texttt{The Matrix}\rangle, \langle \texttt{year}, 1999\rangle, \langle \texttt{length}, 136\rangle\}$$

je rovna `t`. Uspořádání hodnot typu `integer` splývá s přirozeným uspořádáním čísel. Uspořádání hodnot typu `text` je lexikografické.

Pokud je hodnota podmínky ***`condition`*** vzhledem k n-tici t rovna hodnotě `t`, říkáme, že n-tice t **splňuje** podmínku ***`condition`***. Například n-tice:

$$\{\langle \texttt{title}, \texttt{The Matrix}\rangle, \langle \texttt{year}, 1999\rangle, \langle \texttt{length}, 136\rangle\}$$

splňuje podmínku `year < 2000`.

Následující tabulka udává přehled běžně používaných operátorů seřazených podle priority.

operátor	arita	asociativita	popis
`+`, `-`	1	zprava	unární plus a minus
`^`	2	zleva	umocňování
`*`, `/`, `%`	2	zleva	součin, podíl, modulo
`+`, `-`	2	zleva	součet, rozdíl
`<`, `>`, `=`, `<=`, `>=`, `<>`	2		komparátory
`NOT`	1	zprava	logická negace
`AND`	2	zleva	logická konjunkce
`OR`	2	zleva	logická disjunkce

Prioritu a asociativitu operátoru můžeme využít k vynechávání závorek.

4 Řazení n-tic

Rozšíříme si SELECT výraz o ORDER BY klauzuli:

```
( SELECT   DISTINCT
           attributes
  FROM     relations
  WHERE    condition
  ORDER BY ordering )
```

ORDER BY klauzule se vyhodnocuje po SELECT klauzuli. Předpokládejme, že klauzule dostala na vstupu relaci R nad S. Část ***ordering*** se skládá z prvků tvaru:

y `ASC`

nebo:

y `DESC`

oddělených čárkou, kde $y \in S$. Samotné y je zkratkou za y `ASC`. Obecně tedy má část ***ordering*** tvar:

y_1 ***direction1*** `,` `...` `,` y_n ***direction***n

Pro každé $1 \leq k \leq n$ si zavedeme binární relaci $<_k$ na R následovně. Pro dvě n-tice $t_1, t_2 \in R$ položíme $t_1 <_k t_2$, jestliže se t_1 a t_2 shodují na $\{y_1, \ldots, y_{k-1}\}$ a

1. $t_1(y_k) < t_2(y_k)$, jestliže ***direction***k je `ASC`,
2. a $t_2(y_k) < t_1(y_k)$, jestliže ***direction***k je `DESC`.

Poznamenejme, že symbol $<$ v právě uvedených podmínkách označuje ostrou nerovnost v doménách atributů.

Zavedeme si binární relaci $<$ na R následovně. Pro dvě n-tice $t_1, t_2 \in R$ položíme $t_1 < t_2$, jestliže existuje číslo $1 \leq k \leq n$ takové, že $t_1 <_k t_2$.

Věta 1. *Relace $<$ je tranzitivní a asymetrická.*

Důkaz:

1. Nejprve dokážeme tranzitivitu. Vezměme $t_1, t_2, t_3 \in R$ takové, že $t_1 < t_2$ a $t_2 < t_3$. Pak z definice relace $<$ existují $1 \leq k_1, k_2 \leq n$ taková, že $t_1 <_{k_1} t_2$ a $t_2 <_{k_2} t_3$. Položme k_3 rovno minimu z k_1 a k_2. Označme $S_3 = \{y_1, \ldots, y_{k_3-1}\}$. Jistě n-tice t_1 a t_2 se shodují na S_3 a také n-tice t_2 a t_3 se shodují na S_3. Tedy t_1 a t_3 se shodují na S_3. Omezme se na situaci, kde ***direction***k_3 je `ASC` (pro `DESC` bychom postupovali dál analogicky) a $k_3 = k_1$ (opět pro $k_3 = k_2$ bychom postupovali dál analogicky). Dostáváme, že $t_1(y_{k_3}) < t_2(y_{k_3}) \leq t_3(y_{k_3})$ a tedy $t_1(y_{k_3}) < t_3(y_{k_3})$. Dohromady jsem ověřili, že $t_1 <_{k_3} t_3$ a proto $t_1 < t_3$.

