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

4. Relační algebra

Those who are enamored of practice without theory are like a pilot who goes into a ship without rudder or compass and never has any certainty where he is going. Practice should always be based upon a sound knowledge of theory.

Leonardo da Vinci (1452–1519)

1 Relační algbera

Relační algebra je tvořena těmito operacemi:

1. sjednocení
2. průnik
3. rozdíl
4. restrikce
5. projekce
6. spojení
7. přejmenování atributů

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 prázdným záhlavím. V dalších přednáškách se seznámíme se závažnějšími prohřešky proti relačnímu modelu.

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 možných hodnot objektových proměnných. Za prvé *n-ticový relační kalkul*, kde objektové proměnné nabývají hodnot n-tic. Za druhé *doménový relační kalkul*. Zde se hodnoty objektových proměnných jsou přímo prvky typů (například `integer`) nazývaných též *domén*. 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 Ekvivalence relačních výrazů

Dvě relace se *rovnají*, právě když jsou stejného typu (rovnají se jejich záhlaví) a jejich těla se rovnají.

Užitečné je, že můžeme znalosti množinových operací využít v relačních výrazech. Příklad následuje.

Uvažujme dva relační výrazy v_1, v_2 stejného typu. Díky vlastnostem sjednocení víme, že hodnota výrazu:

```
v1 UNION ( v2 UNION v3 )
```

bude vždy stejná jako hodnota výrazu:

```
( v1 UNION v2 ) UNION v3
```

To znamená, že se hodnoty rovnají pro libovolné hodnoty relačních proměnných ve výrazech se vyskytujících.

Máme dva relační výrazy v_1 a v_2. Výrazy v_1 a v_2 nazveme *ekvivalentní*, pokud se hodnota v_1 vždy rovná hodnotě v_2.

Platí, že výrazy

```
v1 UNION ( v2 UNION v3 )
```

a

```
( v1 UNION v2 ) UNION v3
```

jsou ekvivalentní. Závorky můžeme bez ztráty jednoznačnosti hodnoty výrazu vynechat a přímo psát:

```
v1 UNION v2 UNION v3
```

Podobně i u výrazů průniku můžeme závorky vynechat a rovnou psát:

```
v1 INTERSECT v2 INTERSECT v3
```

Z vlastností množinových operací dále plyne, že výrazy:

```
v1 INTERSECT v2
```

a

```
v1 EXCEPT ( v1 EXCEPT v2 )
```

jsou ekvivalentní. Můžeme tedy operaci průniku vyjádřit pomocí operace rozdílu.

3 Ekvivalence podmínek

Dvě podmínky c_1 a c_2 nad $A_1, \ldots, A_n$ jsou *ekvivalentní*, pokud pro každou n-tici t nad $A_1, \ldots, A_n$ platí, že podmínka c_1 je v t splněna, právě když podmínka c_2 je v t splněna.

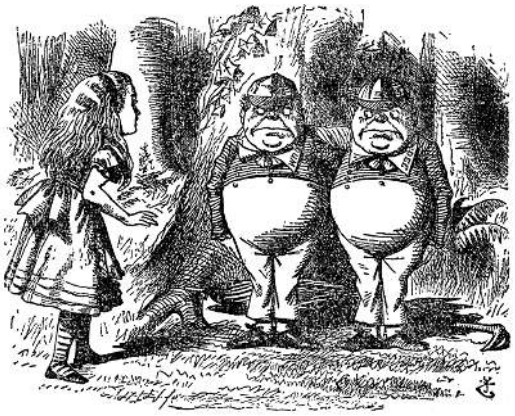
Pro práci s podmínkami můžeme využít znalosti zákonů logiky. Příklad následuje. Uvažujme tři libovolné podmínky c_1, c_2 a c_3 nad $A_1, \ldots, A_n$, pak podmínka:

```
c1 OR ( c2 OR c3 )
```

je ekvivalentní podmínce:

```
( c1 OR c2 ) OR c3
```

Závorky tedy můžeme vynechat:



