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

4. Relační algebra

verze z 15. října 2024

1 Relační algebra

Relační algebra je tvořena relačními operacemi:

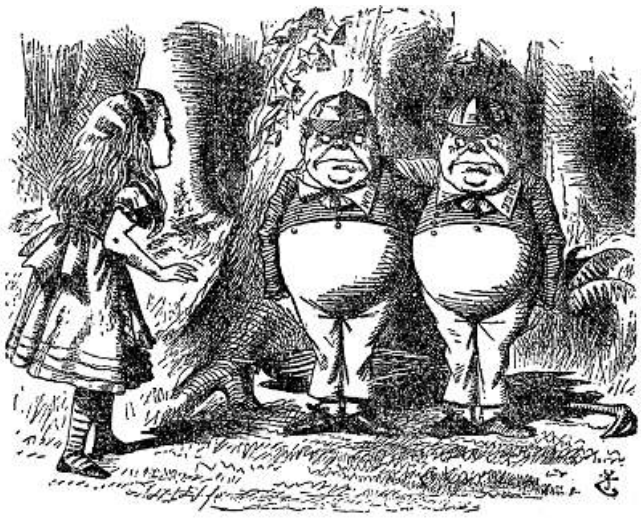
1. sjednocení ($\cup$),
2. průnik ($\cap$),
3. rozdíl ($-$),
4. restrikce (σ),
5. projekce (π),
6. spojení ($\bowtie$),
7. přejmenování atributů (ρ).

Relační model dat představil E. F. Codd v roce 1970. Navrhl relační algebru jako základ dotazovacích jazyků. V originální relační algebře chybí přejmenování atributů (atributy v relaci měly pevně dané pořadí) a navíc zde byl kartézský součin a relační dělení. Víme, že kartézský součin je speciální případ spojení. Relačnímu dělení je věnována část níže.

SQL vychází z relačního modelu, ale některé jeho principy porušuje. Například neumožňuje pracovat s relacemi s nad prázdným schématem. Jak bylo zmíněno dříve tabulky v SQL přinášení závažnější prohřešky proti relačnímu modelu (komponenta nemusí mít hodnotu, tabulka může mít duplicitní řádky a názvy sloupců nemusí být jedinečné).

Christopher J. Date a Hugh Darwen navrhli v třetím manifestu (The Third Manifesto) publikovaném v roce 1995 požadavky na jazyk respektující relační model. Jejich specifikace jazyka se nazývá D. Manifest konkrétněji popisuje jazyk Tutorial D, který specifikacím D vyhovuje. Známá implementace jazyka Tutorial D se jmenuje Rel.

Relační kalkul je dotazovací jazyk, který vychází z predikátové logiky. Dotaz formulujeme pomocí relačních symbolů (odpovídají relačním proměnným), logických spojek (disjunkce, implikace,...) a kvantifikátorů (existenční a obecný). Relační kalkul pracuje s vnitřními strukturami relací. Dělíme jej na dva typy podle



Obrázek 1: Tweedledum a Tweedledee (česky dvojčata Tydliták a Tydlitek) z knihy *Through the Looking-Glass* (česky *Za zrcadlem a co tam Alenka našla*) od Lewise Carrolla.

možných hodnot objektových proměnných. Za prvé **n-ticový relační kalkul**, kde objektové proměnné nabývají hodnot n-tic. Za druhé **doménový relační kalkul**. Zde hodnoty objektových proměnných jsou přímo prvky domén atributů. Oba relační kalkuly a relační algebra mají stejnou vyjadřovací sílu. Tím se rozumí, že libovolný dotaz v jednom jazyku jsme schopni přeformulovat do ostatních jazyků tak, že výsledky všech dotazů jsou vždy stejné.