2. Nyní dokážeme asymetrii. Vezměme $t_1, t_2 \in R$ takové, že $t_1 < t_2$ a $t_2 < t_1$. Musí existovat $1 \leq k_1, k_2 \leq n$ taková, že $t_1 <_{k_1} t_2$ a $t_2 <_{k_2} t_1$. Předpokládejme, že $k_1 = k_2$ a omezme se na situaci, kde ***direction***k je `ASC`, pak $t_1(y_{k_1}) < t_2(y_{k_1})$ a $t_2(y_{k_1}) < t_1(y_{k_1})$, což je spor. Předpokládejme, že $k_1 < k_2$ a omezme se na ***direction***k_1 rovno `ASC`, pak t_1 a t_2 se shodují na $S_2 = \{y_1, \ldots, y_{k_2-1}\}$, ale $k_1 \in S_2$. Tedy $t_1(y_{k_1}) < t_2(y_{1_2})$ a $t_1(y_{k_1}) = t_2(y_{k_1})$, což je také spor. Dohromady dostáváme, že pokud $t_1 < t_2$, pak $t_2 \not< t_1$. Tedy $<$ je asymetrická relace.

□

Reflexivní uzávěr relace $<$ si označíme $\leq$. Tedy $\leq$ je sjednocením $<$ a $\{\langle t, t\rangle \mid t \in R\}$. Relace $\leq$ je (částečné) uspořádání na množině R.

Klauzule vrátí posloupnost $(t_i)_1^m$ n-tic relace R takovou, že každá n-tice $t_i \in R$ se v posloupnosti vyskytuje právě jednou a dále pro každé $1 \leq i, j \leq m$ platí, že $t_i \leq t_j$ implikuje $i \leq j$.

Řekneme, že $\leq$ **úplně řadí** R, pokud $\leq$ je úplné (lineární) uspořádání množiny R. Tedy když pro každé dvě n-tice $t_1, t_2 \in R$ platí, že $t_1 \leq t_2$ nebo $t_2 \leq t_1$.

Posloupnost n-tic nad S můžeme opět znázornit tabulkou. Na rozdíl od relace je však pořadí řádků v tabulce významné, protože určuje pořadí n-tic v posloupnosti. Například:

```
# SELECT * FROM movie;

    director     |              title               | year
-----------------+----------------------------------+------
 Stanley Kubrick | Barry Lyndon                     | 1975
 Terry Gilliam   | Monty Python and the Holy Grail  | 1975
 Milos Forman    | One Flew Over the Cuckoo's Nest  | 1975
 Stanley Kubrick | Lolita                           | 1962
 Robert Mulligan | To Kill a Mockingbird            | 1962
 Stanley Kubrick | 2001: A Space Odyssey            | 1968
 Carol Reed      | Oliver!                          | 1968
(7 rows)


# SELECT   *
  FROM     movie
  ORDER BY year ASC;

    director     |              title               | year
-----------------+----------------------------------+------
 Stanley Kubrick | Lolita                           | 1962
 Robert Mulligan | To Kill a Mockingbird            | 1962
 Carol Reed      | Oliver!                          | 1968
 Stanley Kubrick | 2001: A Space Odyssey            | 1968
```

```
 Milos Forman    | One Flew Over the Cuckoo's Nest | 1975
 Stanley Kubrick | Barry Lyndon                    | 1975
 Terry Gilliam   | Monty Python and the Holy Grail | 1975
(7 rows)


# SELECT   *
  FROM     movie
  ORDER BY year DESC;

    director     |              title              | year
-----------------+---------------------------------+------
 Milos Forman    | One Flew Over the Cuckoo's Nest | 1975
 Stanley Kubrick | Barry Lyndon                    | 1975
 Terry Gilliam   | Monty Python and the Holy Grail | 1975
 Stanley Kubrick | 2001: A Space Odyssey           | 1968
 Carol Reed      | Oliver!                         | 1968
 Stanley Kubrick | Lolita                          | 1962
 Robert Mulligan | To Kill a Mockingbird           | 1962
(7 rows)


# SELECT   *
  FROM     movie
  ORDER BY director;

    director     |              title              | year
-----------------+---------------------------------+------
 Carol Reed      | Oliver!                         | 1968
 Milos Forman    | One Flew Over the Cuckoo's Nest | 1975
 Robert Mulligan | To Kill a Mockingbird           | 1962
 Stanley Kubrick | Lolita                          | 1962
 Stanley Kubrick | Barry Lyndon                    | 1975
 Stanley Kubrick | 2001: A Space Odyssey           | 1968
 Terry Gilliam   | Monty Python and the Holy Grail | 1975
(7 rows)

# SELECT   *
  FROM     movie
  ORDER BY director, year;

    director     |              title              | year
-----------------+---------------------------------+------
 Carol Reed      | Oliver!                         | 1968
 Milos Forman    | One Flew Over the Cuckoo's Nest | 1975
 Robert Mulligan | To Kill a Mockingbird           | 1962
```

```
 Stanley Kubrick | Lolita                          | 1962
 Stanley Kubrick | 2001: A Space Odyssey           | 1968
 Stanley Kubrick | Barry Lyndon                    | 1975
 Terry Gilliam   | Monty Python and the Holy Grail | 1975
(7 rows)
```