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.

```
c1 OR c2 OR c3
```

Podobně lze vynechat závory i pro AND a psát:

```
c1 AND c2 AND c3
```

4 Relace s prázdným záhlavím

Prázdná množina atributů $\emptyset$ je záhlavím. Existuje jen jediná n-tice t_0 s tímto záhlavím a to prázdná množina $\emptyset$. Existují dvě relace, kde záhlaví je prázdná množina: prázdná relace a relace, jejíž tělo obsahuje pouze n-tici t_0. První se jmenuje `DUM` a druhá `DEE`. Jména relací jsou podle 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 $A_1, \ldots, A_n$ a podmínku c nad $A_1, \ldots, A_n$. Chceme zjistit, zda existuje n-tice t v těle relace r, která splňuje podmínku c. Můžeme nejprve provést restrikci relace r vzhledem k c a poté udělat projekci na prázdnou množinu atributů. Pokud výsledkem je relace `DEE`, pak odpověď je ano a pokud `DUM`, pak je odpověď ne.

V `SQL` neexistují relace s prázdným záhlavím. Tedy ani relace `DUM` a `DEE`. Každá prázdná relace reprezentuje nepravdu a každá neprázdná relace reprezentuje pravdu.

5 Relační výraz SELECT

Pokud $r_1, \ldots, r_n$ je aspoň jeden popis vstupních relací a $a_1, \ldots, a_m$ je aspoň jeden popis výstupních atributů a c je podmínka, pak

```
( SELECT DISTINCT a_1, ..., a_m
  FROM   r_1, ..., r_n
  WHERE  c )
```

je *relační výraz*.

Popis vstupní relace určuje relaci zvanou *vstupní relace* a její *jméno*. Každá vstupní relace je určitého typu. Jména vstupních relací musí být v rámci výrazu `SELECT` jedinečná. *Vstupní atribut* má tvar $R.A$, kde R je jméno vstupní relace a A je její atribut. *Popis výstupního atributu* určuje vstupní atribut a výstupní atribut. Výstupní atributy jsou v rámci výrazu `SELECT` jedinečné. Podmínka c je nad všemi vstupními atributy.

Pokud v je relační výraz a R jméno proměnné, pak

```
v AS R
```

je popis vstupní relace jménem R, která je rovna hodnotě výrazu v. Typ vstupní relace je roven typu výrazu v. Například:

```
( TABLE child ) AS ch
```

popisuje relaci, která vznikne vyhodnocením výrazu `TABLE child` a bude se jmenovat `ch`.

Pokud $R.A$ je vstupní atribut a B je atribut, pak

```
R.A AS B
```

je popis výstupního atributu B s vstupním atributem $R.A$. Například:

```
ch.name AS child_name
```

je popis výstupního atributu `child_name` se vstupním atributem `ch.name`.

Příklad výrazu `SELECT`:

```
SELECT DISTINCT p.parent_name AS parent_name, ch.age AS child_age
FROM ( TABLE parent ) AS p, ( TABLE child ) AS ch
WHERE  p.child_name = ch.name
```

Výraz se vyhodnotí v následujících krocích:

1. Získáme vstupní relace $r_1, \ldots, r_n$.
2. Přejmenujeme každý atribut A_j ve vstupní relaci r_i na $R_i.A_j$. Tím získáme relace $r'_1, \ldots, r'_n$.
3. Spočítáme spojení relací $r'_1, \ldots, r'_n$. Vzhledem k tomu, že každé dvě relace mají disjunktní záhlaví, bude se jednat o kartézský součin. Získáme relaci s_1.
4. Dále se provede restrikce relace s_1 vzhledem k podmínce c. Jako výsledek obdržíme relaci s_2.
5. Následuje projekce relace s_2 na vstupní atributy uvedené v popisech výstupních atributů. Obdržíme relaci s_3.
6. Nakonec se provede přejmenování vstupních atributů v záhlaví s_3 na výstupní atributy. Získáme výstupní relaci s_4.

Výpočet hodnoty výrazu `SELECT` používá pouze operací relační algebry. Můžeme jej tedy chápat jako zkratku za uvedené operace.

