



Databázové systémy

Spojení v relačním modelu a fyzická reprezentace dat

Petr Krajča



Katedra informatiky
Univerzita Palackého v Olomouci



Sjednocení (zřetězení)

Mějme n -tice $r \in \prod_{y \in R} D_y$ a $s \in \prod_{y \in S} D_y$ takové, že $r(y) = s(y)$ pro každý atribut $y \in R \cap S$. Zobrazení $r \cup s$ (zkráceně) rs nazveme sjednocení (zřetězení) n -tic r a s .

Projekce (zúžení)

Mějme n -tici $r \in \prod_{y \in R} D_y$, pak pro $S \subseteq R$ definujeme $r(S) \in \prod_{y \in S} D_y$ tak, že $(r(S))(y) = r(y)$ pro každý $y \in S$. Zobrazení $r(S)$ se nazývá projekce r na S .

Poznámky:

- sjednocení je komutativní ($sr = rs$), asociativní ($r(st) = (rs)t$), idempotentní ($rr = r$), neutrální vzhledem k \emptyset ($r\emptyset = r$)
- sjednocení n -tic $r \cup s$ je množinově teoretické sjednocení zobrazení, odtud:

$$(rs)(y) = \begin{cases} r(y), & \text{pokud } y \in R \\ s(y), & \text{jinak.} \end{cases}$$

Mějme relace \mathcal{D}_1 nad schématem $R \cup S$ a \mathcal{D}_2 nad schématem $S \cup T$ tak, že $R \cap T = \emptyset$.
Relace $\mathcal{D}_1 \bowtie \mathcal{D}_2$ nad $R \cup S \cup T$ definovaná

$$\mathcal{D}_1 \bowtie \mathcal{D}_2 = \{rst \in \prod_{y \in R \cup S \cup T} D_y \mid rs \in \mathcal{D}_1, st \in \mathcal{D}_2 \text{ a } s \in \prod_{y \in S} D_y\}$$

se nazývá **(přirozené) spojení relací** \mathcal{D}_1 a \mathcal{D}_2 .

Poznámky:

- S je množina společných atributů \mathcal{D}_1 a \mathcal{D}_2
- $\mathcal{D}_1 \bowtie \mathcal{D}_2$ obsahuje všechny atributy z obou tabulek
- interpretace: spojení relací $\mathcal{D}_1 \bowtie \mathcal{D}_2$ je relace obsahující spojení všech spojitelných n-tic a je to nejmenší relace této vlastnosti.

| FOO | BAR | BAZ |
|-----|-----|-----|
| 444 | ghi | 103 |
| 555 | def | 102 |
| 555 | ghi | 103 |
| 666 | abc | 101 |

 \bowtie

| BAR | BAZ | QUX |
|-----|-----|-----|
| abc | 111 | zzz |
| def | 102 | www |
| def | 102 | yyy |
| ghX | 103 | xxx |
| ghi | 103 | ttt |
| ghi | 103 | uuu |
| ghi | 103 | vvv |

 $=$

| FOO | BAR | BAZ | QUX |
|-----|-----|-----|-----|
| 555 | def | 102 | www |
| 555 | def | 102 | yyy |
| 444 | ghi | 103 | vvv |
| 555 | ghi | 103 | vvv |
| 444 | ghi | 103 | ttt |
| 555 | ghi | 103 | ttt |
| 444 | ghi | 103 | uuu |
| 555 | ghi | 103 | uuu |

| FOO | BAR | BAZ |
|-----|-----|-----|
| 555 | def | 102 |
| 555 | ghi | 103 |
| 666 | abc | 101 |

 \bowtie

| BAR | BAZ | QUX |
|-----|-----|-----|
| AAA | 101 | AAA |
| def | 102 | BBB |
| ghi | 333 | CCC |

 $=$

| FOO | BAR | BAZ | QUX |
|-----|-----|-----|-----|
| 555 | def | 102 | BBB |

| FOO | BAR | BAZ |
|-----|-----|-----|
| 333 | jkl | 103 |
| 444 | ghi | 102 |
| 555 | def | 101 |
| 666 | abc | 101 |

 \bowtie

| BAZ |
|-----|
| 101 |
| 103 |

 $=$

| FOO | BAR | BAZ |
|-----|-----|-----|
| 333 | jkl | 103 |
| 555 | def | 101 |
| 666 | abc | 101 |



```
SELECT * FROM tab1 NATURAL JOIN tab2;
```

```
SELECT tab1.*, tab2.t_attr1, tab2.t_attr_2, tab2.t_attrn  
FROM tab1, tab2  
WHERE tab1.s_attr1 = tab2.s_attr1 AND  
       tab1.s_attr2 = tab2.s_attr2 AND  
       tab1.s_attrn = tab2.s_attrm
```