5 Část posloupnosti

LIMIT klauzule se píše za ORDER BY klauzuli a má tvar:

```
LIMIT k1
OFFSET k2
```

kde k_1 a k_2 jsou přirozená čísla. Vyhodnocuje se po ORDER BY klauzuli. Klauzule pro posloupnost $(t_i)_1^m$ vrátí posloupnost $(t_i)_{k_2+1}^{k_1+k_2}$. Například pro posloupnost n-tic:

```
    director     |              title               | year
-----------------+----------------------------------+------
 Carol Reed      | Oliver!                          | 1968
 Milos Forman    | One Flew Over the Cuckoo's Nest  | 1975
 Robert Mulligan | To Kill a Mockingbird            | 1962
 Stanley Kubrick | Lolita                           | 1962
 Stanley Kubrick | 2001: A Space Odyssey            | 1968
 Stanley Kubrick | Barry Lyndon                     | 1975
 Terry Gilliam   | Monty Python and the Holy Grail  | 1975
(7 rows)
```

klauzule LIMIT:

```
LIMIT    2
OFFSET   2
```

vrátí:

```
    director     |         title          | year
-----------------+------------------------+------
 Robert Mulligan | To Kill a Mockingbird  | 1962
 Stanley Kubrick | Lolita                 | 1962
(2 rows)
```

Klauzuli LIMIT je vhodné používat, jen pokud je přítomná ORDER BY klauzule, která vždy řadí úplně. V opačném se může měnit hodnota SELECT výrazu více než jen pořadím n-tic.

Pokud místo k_1 uvedeme `ALL`, vrátí se posloupnost $(t_i)_{k_2+1}^m$. Například:

```
# SELECT   *
  FROM     movie
  ORDER BY director, year
  LIMIT ALL
  OFFSET   2;

    director     |              title               | year
-----------------+----------------------------------+------
 Robert Mulligan | To Kill a Mockingbird            | 1962
 Stanley Kubrick | Lolita                           | 1962
 Stanley Kubrick | 2001: A Space Odyssey            | 1968
 Stanley Kubrick | Barry Lyndon                     | 1975
 Terry Gilliam   | Monty Python and the Holy Grail | 1975
(5 rows)
```

Vynechání klauzule OFFSET:

```
LIMIT k1
```

je zkratkou za:

```
LIMIT k1
OFFSET 0
```

Například:

```
 # SELECT   *
  FROM     movie
  ORDER BY director, year
  LIMIT    2;

   director   |              title               | year
--------------+----------------------------------+------
 Carol Reed   | Oliver!                          | 1968
 Milos Forman | One Flew Over the Cuckoo's Nest | 1975
(2 rows)
```

6 Agregace

Skalární výraz:

```
function(expr1, ..., exprn)
```

kde *function* je název funkce a *expr1*, ..., *exprn* jsou skalární výrazy typu, který funkce vyžaduje, nazveme **volání funkce**.