`SELECT` z předchozí ukázky by se vyhodnotil na:

```
# TABLE parent;

 parent_name | child_name
-------------+------------
 Pavel       | Anna
 Monika      | Bert
 Petr        | Bert
 Marie       | Daniela
(4 rows)

# TABLE child;

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

# SELECT DISTINCT p.parent_name AS parent_name, ch.age AS child_age
  FROM ( TABLE parent ) AS p, ( TABLE child ) AS ch
  WHERE  p.child_name = ch.name;

 parent_name | child_age
-------------+-----------
 Pavel       |         3
 Petr        |         4
 Monika      |         4
(3 rows)
```

Rozebereme si vyhodnocení výrazu krok za krokem.

1. Získáme vstupní relace. Obdržíme vstupní relaci r_1 jménem `p`:

parent_name	child_name
Pavel	Anna
Monika	Bert
Petr	Bert
Marie	Daniela

   a r_2 jménem `ch`:

name	age
Anna	3
Bert	4
Cyril	4

2. Provedeme přejmenování atributů vstupních relací na vstupní atributy. Zís-

káme relaci r'_1:

p.parent_name	p.child_name
Pavel	Anna
Monika	Bert
Petr	Bert
Marie	Daniela

a relaci r'_2:

ch.name	ch.age
Anna	3
Bert	4
Cyril	4

3. Provedeme spojení (kartézský součin) relací r'_1 a r'_2. Obdržíme relaci s_1:

p.parent_name	p.child_name	ch.name	ch.age
Pavel	Anna	Anna	3
Monika	Bert	Anna	3
Petr	Bert	Anna	3
Marie	Daniela	Anna	3
Pavel	Anna	Bert	4
Monika	Bert	Bert	4
Petr	Bert	Bert	4
Marie	Daniela	Bert	4
Pavel	Anna	Cyril	4
Monika	Bert	Cyril	4
Petr	Bert	Cyril	4
Marie	Daniela	Cyril	4

4. Provedeme restrikci s_1 vzhledem k podmínce `p.child_name = ch.name`. Výsledkem bude relace s_2:

p.parent_name	p.child_name	ch.name	ch.age
Pavel	Anna	Anna	3
Monika	Bert	Bert	4
Petr	Bert	Bert	4

5. Spočítáme projekci relace s_2 na `p.parent_name` a `ch.age`. Získáme relaci s_3:

p.parent_name	ch.age
Pavel	3
Monika	4
Petr	4

6. Nakonec přejmenujeme atributy `p.parent_name` a `ch.age` v záhlaví s_3 na `parent_name` a `child_age`. Zde je výstupní relace s_4:

parent_name	child_age
Pavel	3
Monika	4
Petr	4

Stejnou hodnotu můžeme dostat použitím operací relační algebry:

```
# SELECT DISTINCT parent_name, child_age
  FROM (
         SELECT *
         FROM ( TABLE parent ) AS t1
         NATURAL JOIN (
                        SELECT name AS child_name, age AS child_age
                        FROM ( TABLE child ) AS t
                      ) AS t2
       ) AS t;


 parent_name | child_age
-------------+-----------
 Pavel       |         3
 Monika      |         4
 Petr        |         4
```

Každá relační proměnná R popisuje relaci jménem R. Vstupní relace je rovna hodnotě proměnné R. Typ vstupní relace je roven typu proměnné R. Například `child` je popis vstupní relace jménem `child`, která je hodnotou relační proměnné `child`.

Ukázkový výraz `SELECT` můžeme zapsat i takto:

```
# SELECT DISTINCT parent.parent_name AS parent_name,
                  child.age AS child_age
  FROM   parent, child
  WHERE  parent.child_name = child.name;

 parent_name | child_age
-------------+-----------
 Pavel       |         3
 Petr        |         4
 Monika      |         4
(3 rows)
```

Pokud R je relační proměnná a S jméno relace, pak:

```
R AS S
```

popisuje relaci jménem S. Vstupní relace je rovna hodnotě proměnné R. Typ vstupní

relace je roven typu proměnné R.