- s_attrX – atributy společné tab1 i tab2
- t_attrX – atributy pouze z tab2 tab2

(Přirozené) Spojení: Příklad



```
SELECT * FROM tab1 NATURAL JOIN tab2;  
SELECT * FROM tab1, tab2  
    WHERE tab1.bar = tab2.bar AND  
          tab1.baz = tab2.baz
```

| tab1 | | | tab2 | | | | | | |
|------|-----|-----|------|-----|-----|-----|-----|-----|-----|
| foo | bar | baz | bar | baz | qux | foo | bar | baz | qux |
| 444 | ghi | 103 | abc | 111 | zzz | 555 | def | 102 | www |
| 555 | def | 102 | def | 102 | www | 555 | def | 102 | yyy |
| 555 | ghi | 103 | def | 102 | yyy | 444 | ghi | 103 | vvv |
| 666 | abc | 101 | ghX | 103 | xxx | 555 | ghi | 103 | vvv |
| | | | ghi | 103 | ttt | 444 | ghi | 103 | ttt |
| | | | ghi | 103 | uuu | 555 | ghi | 103 | ttt |
| | | | ghi | 103 | vvv | 444 | ghi | 103 | uuu |
| | | | | | | 555 | ghi | 103 | uuu |

Pro libovolné relace $\mathcal{D}, \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ platí:

- $\mathcal{D} \bowtie \mathcal{D} = \mathcal{D}$
- $\mathcal{D}_1 \bowtie \mathcal{D}_2 = \mathcal{D}_2 \bowtie \mathcal{D}_1$
- $\mathcal{D}_1 \bowtie (\mathcal{D}_2 \bowtie \mathcal{D}_3) = (\mathcal{D}_1 \bowtie \mathcal{D}_2) \bowtie \mathcal{D}_3$
- $\mathcal{D} \bowtie \mathcal{D}_\top = \mathcal{D}$, kde \mathcal{D}_\top je relace nad schématem \emptyset obsahující právě jednu n-tici
- $\mathcal{D} \bowtie \emptyset_S = \emptyset_{R \cup S}$ pro \mathcal{D} na R a pro libovolné S .

Nechť $\mathcal{D}_1, \mathcal{D}_2$ jsou relace nad stejným schématem, pak platí

- $\mathcal{D}_1 \bowtie \mathcal{D}_2 = \mathcal{D}_1 \cap \mathcal{D}_2$.

Pokud je \mathcal{D} relace na schématu R , $y \in R$ a $d \in D_y$, pak platí

- $\sigma_{y=d}(\mathcal{D}) = \mathcal{D} \bowtie \{\{\langle y, d \rangle\}\}$.



Mějme relace \mathcal{D}_1 a \mathcal{D}_2 na schématech R a T takových, že $R \cap T = \emptyset$. Pak $\mathcal{D}_1 \bowtie \mathcal{D}_2$ se nazývá **kartézský součin** relací \mathcal{D}_1 a \mathcal{D}_2 .

Poznámky:

- speciální případ přirozeného spojení pro $S = \emptyset$
- každá n -tice je spojitelná
- $|\mathcal{D}_1 \bowtie \mathcal{D}_2| = |\mathcal{D}_1| \cdot |\mathcal{D}_2|$ (obecně platí pouze \leq)

Pozor!!!

- 1 (naivní) kartézský součin množin (v daném pořadí), tj. $A \times B$
- 2 obecný kartézský součin (indexovaného systému množin), tj. $\prod_{i \in I} A_i$
- 3 kartézský součin relací nad relačními schématy, tj. $\mathcal{D}_1 \bowtie \mathcal{D}_2$

SQL

```
SELECT tabulka1.*, tabulka2.* FROM tabulka1, tabulka2;
```

```
SELECT tabulka1.*, tabulka2.* FROM tabulka1 CROSS JOIN tabulka2;
```


Motivace: chceme spojovat data ze dvou tabulek tak, aby kritérium spojitelnosti nebylo dáno pouze rovností, ale obecnou podmínkou vztahující se na hodnoty n -tic z obou tabulek