Hodnotu volání funkce získáme aplikací funkce *function* na argumenty rovny hodnotám skalárních výrazů *expr1*, ..., *exprn*.

Některé skalární výrazy nazýváme **agregační**. Pokud v SELECT klauzuli použijeme agregační výraz, musí být být všechny výrazy v klauzuli agregační a říkáme, že SELECT klauzule je **agregační**. Některé funkce jsou **agregační**. Zavoláním agregační funkce získáme agregační výraz. Například funkce `sum` je agregační a proto skalární výraz `sum(duration)` je agregační.

Agregační funkce na vstupu očekává posloupnost $(d_i)_1^n$ hodnot a vrací hodnotu jedinou. Například agregační funkce `sum` pro konečnou posloupnost čísel vrací jejich součet. Dále předpokládáme, že všechny posloupnosti jsou konečné.

Vyhodnocení volání agregační funkce probíhá vzhledem k relaci R, kterou obdrží klauzule SELECT. Agregační funkce přijímají vždy jediný argument. Předpokládejme volání agregační funkce:

function`(`*expr*`)`

Nejprve se uspořádají n-tice v R do libovolné posloupnosti $(t_i)_1^m$. Poté se postupně pro každé t_i vyhodnotí skalární výraz *expr*. Tím získáme posloupnost hodnot $(d_i)_i^m$. Nakonec se na $(d_i)_i^m$ aplikuje funkce *function*. Návratová hodnota funkce bude hodnotou výrazu volání agregační funkce.

Vezměme SELECT klauzuli s agregačními výrazy:

`SELECT` *expr1* `AS` z_1`, ...,` *exprn* `AS` z_n

Vyhodnocením klauzule pro relaci R získáme relaci $\{t\}$ nad $S = \{z_1, \ldots, z_n\}$, kde t je n-tice nad S taková, že pro každé $1 \leq i \leq n$ je $t(z_i)$ hodnota agregačního výrazu *expri* v R.

Například pro relační proměnnou:

```
# TABLE movie;

    director     | duration |              title               | year
-----------------+----------+----------------------------------+------
 Stanley Kubrick |      184 | Barry Lyndon                     | 1975
 Terry Gilliam   |       91 | Monty Python and the Holy Grail | 1975
 Milos Forman    |      133 | One Flew Over the Cuckoo's Nest | 1975
 Stanley Kubrick |      152 | Lolita                           | 1962
 Robert Mulligan |      129 | To Kill a Mockingbird            | 1962
 Stanley Kubrick |      161 | 2001: A Space Odyssey            | 1968
 Carol Reed      |      153 | Oliver!                          | 1968
(7 rows)
```

Dostáváme:

```
# SELECT sum(duration) AS total_duration
  FROM   movie;

 total_duration
----------------
           1003
(1 row)
```

Agregační funkce `min` a `max` počítají minimum a maximum ze zadaných čísel. Operátory z agregačních výrazů vytvářejí opět agregační výrazy. Například:

```
# SELECT max(year) AS oldes_year,
         min(year) AS newest_year,
         max(year) - min(year) AS year_range
  FROM   movie;

 oldes_year | newest_year | year_range
------------+-------------+------------
       1975 |        1962 |         13
(1 row)
```

Hodnotou agregačního výrazu `count(*)` je počet n-tic relace R. Například:

```
# SELECT count(*) AS count FROM movie;

 count
-------
     7
(1 row)
```

7 Seskupování

Klauzule GROUP BY se vkládá mezi WHERE a ORDER BY klauzuli a má tvar:

`GROUP BY` y_1`, ...,` y_n

kde $S_G = \{y_1, ..., y_n\} \subseteq S$ a S je relační schéma. Klauzule se vyhodnocuje po WHERE klauzuli před SELECT klauzulí. Je-li klauzule GROUP BY přítomna v SELECT výrazu, pak SELECT klauzule musí být agregační. Atributy y_1, ..., y_n se považují za agregační výrazy. Jejich vyhodnocení popíšeme dále.