2 Relace nad prázdným schématem

Prázdná množina atributů $\emptyset$ je také relační schéma. Existuje jen jediná n-tice $t_0 \in \text{Tupl}(\emptyset)$ nad $\emptyset$ a to prázdná množina $\emptyset$. Existují dvě relace nad schématem $\emptyset$: prázdná relace $\emptyset$ a množina obsahující pouze n-tici t_0 tedy množina $\{t_0\} = \{\emptyset\}$. První se jmenuje `DUM` a druhá `DEE`. Jména relací vychází z anglických jmen postav Tweedledum a Tweedledee (česky dvojčata Tydliták a Tydlitek) z knihy *Za zrcadlem a co tam Alenka našla* od Lewise Carrolla. Postavy jsou zachyceny na Obrázku 1. Relace `DUM` reprezentuje *nepravdu* a relace `DEE` *pravdu*.

Uvažujme například relaci R nad S a podmínku θ nad S. Chceme zjistit, zda existuje n-tice $t \in R$, která splňuje podmínku θ. Můžeme nejprve provést restrikci relace R vzhledem k θ a poté udělat projekci na prázdnou množinu atributů, tedy zjistit hodnotu výrazu $\pi_\emptyset(\sigma_\theta(R))$. Pokud výsledkem je relace `DEE`, pak odpověď je ano a pokud `DUM`, pak je odpověď ne.

V SQL neexistují relace nad prázdným schématem. Zde tedy nenajdeme ani relaci `DUM` ani relaci `DEE`. Musíme se spokojit s tím, že každá prázdná relace reprezentuje nepravdu a každá neprázdná relace reprezentuje pravdu.

3 SELECT výraz

Vezměme relační schéma $S = \{y_1, \ldots, y_n\}$ a řetězec ***relation***. Pak **schématem** S s **prefixem** ***relation*** myslíme schéma $\{$***relation***$.y_1, \ldots,$ ***relation***$.y_n\}$. Například schéma {`title`, `year`} s prefixem `movie` je schéma {`movie.title`, `movie.year`}.

Relační výraz

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

kde ***expri*** jsou relační výrazy, ***relationi*** jsou po dvou různé dočasné relační proměnné, S_i je schéma výrazu ***expri*** s prefixem ***relationi***, $S = S_1 \cup \ldots \cup S_m$, θ je podmínka nad S, $y_1, \ldots, y_n \in S$, $z_1, \ldots, z_n$ jsou po dvou různé atributy a $D_{y_j} = D_{z_j}$ pro každé $1 \leq j \leq n$, se nazývá **SELECT výraz**. Schéma SELECT výrazu je $\{z_1, \ldots, z_n\}$. SELECT výraz určuje pořadí atributů $z_1, \ldots, z_n$.

Budeme se zabývat hodnotou SELECT výrazu. Hodnoty výrazů ***expr1***, ..., ***exprm*** si postupně označíme $R_1, \ldots, R_m$. Protože relace $R_1, \ldots, R_m$ nemusí mít po dvou disjunktní schémata, provedeme přejmenování. Konkrétně R'_i si označíme relaci $\rho_{h_i}(R_i)$, kde h_i je přejmenování atributů určené vztahem $h_i(y) =$ ***relationi***$.y$.

Hodnotu SELECT výrazu můžeme určit dvěma ekvivalentními způsoby. První způsob používá relační operace a jmenuje se **procedurální sémantika SELECT výrazu**. V ní je hodnota výrazu rovna:

$$\rho_h(\pi_{S'}(\sigma_\theta(R'_1 \times \cdots \times R'_m)))$$

kde $S' = \{y_1, \ldots, y_n\}$ a h je přejmenování atributů, které splňuje $h(y_i) = z_i$ pro každé $1 \leq i \leq n$.

Druhý způsob se nazývá **deklarativní sémantika SELECT výrazu**. V ní je hodnota výrazu rovna množině:

$$\{\{\langle z_1, t(y_1)\rangle, \ldots, \langle z_n, t(y_n)\rangle\} \mid \begin{array}{l}\text{existují } t_1 \in R'_1 \text{ a } \ldots \text{ a } t_m \in R'_m \text{ tak,} \\ \text{že } t = t_1 \ldots t_m \text{ a } t \text{ splňuje } \theta\}\end{array}$$