Mějme relace \mathcal{D}_1 a \mathcal{D}_2 na schématech R a T takových, že $R \cap T = \emptyset$ a necht' θ je skalární výraz typu „pravdivostní hodnota“, který může obsahovat jména atributů z $R \cup T$, pak relaci

$$\mathcal{D}_1 \bowtie_{\theta} \mathcal{D}_2 = \sigma_{\theta}(\mathcal{D}_1 \bowtie \mathcal{D}_2)$$

nazýváme θ -spojení \mathcal{D}_1 a \mathcal{D}_2 **splňující θ**

Poznámka:

- z definice σ_{θ} a $\mathcal{D}_1 \bowtie \mathcal{D}_2$ plyne:

$$\mathcal{D}_1 \bowtie_{\theta} \mathcal{D}_2 = \{rt \mid r \in \mathcal{D}_1 \text{ a } t \in \mathcal{D}_2 \text{ tak, že } rt \text{ splňuje } \theta\}$$

SQL

```
SELECT t1.*, t2.* FROM tabulka t1, tabulka t2
WHERE t1.attr > t2.attr;
```



Mějme relace \mathcal{D}_1 a \mathcal{D}_2 na schématech R a T takových, že $R \cap T = \emptyset$ a necht' $\{y_1, \dots, y_n\} \subseteq R$ a $\{y'_1, \dots, y'_n\} \subseteq T$ tak, že atributy y_i a y'_i mají stejný typ. Pak θ -spojení \mathcal{D}_1 a \mathcal{D}_2 , kde θ je ve tvaru $y_1 = y'_1 \wedge \dots \wedge y_n = y'_n$ nazýváme **spojení na rovnost** relací \mathcal{D}_1 a \mathcal{D}_2 a označujeme jej $\mathcal{D}_1 \bowtie_{y_1=y'_1, \dots, y_n=y'_n} \mathcal{D}_2$.

SQL

```
SELECT t1.*, t2.* FROM tabulka1 t2 , tabulka2 t2
WHERE (t1.attr1 = t2.attr1) AND (t1.attr2 = t2.attr2);
```



Mějme relace \mathcal{D}_1 a \mathcal{D}_2 na schématech R_1 a R_2 . Pak

$$\mathcal{D}_1 \bowtie \mathcal{D}_2 = \pi_{R_1}(\mathcal{D}_1 \bowtie \mathcal{D}_2).$$

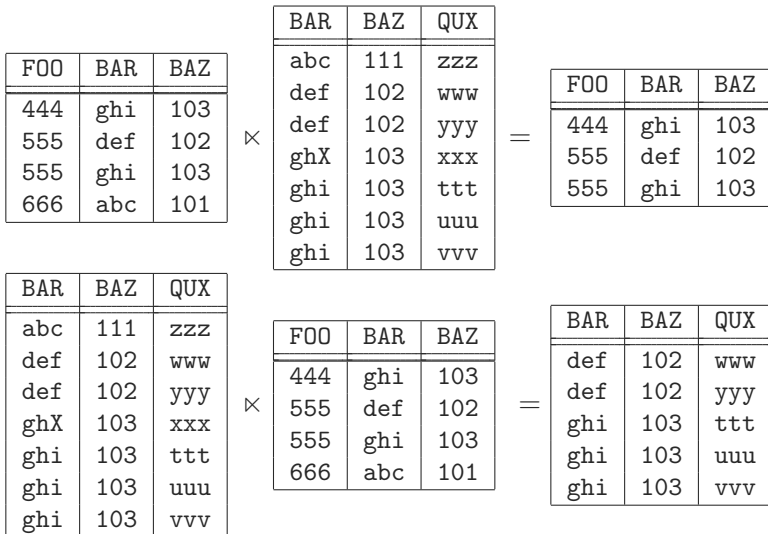
se nazývá **polospojení** relací \mathcal{D}_1 a \mathcal{D}_2

Poznámka:

- $r \in \mathcal{D}_1 \bowtie \mathcal{D}_2$ právě tehdy, když $r_1 \in \mathcal{D}_1$ a r_1 je spojitelná s nějakou $r_2 \in \mathcal{D}_2$.

SQL

```
SELECT DISTINCT tabulka1.* FROM tabulka1 NATURAL JOIN tabulka2;
```





dotazy ve tvaru „některá A jsou B”

- *Studenti, kteří mají zapsaný nějaký předmět.*
- *Součástky, které patří do některého výrobku.*
- *Dodavatelé, kteří dodávají některé výrobky.*
- ...

dotazy ve tvaru „všechna A jsou B”

- *Studenti, kteří mají splněny všechny vyžadované předměty.*
- *Výrobky, jejichž všechny součástky jsou skladem.*
- *Dodavatelé, kteří dodávají všechny výrobky.*
- ...