Vyhodnocování SELECT výrazu se mění následovně. Předpokládejme, že GROUP BY klauzule obdržela relaci R nad S. Zavedeme ekvivalenci $\equiv$ na R definovanou tak, že položíme $t_1 \equiv t_2$, jestliže t_1 a t_2 se shodují na S_G. Třídu ekvivalence podle $\equiv$ nazýváme **skupinou**. Nechť $R_1, \ldots, R_m$ jsou všechny skupiny. Dále vyhodnotíme SELECT klauzuli pro každou skupinu R_i. Získáme relace $R'_1, \ldots, R'_m$. Každá z relací $R'_1, \ldots, R'_m$ obsahuje právě jednu n-tici. Hodnotou SELECT klauzule bude sjednocení $R'_1 \cup \cdots \cup R'_m$. Dál se vyhodnocování řídí běžnými pravidly.

Agregační výraz $y_i \in S_R$ se pro skupinu R_j vyhodnotí na $t(y_i)$, kde $t \in R_j$ je libovolné.

Vezměme například relaci:

```
# TABLE movie;

              title                | year | country | duration
-----------------------------------+------+---------+----------
 2001: A Space Odyssey             | 1968 | UK      |      161
 Barry Lyndon                      | 1975 | UK      |      184
 The Man Who Shot Liberty Valance  | 1962 | USA     |      113
 The Return of the Pink Panther    | 1975 | UK      |      113
 One Flew Over the Cuckoo's Nest   | 1975 | USA     |      133
 To Kill a Mockingbird             | 1962 | USA     |      129
 Monty Python and the Holy Grail   | 1975 | UK      |       91
 Lolita                            | 1962 | UK      |      152
 Jaws                              | 1975 | USA     |      130
(9 rows)
```

a vyhodnoťme výraz:

```
SELECT   year,
         country,
         count(*) AS count
FROM     movie
GROUP BY year, country
```

relaci obdrží GROUP BY klauzule:

```
GROUP BY year, country
```

která pro $S_G = \{\texttt{year}, \texttt{country}\}$ vytvoří pět skupin $R_1, \ldots, R_5$:

```
 title  | year | country | duration
--------+------+---------+----------
 Lolita | 1962 | UK      |      152
(1 row)

         title          | year | country | duration
```

```
----------------------+------+---------+----------
 2001: A Space Odyssey | 1968 | UK      |      161
(1 row)

              title               | year | country | duration
----------------------------------+------+---------+----------
 One Flew Over the Cuckoo's Nest | 1975 | USA     |      133
 Jaws                            | 1975 | USA     |      130
(2 rows)

              title               | year | country | duration
----------------------------------+------+---------+----------
 Barry Lyndon                    | 1975 | UK      |      184
 The Return of the Pink Panther  | 1975 | UK      |      113
 Monty Python and the Holy Grail | 1975 | UK      |       91
(3 rows)

              title                | year | country | duration
-----------------------------------+------+---------+----------
 The Man Who Shot Liberty Valance | 1962 | USA     |      113
 To Kill a Mockingbird            | 1962 | USA     |      129
(2 rows)
```

Pro každou skupinu vyhodnotíme SELECT klauzuli:

```
SELECT   year,
         country,
         count(*) AS count
```

a získáme relace $R'_1, \ldots, R'_5$:

```
 year | country | count
------+---------+-------
 1962 | UK      |     1
(1 row)

 year | country | count
------+---------+-------
 1968 | UK      |     1
(1 row)

 year | country | count
------+---------+-------
 1975 | USA     |     2
(1 row)

 year | country | count
```

```
------+---------+-------
 1975 | UK      |     3
(1 row)

 year | country | count
------+---------+-------
 1962 | USA     |     2
(1 row)
```

Výstup klauzule SELECT bude sjednocení $R'_1 \cup \cdots \cup R'_5$:

```
 year | country | count
------+---------+-------
 1962 | UK      |     1
 1968 | UK      |     1
 1975 | USA     |     2
 1975 | UK      |     3
 1962 | USA     |     2
(5 rows)
```

Obdrželi jsme i hodnotu agregačního SELECT výrazu.