Určíme charakteristickou vlastnost SELECT výrazu. Vezměme charakteristické vlastnosti $V_1, \ldots, V_m$ hodnot $R'_1, \ldots, R'_m$. Charakteristická vlastnost $V(t')$ SELECT výrazu je: „Existuje $t \in \mathrm{Tupl}(S)$ tak, že $V_1(t)$ a $\ldots$ a $V_m(t)$ a t splňuje θ a $t'(z_1) = t(y_1)$ a $\ldots$ a $t'(z_n) = t(y_n)$.“

Například vezměme relační proměnnou `movie` danou: „Film `title` byl vydán roku `year`.“ s hodnotou:

`title`	`year`
The Matrix	1999
Dracula	1992
Duna	1984

a relační proměnnou `movie_cast` danou: „Herec `actor_name` hrál ve filmu `movie_title`.“ s hodnotou:

`actor_name`	`movie_title`
Keano Reeves	The Matrix
Keano Reeves	Dracula
Laurence Fishburne	The Matrix
Gary Oldman	Dracula

Pak hodnotou výrazu:

```
SELECT DISTINCT m_c.actor_name AS actor_name,
                m.year AS movie_year
FROM ( TABLE movie ) AS m,
     ( TABLE movie_cast ) AS m_c
WHERE  m_c.movie_title = m.title
```

určeném procedurální nebo deklarativní sémantikou je relace:

`actor_name`	`movie_year`
Gary Oldman	1992
Keano Reeves	1999
Keano Reeves	1992
Laurence Fishburne	1999

Odvodíme charakteristickou vlastnost výrazu. Nejprve si stanovíme použité výrokové formy:

- $V_1(t)$ je „Film t(`m.title`) byl vydán roku t(`m.year`).“
- $V_2(t)$ je „Herec t(`m_c.actor_name`) hrál ve filmu t(`m_c.movie_title`).“

Pro n-tici t' nad $\{$`actor_name`, `movie_year`$\}$ je charakteristická vlastnost $V(t')$ SELECT výrazu dána následovně.

Existuje $t \in \mathrm{Tupl}(\{$`m.title`, `m.year`, `m_c.actor_name`, `m_c.movie_title`$\})$ tak, že

- film t(`m.title`) byl vydán roku t(`m.year`)
- a herec t(`m_c.actor_name`) hrál ve filmu t(`m_c.movie_title`)
- a t(`m_c.movie_title`) $= t$(`m.title`)
- a t'(`actory_name`) $= t$(`m_c.actor_name`)
- a t'(`movie_year`) $= t$(`m.year`).

Vlastnost $V(t')$ budeme dále zjednodušovat.

1. Protože schémata n-tic t' a t jsou disjunktní, můžeme proměnné vynechat.

   Existují `m.title`, `m.year`, `m_c.actor_name`,`m_c.movie_title` tak, že

   - film `m.title` byl vydán roku `m.year`
   - a herec `m_c.actor_name` hrál ve filmu `m_c.movie_title`
   - `m_c.movie_title` = `m.title`
   - a `actory_name` = `m_c.actor_name`
   - a `movie_year` = `m.year`.

2. Víme, že `m_c.movie_title` se rovná `m.title`. Můžeme nahradit `m_c.movie_title` za `m.title` a `m_c.movie_title` vypustit.

   Existují `m.title`, `m.year`, `m_c.actor_name` tak, že

   - film `m.title` byl vydán roku `m.year`
   - a herec `m_c.actor_name` hrál ve filmu `m.title`
   - a `actory_name` = `m_c.actor_name`
   - a `movie_year` = `m.year`.

3. Nyní lze vlastnosti v prvních dvou bodech spojit do jedné.

   Existují `m.title`, `m.year`, `m_c.actor_name` tak, že

   - herec `m_c.actor_name` hrál ve filmu `m.title` vydaném v roce `m.year`
   - a `actory_name` = `m_c.actor_name`
   - a `movie_year` = `m.year`.

4. Nahradíme `m.year` za `movie_year` a `m_c.actor_name` za `actor_name`.

   Existuje `m.title` tak, že

   - herec `actory_name` hrál ve filmu `m.title` vydaném v roce `movie_year`.