Vyjadřování dotazů **existenčního** a **všeobecného** charakteru:

- **existenční** – polospojení (projekce a přirozené spojení)
- **všeobecný** – dělení (množinový rozdíl, přirozené spojení, projekce)

Uvažujme následující relace: \mathcal{D}_1 (dělenec) na schématu R , \mathcal{D}_2 (dělitel) na schématu S a \mathcal{D}_3 (prostředník) na schématu T . Pak relace

$$\mathcal{D}_1 \div_{\mathcal{D}_3} \mathcal{D}_2 = \{r(R \cap T) \mid r \in \mathcal{D}_1 \text{ tak, že pro každou } s \in \mathcal{D}_2 \text{ splňující podmínku } r(R \cap S \cap T) = s(R \cap S \cap T) \text{ platí, že } r(R \cap T)s(S \cap T) \in \pi_{(R \cup S) \cap T}(\mathcal{D}_3)\}$$

se nazývá **podíl** \mathcal{D}_1 a \mathcal{D}_2 přes \mathcal{D}_3 na schématu $R \cap T$.

Interpretace pro \mathcal{D}_1 (výrobci), \mathcal{D}_2 (výrobek), \mathcal{D}_3 a (kdo vyrábí co):

- $\mathcal{D}_1 \div_{\mathcal{D}_3} \mathcal{D}_2$ – výrobci vyrábějící všechny výrobky
- ...ještě jinak: r z dělence (výrobce) je ve výsledném podílu právě tehdy, pokud pro každou s (výrobek) z dělitele platí, že rs je v prostředníkovu (výrobce-výrobek), tj. výrobce vyrábí každý výrobek.
- $\mathcal{D}_2 \div_{\mathcal{D}_3} \mathcal{D}_1$ – výrobky vyráběné všemi výrobci

Příklad dat pro relační dělení



| NAME |
|---------|
| Abbe |
| Blangis |
| Curval |
| Durcet |

| COURSE |
|-----------|
| KMI/DATA1 |
| KMI/DATA2 |
| KMI/PAPR1 |

| NAME | COURSE |
|---------|-----------|
| Abbe | KMI/DATA1 |
| Abbe | KMI/DATA2 |
| Blangis | KMI/DATA1 |
| Blangis | KMI/DATA2 |
| Blangis | KMI/PAPR1 |
| Curval | KMI/DATA1 |
| Curval | KMI/DATA2 |
| Curval | KMI/PAPR1 |

| NAME | MAJOR |
|---------|-------|
| Abbe | CS |
| Blangis | CS |
| Curval | EE |
| Durcet | SS |

| COURSE | VERSION |
|-----------|---------|
| KMI/DATA1 | 2 |
| KMI/DATA2 | 1 |
| KMI/PAPR1 | 2 |

| NAME | COURSE | YEAR |
|---------|-----------|------|
| Abbe | KMI/DATA1 | 2011 |
| Abbe | KMI/DATA2 | 2012 |
| Blangis | KMI/DATA1 | 2012 |
| Blangis | KMI/DATA2 | 2012 |
| Blangis | KMI/PAPR1 | 2007 |
| Curval | KMI/DATA1 | 2011 |
| Curval | KMI/DATA2 | 2013 |
| Curval | KMI/PAPR1 | 2008 |



Pokud platí $R \cap S = \emptyset$:

$\mathcal{D}_1 \div_{\mathcal{D}_3} \mathcal{D}_2 = \{r(R \cap T) \mid r \in \mathcal{D}_1 \text{ tak, že pro každou } s \in \mathcal{D}_2 \text{ platí, že } r(R \cap T)s(S \cap T) \in \pi_{(R \cup S) \cap T}(\mathcal{D}_3)\}$

Pokud platí $R \cap S = \emptyset$ a $T = R \cup S$:

$\mathcal{D}_1 \div_{\mathcal{D}_3} \mathcal{D}_2 = \{r \in \mathcal{D}_1 \mid \text{pro každou } s \in \mathcal{D}_2 \text{ platí, že } rs \in \mathcal{D}_3\}$



Pro \mathcal{D}_1 na R , \mathcal{D}_2 na S , \mathcal{D}_3 na T platí:

$$\mathcal{D}_1 \div_{\mathcal{D}_3} \mathcal{D}_2 = \pi_{R \cap T}(\mathcal{D}_1) \setminus \pi_{R \cap T}((\pi_{R \cap T}(\mathcal{D}_1) \bowtie \pi_{S \cap T}(\mathcal{D}_2)) \setminus \pi_{(R \cup S) \cap T}(\mathcal{D}_3)).$$

