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

Databáze ◇ poznámky k přednášce

7. Další klauzule SELECT výrazu

verze z 2. listopadu 2023

1 Skalární výraz

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

Předpokládáme, že pro každý atribut $y \in \mathcal{Y}$ máme skalární typ $\texttt{scalar\_type}_y$ takový, že množina hodnot typu $\texttt{scalar\_type}_y$ je rovna doméně $\mathcal{D}_y$ 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 R. Každý atribut $y \in R$ je skalární výraz nad R typu $\texttt{scalar\_type}_y$. Například uvažujme relační schéma $R = \{\texttt{title}, \texttt{year}, \texttt{length}\}$, pak `title` je skalární výraz nad R typu `text`.

Hodnota skalárního výrazu nad R se určuje vzhledem k n-tici r nad R. Hodnota skalárního výrazu y nad R vzhledem k r je $r(y)$. Například hodnota skalárního výrazu `title` nad $R = \{\texttt{title}, \texttt{year}, \texttt{length}\}$ vzhledem k n-tici $r = \{\langle \texttt{title}, \texttt{The Matrix}\rangle, \langle \texttt{year}, 1999\rangle, \langle \texttt{length}, 136\rangle\}$ nad R je $r(\texttt{title}) =$ '`The Matrix`'.

Libovolná skalární hodnota d typu ***`scalar_type`*** je skalárním výrazem nad $\emptyset$, 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 R, pak

```
( operator operand )
```

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

Představíme si zatím jediný 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 $R = \{\texttt{title}, \texttt{year}, \texttt{length}\}$ ale `-'The Matrix'` nebo `-title` není skalární výraz nad R.

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

```
( operand1 operator operand2 )
```

je skalární výraz nad R.

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 $r = \{\langle \texttt{title}, \texttt{The Matrix}\rangle, \langle \texttt{year}, 1999\rangle, \langle \texttt{length}, 136\rangle\}$ je hodnota skalárního výraz `1 + 2` rovna 3 a hodnota `1 + year` je $2000 = 1 + r(\texttt{year})$.

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
```

(klauzuli DISTINCT nyní bereme jako součást klauzule SELECT) klauzule

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

a klauzule WHERE:

```
WHERE  condition
```

Připomeňme si, že můžeme vyhodnocení SELECT výrazu rozbít do fází příslušícím KLAUZULÍM:

1. Klauzule FROM získá hodnoty výrazů *expr1*, ..., *exprm*, provede přejmenování, které přidá prefixy prefixy *relation1*, ..., *relationm*, a spočítá kartézský součin.
2. Klauzule WHERE provede dále restrikci podle podmínky *condition*.

3. Na závěr klauzule SELECT spočítá projekci na $\{y_1, \ldots, y_n\}$ a na závěr provede přejmenování každého atributu y_i na z_i.

Více podrobností naleznete v šesté části s názvem „SELECT jako relační výraz“ poznámek k páté přednášce.

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

```
SELECT DISTINCT scalar_expr_1 AS z_1, ..., scalar_expr_n AS z_n
```

kde *scalar_expr*$_1$, ..., *scalar_expr*$_n$ jsou skalární výrazy nad R.

Klauzule pak očekává relaci $\mathcal{D}$ nad R a vrátí relaci $\mathcal{D}'$ nad $S = \{z_1, \ldots, z_n\}$, pro kterou platí, že $s \in \mathcal{D}'$, právě když existuje $r \in \mathcal{D}$ taková, že $s(z_i)$ je hodnota skalárního výrazu *scalar_expr*$_i$ v n-tici r pro každé $1 \leq i \leq n$.

Například pokud 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

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 $\{$`title`, `year`, `length`$\}$ vzhledem k n-tici $\{\langle$`title`, `The Matrix`$\rangle, \langle$`year`, 1999$\rangle, \langle$`length`, 136$\rangle\}$ rovna `t`.

Pokud je hodnota podmínky ***condition*** vzhledem k n-tici r rovna hodnotě `t`, říkáme, že n-tice r *splňuje* podmínku ***condition***. Tedy n-tice $\{\langle$`title`, `The Matrix`$\rangle, \langle$`year`, 1999$\rangle, \langle$`l` splňuje podmínku `year = 1999`.

4 Uspořádání

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 tabulku $\mathcal{D}$ nad R. Část ***ordering*** se skládá z výrazů tvaru:

y `ASC`

a

y `DESC`

oddělených čárkou, kde $y \in R$. Můžeme zadat pouze y, pak výraz znamená y `ASC`.

Obecně tedy má část ***ordering*** tvar:

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

Zavedeme si binární relaci $\leq$ na $\mathcal{D}$ následovně.

Pro dvě n-tice $r_1, r_2 \in \mathcal{D}$ položíme $r_1 < r_2$, pokud existuje číslo $1 \leq k \leq m$ takové, že r_1 a r_2 se shodují na $\{y_1, \ldots, y_{k-1}\}$ a $r_1(y_k) < r_2(y_k)$ v případě, že ***direction***k je `ASC`, a $r_2(y_k) < r_1(y_k)$ v případě, že ***direction***k je `DESC`.

Klauzule vrátí posloupnost $(r_i)_1^m$ n-tic relace $\mathcal{D}$ takovou, že každá n-tice $r_i \in \mathcal{D}$ se v posloupnosti vyskytuje právě jednou a dále pro každé $1 \leq i, j \leq n$ platí, že $r_i < r_j$ implikuje $i < j$.

Řekneme, že $<$ *úplně* řadí $\mathcal{D}$, pokud pro každé dvě různé n-tice $r_1, r_2 \in \mathcal{D}$ platí, že $r_1 < r_2$ nebo $r_2 < r_1$.

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

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

Následující klauzule se píše za ORDER BY klauzuli.