5. Vlastnost můžeme zapsat stručněji následovně.

   „Herec `actor_name` hrál ve filmu vydaném v roce `movie_year`.“

4 Klauzule SELECT výrazu

Příkazy a výrazy v SQL se skládají z **klauzulí** pojmenovaných podle klíčového slova, které je uvozuje.

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 y_1 AS z_1, ..., y_n AS z_n
```

klauzule FROM:

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

a klauzule WHERE:

```
WHERE condition
```

Zavedeme zkratky, které zjednoduší zápis SELECT výrazů. Neuvedená WHERE klauzule, je zkratkou za

```
WHERE TRUE
```

kde `TRUE` je vždy splněná podmínka.

Například:

```
SELECT DISTINCT m.year AS year
FROM ( TABLE movie ) AS m
```

je zkratkou za:

```
SELECT DISTINCT m.year AS year
FROM ( TABLE movie ) AS m
WHERE  TRUE
```

Část SELECT klauzule

```
y AS z
```

je zkratkou ze

```
relationi.y AS z
```

pokud existuje právě jedno i takové, že z je prvkem schématu výrazu ***expri***. Například:

```
SELECT DISTINCT year AS year
FROM ( TABLE movie ) AS m
```

je zkratkou za:

```
SELECT DISTINCT m.year AS year
FROM ( TABLE movie ) AS m
```

Dále část SELECT klauzule:

```
y
```

je zkratkou za:

```
y AS y
```

Tedy například:

```
SELECT DISTINCT year
FROM ( TABLE movie ) AS m
```

je zkratkou za:

```
SELECT DISTINCT year AS year
FROM ( TABLE movie ) AS m
```

V SELECT klauzuli je:

```
relationi.*
```

zkratkou za:

```
x1, ..., xn
```

kde $\{x_1, \ldots, x_n\}$ je schéma výrazu ***expri***. Zkratku lze použít pouze když $\{x_1, \ldots, x_n\}$ je disjunktní s každým schématem jiného výrazu ***exprj***. Pořadí atributů x_1, ..., x_n je shodné s pořadím určeným výrazem ***expri***. Předpokládejme, že `movie` je nad $\{\texttt{title}, \texttt{year}\}$. Pak například:

```
SELECT DISTINCT m.*
FROM ( TABLE movie ) AS m
```

je zkratkou za:

```
SELECT DISTINCT title, year
FROM ( TABLE movie ) AS m
```

V SELECT klauzuli je:

```
*
```

zkratkou za:

```
relation1.*, ..., relationm.*
```

Zkratku lze použít pouze když jsou schémata výrazů ***expr1***, ..., ***expr***m po dvou disjunktní. Například:

```
SELECT DISTINCT *
FROM ( TABLE movie ) AS m
```

je zkratkou za:

```
SELECT DISTINCT m.*
FROM ( TABLE movie ) AS m
```

Pokud je schéma $\{y_1, \ldots, y_n\}$ rovno sjednocení atributů ze schémat výrazů ***expr1***, ..., ***expr***m po přidaných prefixech ***relation1***, ..., ***relation***m, pak můžeme část DISTINCT vynechat. Například:

```
SELECT *
FROM ( TABLE movie ) AS m
```

je zkratkou za:

```
SELECT DISTINCT *
FROM ( TABLE movie ) AS m
```

Část FROM klauzule:

```
relation1 AS relation2
```

je zkratkou za:

```
( TABLE relation1 ) AS relation2
```

Například:

```
SELECT *
FROM movie AS m
```

je zkratkou za

```
SELECT *
FROM ( TABLE movie ) AS m
```

Relační proměnná ve FROM klauzuli:

```
relation
```

je zkratkou za:

```
relation AS relation
```

Například:

```
SELECT *
FROM movie
```

je zkratkou za

```
SELECT *
FROM movie AS movie
```

Všimněme si, že výraz:

```
SELECT *
FROM relation
```

je ekvivalentní výrazu:

```
TABLE relation
```

Výraz uvedený ve FROM klauzuli:

```
expr
```

je zkratkou za:

```
expr AS relation
```

kde *relation* je zvolená dosud nepoužitá dočasná relační proměnná. Zkratku lze použít pouze v případě, že schéma výrazu *expr* je disjunktní se schématy ostatních výrazů uvedených ve FROM klauzuli. Nejedná se o standardní SQL. Zkratku lze použít v PostgreSQL od verze 16. Například pokud `movie1` a `movie2` jsou proměnné nad $\{\texttt{title}, \texttt{year}\}$, pak:

```
SELECT DISTINCT year
FROM ( ( TABLE movie1 ) UNION ( TABLE movie2 ) )
```

je zkratkou za:

```
SELECT DISTINCT year
FROM ( ( TABLE movie1 ) UNION ( TABLE movie2 ) ) AS tmp
```

kde *tmp* je zvoleno systémem.

5 Relační dělení

Pro motivaci si vezměme relaci R_1:

person	movie
Anna	Blue Velvet
Anna	Eraserhead
Bert	Blue Velvet
Bert	The Matrix
Cyril	Blue Velvet
Cyril	Eraserhead
Cyril	The Matrix

danou vlastností: „Osoba `person` má ráda film `movie`.“ a relaci R_2:

movie
Blue Velvet
Eraserhead

danou vlastností: „Film `movie` režíroval David Lynch.“ Chceme z nich vypočítat relaci danou vlastností „Osoba `person` má ráda aspoň jeden film a všechny filmy od Davida Lynche.“ Tedy relaci R_3:

person
Anna
Cyril

Obecně vezměme dvě schémata S_1 a S_2, která jsou disjunktní. Pro relaci R_1 nad $S_1 \cup S_2$ a relaci R_2 nad S_2 se množina

$$R_1 \div R_2 = \{t_1 \in \pi_{S_1}(R_1) \mid \text{pro každé } t_2 \in R_2 \text{ platí, že } t_1 t_2 \in R_1\}$$

nazývá **podíl** R_1 a R_2. Platí, že $R_1 \div R_2$ je relací nad S_1.

Určíme si charakteristickou vlastnost relace $R_1 \div R_2$. Předpokládejme, že R_1 je dána vlastností $V_1(t_1)$ a R_2 vlastností $V_2(t_2)$. Vlastnost podílu $V_3(t_3)$ je

„Existuje $t_1 \in \text{Tupl}(R_1)$ tak, že $V_1(t_1)$ a $t_1(S_1 - S_2) = t_3$ a pro každé $t_2 \in \text{Tupl}(S_2)$ takové, že $V_2(t_2)$ platí, že $V_1(t_3 t_2)$“.

Relaci R_3 z motivačního příkladu dostaneme jako $R_1 \div R_2$. Vlastnost podílu je:

„Existuje $t_1 \in \text{Tupl}(\{\texttt{person}, \texttt{movie}\})$ tak, že osoba $t_1(\texttt{person})$ má ráda film $t_1(\texttt{movie})$ a $t_1(\texttt{person}) = t_3(\texttt{person})$ a pro každou $t_2 \in \text{Tupl}(\{\texttt{movie}\})$ takovou, že $t_2(\texttt{movie})$ je film od Davida Lynche platí, že osoba $t_3(\texttt{person})$ má ráda film $t_2(\texttt{movie})$“.

Bez ztráty na jednoznačnosti můžeme odstranit proměnné t_1, t_2, t_3:

„Existuje `movie` tak, že osoba `person` má ráda film `movie` a pro každý `movie` takový, že `movie` je film od Davida Lynche platí, že osoba `person` má ráda film `movie`.“

Což můžeme stručněji zapsat:

„Osoba `person` má ráda aspoň jeden film a všechny filmy od Davida Lynche.“

Jak jsme si dříve řekli, relační dělení nemusí být součástí relační algebry, protože jej dokážeme vyjádřit, jak následující věta tvrdí, pomocí operací relační algebry.

Věta 1. *Platí, že* $R_1 \div R_2 = \pi_{S_1}(R_1) - \pi_{S_1}((\pi_{S_1}(R_1) \times R_2) - R_1)$.

Důkaz: Zvolme libovolné $t_1 \in \text{Tupl}(S_1)$. Máme
$t_1 \in \pi_{S_1}(R_1) - \pi_R((\pi_{S_1}(R_1) \times R_2) - R_1)$, právě když
$t_1 \in \pi_{S_1}(R_1)$ a $t_1 \notin \pi_R((\pi_R(R_1) \times R_2) - R_1)$, právě když
$t_1 \in \pi_{S_1}(R_1)$ a pro každé $t_2 \in \text{Tupl}(S_2)$ platí, že

- $t_1 t_2 \notin (\pi_{S_1}(R_1) \times R_2) - R_1$,

právě když
$t_1 \in \pi_{S_1}(R_1)$ a pro každé $t_2 \in \mathrm{Tupl}(S_2)$ platí, že $t_1t_2 \notin \pi_{S_1}(R_1) \times R_2$ nebo $t_1t_2 \in R_1$,
právě když
$t_1 \in \pi_{S_1}(R_1)$ a pro každé $t_2 \in \mathrm{Tupl}(S_2)$ platí, že

- $t_1 \notin \pi_{S_1}(R_1)$ nebo $t_2 \notin R_2$ nebo $t_1t_2 \in R_1$,

právě když
$t_1 \in \pi_{S_1}(R_1)$ a pro každé $t_2 \in \mathrm{Tupl}(S_2)$ platí, že

- $t_1 \in \pi_{S_1}(R_1)$ a $t_2 \in R_2$ implikuje $t_1t_2 \in R_1$,

právě když
$t_1 \in \pi_{S_1}(R_1)$ a pro každé $t_2 \in \mathrm{Tupl}(S_2)$ platí, že

- $t_2 \in R_2$ implikuje $t_1t_2 \in R_1$,

právě když
$t_1 \in \pi_{S_1}(R_1)$ a pro každé $t_2 \in R_2$ platí, že $t_1t_2 \in R_1$, právě když
$t_1 \in \{t_1 \in \pi_{S_1}(R_1) \mid$ pro každé $t_2 \in R_2$ platí, že $t_1t_2 \in R_1\}$. □

Relační dělení není přímo podporováno SQL, ale díky předchozí větě jej umíme spočítat.

Uložme si relace z motivačního příkladu do proměnných.