Neformální skica důkazu

- $\mathcal{D}_a = \pi_{R \cap T}(\mathcal{D}_1) \bowtie \pi_{S \cap T}(\mathcal{D}_2)$ – všechny kombinace hodnot, které můžeme uvažovat v \mathcal{D}_3
- $\mathcal{D}_b = (\mathcal{D}_a \setminus \pi_{(R \cup S) \cap T}(\mathcal{D}_3))$ – všechny n-tice, které mohly být v \mathcal{D}_3 , ale nejsou.
- $\mathcal{D}_c = \pi_{R \cap T}(\mathcal{D}_b)$ – všechny n-tice z \mathcal{D}_1 , které nemají v \mathcal{D}_3 **nějakou** spojitelnou n-tici.
- $\mathcal{D}_d = \pi_{R \cap T}(\mathcal{D}_1) \setminus \mathcal{D}_c$ – ponechá n-tice \mathcal{D}_1 , kterým nechybí spojitelná n-tice v \mathcal{D}_3 .

- do relačního modelu byl přidán příznak ω značící absenci hodnoty; v SQL NULL
- řeší některé problémy (např. nespojitelné n-tice)
- přináší řadu nových problémů
- formální model s ω má nedostatky, potenciál pro vznik chyb

Přístup k nedefinovaným hodnotám v SQL

- aritmetická operace a NULL dává NULL ($1 + \text{NULL}$) \rightarrow NULL
- logický operátor a NULL dává NULL ($1 = \text{NULL}$) \rightarrow NULL
- pozor, i ($\text{NULL} = \text{NULL}$) \rightarrow NULL (potřeba použít operátor `foo IS NULL`)
- Kleeneho tříhodnotová logika

| \wedge | 0 | ω | 1 |
|----------|---|----------|----------|
| 0 | 0 | 0 | 0 |
| ω | 0 | ω | ω |
| 1 | 0 | ω | 1 |

| \vee | 0 | ω | 1 |
|----------|----------|----------|---|
| 0 | 0 | ω | 1 |
| ω | ω | ω | 1 |
| 1 | 1 | 1 | 1 |

| \rightarrow | 0 | ω | 1 |
|---------------|----------|----------|---|
| 0 | 1 | 1 | 1 |
| ω | ω | ω | 1 |
| 1 | 0 | ω | 1 |

| \neg | 0 |
|----------|----------|
| 0 | 1 |
| ω | ω |
| 1 | 0 |

Příklady problémů s NULL (1/2)



```
CREATE TABLE r1 (x NUMERIC NOT NULL PRIMARY KEY, yn VARCHAR);
CREATE TABLE r2 (y NUMERIC NOT NULL PRIMARY KEY, zn VARCHAR);
INSERT INTO r1 VALUES (45, 'London');
INSERT INTO r2 VALUES (33, NULL);
SELECT x, y FROM r1, r2 WHERE (yn <> zn OR zn <> 'Paris');
```

Očekávaný výsledek:

| x | | y |
|----|--|----|
| 45 | | 33 |

Protože buď $zn = 'Paris'$ a tedy $yn \neq zn$, nebo $zn \neq 'Paris'$

Skutečný výsledek:

| x | | y |
|---|--|---|
|---|--|---|

Protože $\omega \vee \omega = \omega$



Neplatí některá tvrzení

Např. $\mathcal{D} \bowtie \mathcal{D} = \mathcal{D}$ nebo $\mathcal{D} \bowtie \mathcal{D} = \mathcal{D} \cap \mathcal{D}$

```
CREATE TABLE foo (a INTEGER PRIMARY KEY, b INTEGER);
INSERT INTO foo (a, b) VALUES (1, 10), (2, 20), (3, NULL);
```

```
SELECT * FROM foo f1
      NATURAL JOIN foo f2;
```

| a | b |
|---|----|
| 1 | 10 |
| 2 | 20 |

```
SELECT * FROM foo
INTERSECT SELECT * FROM foo;
```

| a | b |
|---|----|
| 1 | 10 |
| 2 | 20 |
| 3 | |

Nekonzistentní chování

```
SELECT 10 + 20 + NULL; // --> NULL
SELECT SUM(b) FROM foo; // --> 30
```