Příklad použití:

```
# SELECT DISTINCT p.parent_name AS parent_name, ch.age AS child_age
  FROM   parent AS p, child AS ch
  WHERE  p.child_name = ch.name;

 parent_name | child_age
-------------+-----------
 Pavel       |         3
 Petr        |         4
 Monika      |         4
(3 rows)
```

Každý vstupní atribut $R.A$ je popis atributu s výstupním atributem A.

Popis výstupního atributu můžeme zjednodušit:

```
# SELECT DISTINCT p.parent_name, ch.age AS child_age
  FROM   parent AS p, child AS ch
  WHERE  p.child_name = ch.name;

 parent_name | child_age
-------------+-----------
 Pavel       |         3
 Petr        |         4
 Monika      |         4
(3 rows)
```

Místo vstupního atributu $R.A$ můžeme psát jen A, pokud neexistuje jiný vstupní atribut $R'.A$

Předchozí pravidlo přinese další zjednodušení:

```
# SELECT DISTINCT parent_name, age AS child_age
  FROM   parent, child
  WHERE  child_name = name;

 parent_name | child_age
-------------+-----------
 Pavel       |         3
 Petr        |         4
 Monika      |         4
(3 rows)
```

Je dána neprázdná relace r nad $A_1, \ldots, A_n$. Tělo r je množina n-tic $\{t_1, \ldots, t_m\}$. Pro každé $1 \leq i \leq n$ a $1 \leq j \leq m$ je v_{ij} hodnota, kterou přiřadí n-tice t_j atributu A_i. Zvolme jméno relace R. Pak

```
( VALUES ( v11, ..., v1n ),
                 ⋮
         ( vm1, ..., vmn ) ) AS R ( A1, ..., An ) )
```

je popis vstupní relace r jménem R. Typ vstupní relace je $\{A_1, \ldots, A_n\}$.

Například tento dotaz zjistí stáří vyjmenovaných dětí:

```
# SELECT DISTINCT age
  FROM   child, ( VALUES ( 'Anna' ), ( 'Bert' ) ) AS const ( val )
  WHERE  name = val;

 age
-----
   3
   4
(2 rows)
```

Víme, že věk každého dítěte je buď tři nebo čtyři.

V popisu atributů můžeme napsat znak hvězdička (*), který je zkratkou za všechny vstupní atributy. Hvězdičku můžeme použít pouze v případě, že vstupní relace mají atributy s jedinečnými jmény. Například:

```
# SELECT DISTINCT *
  FROM   parent, child
  WHERE  child_name = name;

 parent_name | child_name | name | age
-------------+------------+------+-----
 Petr        | Bert       | Bert |   4
 Monika      | Bert       | Bert |   4
 Pavel       | Anna       | Anna |   3
(3 rows)
```

Hvězdička je zde zkratkou za `parent.parent_name`, `parent.child_name`, `child.name`, `child.age`:

```
# SELECT DISTINCT parent.parent_name, parent.child_name,
                  child.name, child.age
  FROM   parent, child
  WHERE  child_name = name;

 parent_name | child_name | name | age
-------------+------------+------+-----
 Petr        | Bert       | Bert |   4
 Monika      | Bert       | Bert |   4
 Pavel       | Anna       | Anna |   3
(3 rows)
```

Výraz `SELECT` je rozdělen do tří částí nazývaných *klauzule*. Klauzule `SELECT`:

`SELECT DISTINCT` $a_1, \ldots, a_m$

klauzule `FROM`:

`FROM` $r_1, \ldots, r_n$

a klauzule `WHERE`:

`WHERE` c

Klauzule `WHERE` je nepovinná a může být vynechána. V takovém případě se přeskočí čtvrtý bod vyhodnocení výrazu `SELECT` počítající restrikci. Například:

```
# SELECT DISTINCT name FROM child;

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

6 Relační dělení

Jak víme, relační dělení bylo součástí originální relační algebry.

Pokud t_1 je n-tice nad $A_1, \ldots A_n$ a t_2 je n-tice nad $B_1 \ldots, B_m$ a $\{A_1, \ldots A_n\}$ a $\{B_1 \ldots, B_m\}$ jsou disjunktní, pak $t_1 \cup t_2$ je n-tice nad $A_1, \ldots A_n, B_1 \ldots, B_m$.

Uvažujme relace r_1 nad $A_1, \ldots, A_n$ a r_2 nad $A_m, \ldots, A_n$, kde $1 < m \leq n$.

Výsledkem *dělení* relace r_1 relací r_2 je relace r' nad $A_1, \ldots, A_{m-1}$. Tělo r' obsahuje všechny n-tice t' nad $A_1, \ldots, A_{m-1}$, které splňují následující podmínku. Pro každou n-tici t_2 v těle r_2 je $t' \cup t_2$ v těle r_1.

Například uvažujme relační proměnné `completed` a `disk_course`:

```
# TABLE completed;

 student | task
---------+-------
 Anna    | DISK1
 Anna    | DISK2
 Bert    | DISK1
 Bert    | PAPR1
 Cyril   | DISK2
 Cyril   | DISK1
 Cyril   | PAPR1
(7 rows)

# TABLE disk_course;

 task
-------
 DISK1
 DISK2
(2 rows)
```

Relační proměnná `completed` obsahuje relaci r_1 zachycující předměty splněné studenty. Relační proměnná `disk_course` obsahuje relaci r_2, která vyjadřuje jaké předměty patří do kurzu diskrétních struktur.

Dělením relace r_1 relací r_2 dostaneme relaci všech studentů, které splnily všechny předměty patřící do kurzu diskrétních struktur:

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

Dělení můžeme vyjádřit pomocí představených operací relační algebry. Postup ukážeme na předchozím příkladu. Zde r_1 bude hodnota proměnné `completed` a r_2 hodnota proměnné `disk_course`. Čísla n a m jsou dva. Relace r_1 je nad `student` a `task` a relace r_2 nad `task`. Atribut A_1 je `student` a atribut A_2 je `task`.

Nejprve provedeme kartézský součin všech projekcí r_1 na atribut A_i $(1 \leq i \leq m-1)$ a relace r_2:

```
# SELECT *
  FROM (SELECT DISTINCT student FROM completed) AS t1
  NATURAL JOIN (TABLE disk_course) AS t2;

 student | task
---------+-------
 Anna    | DISK1
 Bert    | DISK1
 Cyril   | DISK1
 Anna    | DISK2
 Bert    | DISK2
 Cyril   | DISK2
(6 rows)
```

Výsledkem je relace vyjadřující, že student mohl splnit předmět z kurzu diskrétních struktur.

Od výsledku odečteme relaci r_2:

```
# ( SELECT *
    FROM (SELECT DISTINCT student FROM completed) AS t1
    NATURAL JOIN (TABLE disk_course) AS t2  )
    EXCEPT
  ( TABLE completed );

 student | task
---------+-------
 Bert    | DISK2
(1 row)
```

Máme relaci vystihující, zda student nesplnil předmět z diskrétních struktur.

Uděláme projekci výsledku na $A_1, \ldots, A_{m-1}$:

```
# SELECT DISTINCT student
  FROM (
         ( SELECT *
           FROM ( SELECT DISTINCT student FROM completed ) AS t1
           NATURAL JOIN (TABLE disk_course) AS t2 )
           EXCEPT
         ( TABLE completed )
       ) AS t;

 student
---------
 Bert
(1 row)
```

Máme množinu studentů, kteří nesplnili aspoň jeden předmět z diskrétních struktur.

Nakonec odečteme od projekce r_1 na $A_1, \ldots, A_{m-1}$ předchozí výsledek:

```
# ( SELECT DISTINCT student FROM completed )
    EXCEPT
  ( SELECT DISTINCT student
    FROM (
           ( SELECT *
             FROM ( SELECT DISTINCT student FROM completed ) AS t1
             NATURAL JOIN (TABLE disk_course) AS t2 )
             EXCEPT
           ( TABLE completed )
         ) AS t );

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

Zde již máme množinu všech studentů, kteří splnili všechny předměty z kurzu diskrétních struktur.

Ukázali jsme, že relační dělení lze uskutečnit pomocí operací relační algebry.