```
# TABLE liked;

 person |    movie
--------+-------------
 Anna   | Blue Velvet
 Anna   | Eraserhead
 Bert   | Blue Velvet
 Bert   | The Matrix
 Cyril  | Blue Velvet
 Cyril  | Eraserhead
 Cyril  | The Matrix
(7 rows)

# TABLE lynch_movies;

    movie
-------------
 Blue Velvet
 Eraserhead
(2 rows)
```

Relační podíl `liked` a `lynch_movies` získáme výrazem:

```
# ( SELECT DISTINCT person FROM liked )
    EXCEPT
  ( SELECT   DISTINCT person
    FROM ( ( SELECT DISTINCT liked.person, lynch_movies.*
             FROM  liked, lynch_movies )
             EXCEPT
           ( TABLE liked ) )  );

 person
--------
 Anna
 Cyril
(2 rows)
```

6 Konstantní relace

Pro atribut y a prvek $d \in \mathcal{D}_y$ je

$$[y : d] = \{\{\langle y, d\rangle\}\}$$

relací nad $\{y\}$ nazývanou **singleton**.

Jak následující věta ukazuje, lze restrikci relace R nad S podle podmínky tvaru $y = d$ vyjádřit pomocí restricke podle podmínky $y = y_2$, kde $y_2 \notin S$.

Věta 2. *Platí* $\sigma_{y=d}(R) = \pi_S(\sigma_{y=y_2}(R \times [y_2 : d]))$.

Důkaz: Pro libovolné $r \in \mathrm{Tupl}(R)$ dostáváme, že
$t \in \pi_S(\sigma_{y=y_2}(R \times [y_2 : d]))$, právě když
existuje $t_2 \in \mathrm{Tupl}(S \cup \{y_2\})$ tak, že $t_2(S) = t$ a $t_2 \in R \times [y_2 : d]$ a $t_2(y) = t_2(y_2)$, právě když
existuje $t_2 \in \mathrm{Tupl}(S \cup \{y_2\})$ tak, že $t_2(S) = t$ a $t_2(S) \in T$ a $t_2(y_2) = d$ a $t_2(y) = r_2(y_2)$, právě když
$t \in R$ a $t(y) = d$, právě když
$t \in \sigma_{y=d}(R)$. □

Z předchozí věty vyplývá, že se lze omezit na restrikce podle podmínek tvaru $y_1 = y_2$.

Nechť $R = \{t_1, \ldots, t_m\}$ je relace nad $S = \{y_1, \ldots, y_n\}$. Následující relační výraz lze použít k určení relace R jménem ***relation*** ve FROM klauzuli SELECT výrazu.