- pokud se v relacích, které spojujeme, nacházejí nějaké nespojitelné n-tice, ve výsledku dochází ke „ztrátě informace“
- zavedení vnějších spojení a konceptu chybějících hodnot (konceptně špatně)
- doposud diskutována **vnitřní spojení**

vnější přirozená a θ -spojení v SQL

```
SELECT * FROM tab1 NATURAL FULL OUTER JOIN tab2;
```

```
SELECT * FROM tab1 NATURAL LEFT OUTER JOIN tab2;
```

```
SELECT * FROM tab1 NATURAL RIGHT OUTER JOIN tab2;
```

```
SELECT * FROM tab1 FULL OUTER JOIN tab2 ON podmínka;
```

```
SELECT * FROM tab1 LEFT OUTER JOIN tab2 ON podmínka;
```

```
SELECT * FROM tab1 RIGHT OUTER JOIN tab2 ON podmínka;
```

- ve výsledku levého (pravého/oboustranného) vnějšího spojení jsou zahrnuty i n-tice z první (druhé/obou) relace, které jsou nespojitelné
- v tabulkách se projevuje přítomností NULL (značí „absenci hodnot“)



```
SELECT * FROM tab1 NATURAL LEFT OUTER JOIN tab2;
```

| tab1 | | | tab2 | | | | | | |
|------|-----|-----|------|-----|-----|-----|-----|-----|-----|
| foo | bar | baz | bar | baz | qux | bar | baz | foo | qux |
| 444 | ghi | 103 | abc | 111 | zzz | abc | 101 | 666 | |
| 555 | def | 102 | def | 102 | www | def | 102 | 555 | www |
| 555 | ghi | 103 | def | 102 | yyy | def | 102 | 555 | yyy |
| 666 | abc | 101 | ghX | 103 | xxx | ghi | 103 | 444 | vvv |
| | | | ghi | 103 | ttt | ghi | 103 | 444 | ttt |
| | | | ghi | 103 | uuu | ghi | 103 | 444 | uuu |
| | | | ghi | 103 | vvv | ghi | 103 | 555 | vvv |
| | | | | | | ghi | 103 | 555 | ttt |
| | | | | | | ghi | 103 | 555 | uuu |



```
SELECT * FROM tab1 NATURAL RIGHT OUTER JOIN tab2;
```

| tab1 | | | tab2 | | | | | | |
|------|-----|-----|------|-----|-----|-----|-----|-----|-----|
| foo | bar | baz | bar | baz | qux | foo | bar | baz | qux |
| 444 | ghi | 103 | abc | 111 | zzz | | abc | 111 | zzz |
| 555 | def | 102 | def | 102 | www | 555 | def | 102 | www |
| 555 | ghi | 103 | def | 102 | yyy | 555 | def | 102 | yyy |
| 666 | abc | 101 | ghX | 103 | xxx | 444 | ghi | 103 | vvv |
| | | | ghi | 103 | ttt | 555 | ghi | 103 | vvv |
| | | | ghi | 103 | uuu | 444 | ghi | 103 | ttt |
| | | | ghi | 103 | vvv | 555 | ghi | 103 | ttt |
| | | | | | | 444 | ghi | 103 | uuu |
| | | | | | | 555 | ghi | 103 | uuu |
| | | | | | | | ghX | 103 | xxx |



```
SELECT * FROM tab1 NATURAL FULL OUTER JOIN tab2;
```

| tab1 | | | tab2 | | | | | | |
|------|-----|-----|------|-----|-----|-----|-----|-----|-----|
| foo | bar | baz | bar | baz | qux | foo | bar | baz | qux |
| 444 | ghi | 103 | abc | 111 | zzz | 666 | abc | 101 | |
| 555 | def | 102 | def | 102 | www | | abc | 111 | zzz |
| 555 | ghi | 103 | def | 102 | yyy | | ghX | 103 | xxx |
| 666 | abc | 101 | ghX | 103 | xxx | 555 | def | 102 | www |
| | | | ghi | 103 | ttt | 555 | def | 102 | yyy |
| | | | ghi | 103 | uuu | 444 | ghi | 103 | vvv |
| | | | ghi | 103 | vvv | 444 | ghi | 103 | ttt |
| | | | | | | 444 | ghi | 103 | uuu |
| | | | | | | 555 | ghi | 103 | vvv |
| | | | | | | 555 | ghi | 103 | ttt |
| | | | | | | 555 | ghi | 103 | uuu |

Mějme relace \mathcal{D}_1 a \mathcal{D}_2 na schématech R_1 a R_2 takových, že $R_1 \cap R_2 = \emptyset$. Pak

$$\mathcal{D}_1 \triangleright \mathcal{D}_2 = \mathcal{D}_1 \setminus \pi_{R_1}(\mathcal{D}_1 \bowtie \mathcal{D}_2).$$

se nazývá **polorozdíl** relací \mathcal{D}_1 a \mathcal{D}_2

Poznámka:

- $r \in \mathcal{D}_1 \triangleright \mathcal{D}_2$ právě tehdy, když $r_1 \in \mathcal{D}_1$ a r_1 není spojitelná s nějakou $r_2 \in \mathcal{D}_2$.