```
LIMIT k1
OFFSET k2
```

kde k_1 a k_2 jsou přirozená čísla.

Vyhodnocuje se po ORDER BY klauzuli. Klauzule pro posloupnost $(r_i)_1^m$ vrátí posloupnost $(r_i)_{k_2+1}^{k_1+k_2}$.

Například pro posloupnost n-tic:

```
# 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)
```

klauzule limit:

```
LIMIT    2
OFFSET   2
```

vrátí:

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

    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 $(r_i)_{k_2+1}^{n}$. 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)
```

Tvar klauzule:

```
LIMIT k1
OFFSET 0
```

můžeme zjednodušit vynecháním části OFFSET:

```
LIMIT k1
```

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

Volání funkce. Hodnotu skalární výrazu

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

získáme zavoláním funkce ***function*** s 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 $\mathcal{D}$, kterou obdrží klauzule SELECT. Agregační funkce přijímají vždy jediný argument. Předpokládejme volání agregační funkce:

***function*(*expr*)**

Nejprve se uspořádají n-tice v $\mathcal{D}$ do libovolné posloupnosti $(r_i)_1^m$. Poté se postupně pro každé r_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, ..., ***expr***n AS z_n

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

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 $\mathcal{D}$. 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 $S = \{y_1, ..., y_n\} \subseteq R$ a R 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 $\mathcal{D}$ nad R. Zavedeme ekvivalenci $\equiv$ na $\mathcal{D}$ definovanou tak, že položíme $r_1 \equiv r_2$, jestliže r_1 a r_2 se shodují na S. Třídu ekvivalence podle $\equiv$ nazýváme *skupinou*. Nechť $\mathcal{D}_1, \ldots, \mathcal{D}_m$ jsou všechny skupiny. Dále vyhodnotíme SELECT klauzuli pro každou skupinu $\mathcal{D}_i$. Získáme relace $\mathcal{D}'_1, \ldots, \mathcal{D}'_m$. (Každá z relací $\mathcal{D}'_1, \ldots, \mathcal{D}'_m$ obsahuje právě jednu n-tici.) Hodnotou SELECT klauzule bude sjednocení $\mathcal{D}'_1 \cup \cdots \cup \mathcal{D}'_m$. Dál se vyhodnocování řídí běžnými pravidly.

Agregační výraz $y_i \in S$ se pro skupinu $\mathcal{D}$ vyhodnotí na $r(y_i)$, kde $r \in \mathcal{D}$ 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 položme dotaz:

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

relaci obdrží GROUP BY klauzule:

```
GROUP BY year, country
```

která pro $S = \{\texttt{year}, \texttt{country}\}$ vytvoří pět skupin $\mathcal{D}_1, \ldots, \mathcal{D}_5$:

```
 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 $\mathcal{D}'_1, \ldots, \mathcal{D}'_5$:

```
 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 $\mathcal{D}'_1 \cup \cdots \cup \mathcal{D}'_5$:

```
 year | country | count
```

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

Obdrželi jsme i hodnotu SELECT výrazu.

8 Filtrování skupin

Klauzule HAVING se vkládá mezi 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 $\mathcal{D}_1, \ldots, \mathcal{D}_n$ 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 a DISTINCT (výpočet výstupní relace)
6. ORDER BY (seřazení řádků)
7. LIMIT (a OFFSET) (výběr části posloupnosti)