```
( VALUES ( t1(y1), ..., t1(yn) ),

         ( tm(y1), ..., tm(yn) )
  AS relation ( y1, ..., yn )
```

Například:

```
# SELECT *
  FROM ( VALUES ( 'Anna', 'Blue Velvet' ),
                ( 'Anna', 'Eraserhead' ),
                ( 'Bert', 'Blue Velvet' ),
                ( 'Bert', 'The Matrix' ),
                ( 'Cyril', 'Blue Velvet' ),
                ( 'Cyril', 'Eraserhead' ),
                ( 'Cyril', 'The Matrix' ) )
         AS liked ( person, movie );

 person |    movie
--------+-------------
 Anna   | Blue Velvet
 Anna   | Eraserhead
 Bert   | Blue Velvet
 Bert   | The Matrix
 Cyril  | Blue Velvet
 Cyril  | Eraserhead
 Cyril  | The Matrix
```

Uvažujme relační proměnnou `likend`

```
# TABLE liked;

 person |    movie
--------+-------------
 Anna   | Blue Velvet
 Anna   | Eraserhead
 Bert   | Blue Velvet
 Bert   | The Matrix
 Cyril  | Blue Velvet
 Cyril  | Eraserhead
 Cyril  | The Matrix
(7 rows)
```

s charakteristickou vlastností „Osoba `person` má ráda film `movie`“. Pak relace

```
# SELECT *
  FROM   liked,
       ( VALUES ( 'Blue Velvet' ) )
        AS const ( movie_blue_velvet );

 person |    movie    | movie_blue_velvet
--------+-------------+-------------------
 Anna   | Blue Velvet | Blue Velvet
 Anna   | Eraserhead  | Blue Velvet
 Bert   | Blue Velvet | Blue Velvet
```

```
 Bert   | The Matrix  | Blue Velvet
 Cyril  | Blue Velvet | Blue Velvet
 Cyril  | Eraserhead  | Blue Velvet
 Cyril  | The Matrix  | Blue Velvet
(7 rows)
```

má charakteristickou vlastnost „Osoba `person` má ráda film `movie` a `movie_blue_velvet` je film Blue Velvet.“ Následující dvě relace jsou podle věty 2 totožné.

```
# SELECT person
  FROM   liked,
       ( VALUES ( 'Blue Velvet' ) )
         AS const ( movie_blue_velvet )
  WHERE movie = movie_blue_velvet;

 person
--------
 Anna
 Bert
 Cyril
(3 rows)

# SELECT person
  FROM   liked
  WHERE  movie = 'Blue Velvet';

 person
--------
 Anna
 Bert
 Cyril
(3 rows)
```

Otázky a úkoly na cvičení

1. Vezměme si proměnnou `director` danou vlastností: „Režisér `name`, který je významným představitelem německého expresionismu, se narodil v roce `born`.“ s hodnotou:

`name`	`born`
Fritz Lang	1890
Friedrich Wilhelm Murnau	1888
Robert Wiene	1873

   a proměnnou `movie` danou vlastností: „Vlastním film `title` vytvořený režisérem `director`.“ s hodnotou:

director	title
Friedrich Wilhelm Murnau	Faust
Friedrich Wilhelm Murnau	Der letzte Mann
Fritz Lang	M
Fritz Lang	Woman in the Moon
Federico Fellini	Otto e mezzo

Určete hodnoty a charakteristické vlastnosti následujících SELECT výrazů

(a)