SQL

```
SELECT * FROM tabulka1
```

```
EXCEPT
```

```
SELECT * FROM tabulka1 NATURAL JOIN tabulka2
```



tab1

| foo | bar | baz |
|-----|-----|-----|
| 444 | ghi | 103 |
| 555 | def | 102 |
| 555 | ghi | 103 |
| 666 | abc | 101 |

tab2

| bar | baz | qux |
|-----|-----|-----|
| abc | 111 | zzz |
| def | 102 | www |
| def | 102 | yyy |
| ghX | 103 | xxx |
| ghi | 103 | ttt |
| ghi | 103 | uuu |
| ghi | 103 | vvv |

$$\mathcal{D}_1 \triangleright \mathcal{D}_2$$

| foo | bar | baz |
|-----|-----|-----|
| 666 | abc | 101 |

$$\mathcal{D}_2 \triangleright \mathcal{D}_1$$

| bar | baz | qux |
|-----|-----|-----|
| ghX | 103 | xxx |
| abc | 111 | zzz |



- Snaha vytvořit nové (virtuální) relační proměnné představující abstrakci nad existujícími (základními) proměnnými a jejich typy. Hodnoty (virtuálních) proměnných jsou dány vyhodnocením dotazů.
- Virtuální relační proměnná (pohled) daného typu je proměnná, která v čase t nabývá hodnoty vzniklé vyhodnocením daného relačního výrazu v čase t .

SQL

```
CREATE VIEW nazev AS dotaz;
```

```
CREATE VIEW it_oddeleni AS  
    SELECT * FROM zamestnanci WHERE pozicie = 'IT';
```

```
SELECT * FROM it_oddeleni;
```



- operace s databází jsou prováděny v **transakcích**
- transakce zajišťují
 - přechod z jednoho konzistentního stavu databáze do druhého (ochrana před poškozením)
 - izolaci souběžně prováděných změn
- požadavky na transakci (ACID)
 - atomic – transakce musí být provedena atomicky, buď celá, nebo vůbec
 - consistent – databáze přechází z jednoho konzistentního stavu do druhého
 - isolated – izolace transakcí, dokud není změna potvrzena, jiná transakce ji nevidí
 - durable – potvrzené změny jsou trvalé

Základní operace s transakcemi

- BEGIN TRANSACTION – zahájení transakce
- COMMIT – potvrzení transakce
- ROLLBACK – vrácení změn nepotvrzené transakce



- 1 dotaz je přeložen z textové reprezentace do interní formy
 - blízká relační algebře
 - stromová struktura
 - listy – rel. proměnné (tabulky)
 - uzly – operace rel. algebry
- 2 následně jsou provedeny optimalizace a dotaz je transformován do proveditelné podoby
 - využívá se
 - vlastností rel. algebry
 - statistik o datech (velikosti tabulek, četnost hodnot, nejčastější hodnoty)
 - heuristik (známá pravidla, odhad času pro čtení z disku)
 - „abstraktní operace“ rel. algebry jsou nahrazeny fyzickými operátory určenými pro přímou práci s dat
- 3 dotaz je vyhodnocen



- množství fyzických operátorů a způsob provádění se mezi SŘBD významně liší
- většina SŘBD dovoluje zobrazit informace o tom, jak bude dotaz vyhodnocen

```
EXPLAIN SELECT * FROM foo INTERSECT SELECT * FROM foo;  
                                QUERY PLAN
```

```
-----  
HashSetOp Intersect  (cost=0.00..127.00 rows=2140 width=8)  
-> Append  (cost=0.00..105.60 rows=4280 width=8)  
    -> Subquery Scan on "*SELECT* 1"  (cost=0.00..52.80 rows=2140 width=8)  
        -> Seq Scan on foo  (cost=0.00..31.40 rows=2140 width=8)  
    -> Subquery Scan on "*SELECT* 2"  (cost=0.00..52.80 rows=2140 width=8)  
        -> Seq Scan on foo  (cost=0.00..31.40 rows=2140 width=8)
```



Naivní řešení

- σ – iterace přes všechny řádky tabulky
- $\cap, \cup, \setminus, \pi, \bowtie$ – iterace ve vnořené smyčce