8 Filtrování skupin

Klauzule HAVING se vkládá mezi klauzule GROUP BY a ORDER BY. Může být použita pouze v kombinaci s GROUP BY klauzulí. Tvar klauzule je následující:

```
HAVING condition
```

kde *condition* je podmínka, která je současně agregačním výrazem.

Klauzule HAVING se vyhodnocuje po klauzuli GROUP BY tak, že pro skupiny $R_1, \ldots, R_n$ vrátí jen ty, které splňují podmínku *condition*.

Následující výraz například odstraní skupiny, které mají jediný prvek:

```
# SELECT   year,
           country,
           count(*) AS count
  FROM     movies
  GROUP BY year, country
  HAVING   count(*) > 1;

 year | country | count
------+---------+-------
 1975 | USA     |     2
 1975 | UK      |     3
```

```
 1962 | USA      |     2
(3 rows)
```

9 Všechny klauzule SELECT výrazu

SELECT výraz obsahující všechny klauzule, které jsme potkali, má tvar:

```
( SELECT   DISTINCT attributes
  FROM     relations
  WHERE    condition
  GROUP BY attributes
  HAVING   condition
  ORDER BY ordering
  LIMIT    count
  OFFSET   start )
```

Klauzule se vyhodnocují v tomto pořadí:

1. FROM (získání vstupní relace),
2. WHERE (restrikce),
3. GROUP BY (seskupování),
4. HAVING (filtrování skupin),
5. SELECT (výpočet výstupní relace),
6. ORDER BY (seřazení řádků),
7. LIMIT (výběr části posloupnosti).

Otázky a úkoly na cvičení

1. Uvažujme proměnnou `dead_actor` danou vlastností „Herec `actor` se narodil roku `born` a umřel roku `die`.“ Napište SELECT výraz daný vlastností „Herec `actor` se dožil `year` let.“
2. Určete hodnotu výrazu z předchozího úkolu pro hodnotu proměnné `dead_actor` rovnou:

actor	born	die
Alain Delon	1935	2024
Audrey Hepburn	1929	1993
Marlon Brando	1924	2004

3. Uvažujme relační proměnnou `film_studio` danou vlastností „Filmové studio `studio` bylo založeno roku `founded`.“ s hodnotou:

`studio`	`founded`
Universal	1912
Paramount Pictures	1912
Warner Bros. Pictures	1923
Walt Disney Studios	1923
Sony Pictures	1987

a relační proměnnou `movie_brand` danou vlastností „Filmovou značku `brand` vlastní filmové studio `studio`.“ s hodnotou:

`brand`	`studio`
Spider-Man	Sony Pictures
Men in Black	Sony Pictures
Harry Potter	Warner Bros. Pictures
Batman	Warner Bros. Pictures
The Lord of the Rings	Warner Bros. Pictures
Transformers	Paramount Pictures
Jurassic Park	Universal

   (a) Napište SELECT výraz, jehož hodnota bude posloupnost všech n-tic $(t_i)_1^n$ relace dané vlastností „`brand` je filmová značka.“, kde pro každé $1 \leq i, j \leq n$ platí, že jestliže značku t_i(`brand`) vlastní starší filmové studio než značku t_j(`brand`), pak $i < j$.

   (b) Určete hodnotu předchozího výrazu.

   (c) Je řazení SELECT výrazem úplné? Pokud ne, změňte SELECT výraz, tak aby úplné bylo.

   (d) Určete hodnotu předchozího výrazu.

4. Vezměme si proměnné z třetího úkolu.

   (a) Napište relační výraz s charakteristickou vlastností „Studio Sony Pictures vlastní `count` filmových značek“ a určete jeho hodnotu.

   (b) Napište relační výraz s charakteristickou vlastností „Studio `studio` vlastní `count` filmových značek“ a určete jeho hodnotu.

   (c) Napište relační výraz s charakteristickou vlastností „Studio `studio` vlastní alespoň dvě filmové značky“ a určete jeho hodnotu.

5. Dokažte, že binární relace $\leq$ na relaci R definovaná ve čtvrté části je opravdu částečné uspořádání na R.