```
SELECT DISTINCT born
FROM   director
```

(b)

```
SELECT DISTINCT director.name AS director_name,
                director.born AS director_born
                movie.title AS movie_title
FROM   director, movie
WHERE  director.name = movie.director
```

(c)

```
SELECT DISTINCT movie.title AS movie_title
FROM   director, movie
WHERE  director.name = movie.director
AND    director.born = 1890
```

(d)

```
SELECT DISTINCT movie1.title AS title1
                movie2.title AS title2
FROM   movie AS movie1,
       movie AS movie2
WHERE  movie1.director = movie2.director
```

2. Uvažujme proměnné z předchozího úkolu. Napište SELECT výrazy, které mají následující charakteristické vlastnosti, a určete jejich hodnoty.

   (a) „Vlastním film `title` natočený režisérem Friedrichem Wilhelmem Murnauem.“

   (b) „V roce `year` se narodil významný představitel německého expresionismu, který natočil film M.“

   (c) „Režiséři `director1` a `director2` jsou dva různí významní představitelé německého expresionismu, kteří se narodili ve stejném roce.“

   (d) „Vlastním film `movie`, který je od významného představitele německého expresionismu.“

3. Vezměme relaci R_1:

actor	movie
Johnny Depp	Alice in Wonderland
Helena Bonham Carter	Cinderella
Johnny Depp	Charlie and the Chocolate Factory
Johnny Depp	Fantastic Beasts and Where to Find Them
Cate Blanchett	Cinderella
Helena Bonham Carter	Alice in Wonderland
Helena Bonham Carter	Charlie and the Chocolate Factory

danou vlastností „Herec `actor` hrál ve filmu `movie`.“ a relaci R_2:

actor
Johnny Depp
Helena Bonham Carter

danou vlastností „Mám rád herce `actor`.“ Zapište tabulkou relaci $R_1 \div R_2$ a určete její charakteristickou vlastnost.

4. Jakou relaci obdržíme dělením $R_1 \div R_2$, když R_2 bude prázdná množina? Relaci R_1 vezmeme z předchozího úkolu.

5. Uvažujte vhodné proměnné pro relace R_1 a R_2 z třetího úkolu a přepište výpočet relace $R_1 \div R_2$ do výrazu SQL.

6. Vezměme si relace R_1 a R_2 ze třetího úkolu. Určete charakteristickou vlastnost relace: $\pi_\emptyset(\sigma_{\texttt{movie='Cinderella'}}(R_1 \bowtie R_2))$.

7. Opět si vezměme relace R_1 a R_2 ze třetího úkolu. Pomocí operací relační algebry vytvořte relaci s charakteristickou vlastností „Herci Johnny Depp a Cate Blanchett si zahráli ve stejném filmu.“

8. Vezměme dva libovolné relační výrazy v SQL ***expression1*** a ***expression2***, kde ***expression1*** je nad schématem S_1 a ***expression1*** je nad schématem S_2. Napište SELECT výraz, jehož hodnota bude rovna spojení hodnot výrazů ***expression1*** a ***expression2***.

9. Vezměme relaci R_1 danou vlastností „Herec `actor` hrál ve filmu `movie`“ a relaci R_2 danou vlastností „Vlastním film `movie`“. Pomocí operací relační algebry (bez podílu) určete relaci danou vlastností „Vlastním aspoň jeden film a herec `actor` hrál ve všech filmech, které vlastním.“

10. Dokažte, že výsledek relačního podílu je opravdu relace.

11. Vezměme disjunktní schémata S_1 a S_2, relace R_1 nad $S_1 \cup S_2$ a R_2 nad S_2. Musí být následující množina připomínající relační podíl vždy relace?

$$\{t_1 \in \mathrm{Tupl}(S_1) \mid \text{pro každé } t_2 \in R_2 \text{ platí, že } t_1 t_2 \in R_1\}$$