Optimalizované řešení

- zjednodušování dotazů na základě vlastností operací
- využití dodatečných struktur (indexů) umožňujících rychlé vyhledávání n-tic, příp. procházení setříděnou posloupností n-tic

Poznámka:

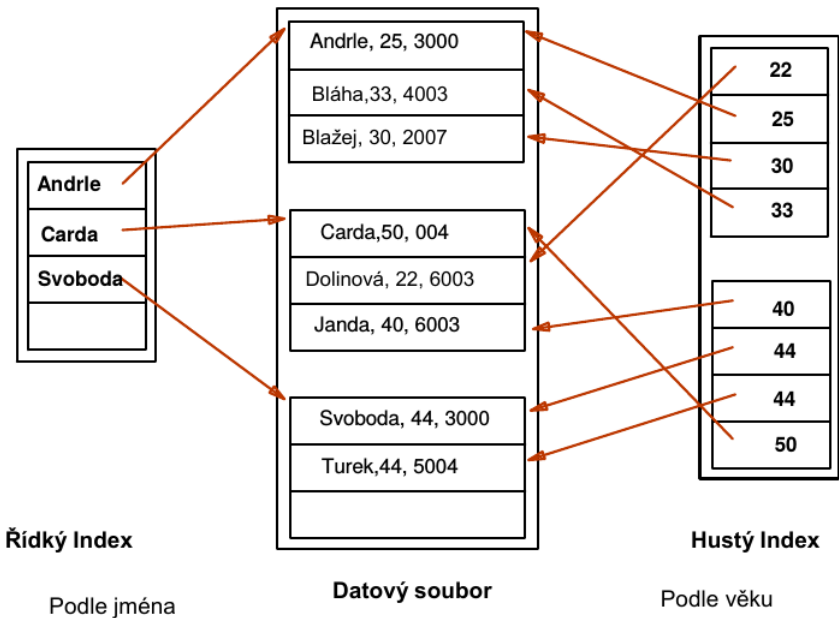
- udržování indexů má svou režii (zpomalení operací INSERT, UPDATE a DELETE)

dva základní typy struktur

- 1 indexy reprezentující **uspořádané množiny**
 - typicky perzistentní B-strom (B+strom, apod.)
 - optimalizace pro diskové (databázové) operace
 - rychlé nalezení hodnot pro podmínky s $<$, \leq , $=$, \geq , $>$
- 2 indexy reprezentující **asociační pole**
 - perzistentní hashovací tabulky
 - rychlé nalezení hodnot pro podmínky s $=$

Typy indexů

- **jednosloupcové / vícesloupcové**
- **husté / řídké** (indexovány jsou pouze první hodnoty bloků)
- index může být **unikátní**
- **úplné / částečné** (indexují jen část tabulky)





```
CREATE TABLE foo (  
x INTEGER PRIMARY KEY,  
y INTEGER NOT NULL,  
z INTEGER NOT NULL);
```

```
CREATE INDEX foo_yz_idx ON foo (y, z);  
CREATE UNIQUE INDEX foo_yz_idx ON foo (y, z);
```

```
-- vyuzitelne v dotazech typu  
SELECT * FROM foo WHERE y = 10;  
SELECT * FROM foo WHERE y > 10;  
SELECT * FROM foo ORDER BY y;  
  
-- ale uz ne pro  
SELECT * FROM foo WHERE z = 20;
```



```
-- pouziti hashe
```

```
CREATE INDEX foo_y_idx ON foo USING hash (y);
```

```
-- castecny index
```

```
CREATE INDEX foo_z_idx ON foo (z) WHERE z > 1000;
```

```
-- index na vyraz
```

```
CREATE INDEX foo_abs_idx ON foo (abs (y + z));
```

```
SELECT * FROM foo WHERE abs (y + z) > 200;
```



Nested-loop join

```
forEach r in Table1
  forEach s in Table2
    if (isJoinable(r, s))
      emit(join(rs))
```

- nepřiliš efektivní
- vnitřní smyčku lze nahradit vyhledáváním pomocí indexu



Sort-merge Join

- obě relace musí být setříděny (buď pomocí indexu nebo dočasně, jako první krok algoritmu)
- níže nastíněná verze pro unikátní indexy

```
left = fetchFirst(Table1)
right = fetchFirst(Table2)

while (left and right)
  if (isJoinable(left, right))
    emit(join(rs))
    left = fetchNext(Table1)
    right = fetchNext(Table2)
  else if (left > right)
    right = fetchNext(Table2)
  else (left < right)
    left = fetchNext(Table1)
end
```