Databázové systémy

Základní relační operace

Vilém Vychodil

KMI/DATA1, Přednáška 3

Databázové systémy

Přednáška 3: Přehled

1. Přehled relačního dotazování:
   - relační operace a relační algebra,
   - relační kalkuly.
2. Množinové relační operace:
   - průnik, sjednocení a rozdíl relací,
   - odvozené množinové operace,
   - implementace v Tutorial D a SQL.
3. Projekce a restrikce:
   - projekce, restrikce,
   - zákony a vztahy k dalším operacím,
   - implementace v Tutorial D a SQL.
4. Fyzická vrstva databáze a otázky efektivity:
   - uspořádané množiny, databázové indexy,
   - práce s indexy v SQL,
   - algoritmy pro výpočet výsledků relačních operací.

Relační model: Opakování

komponenty relačního modelu:

- typový systém: *skalární, n-ticové a relační typy*
- (*základní*) *relační proměnné* (definované typy a *klíče*)
- *relační přiřazení* – přiřazení hodnot (relací) relačním proměnným
- kolekce generických *relačních operací* pro vyjadřování relací z jiných relací

související jazyky:

- Tutorial D (implementace Rel) – relační jazyk (těsně vázaný na RM)
- SQL – jazyk podporovaný nasaditelnými SŘBD (slabší vztah k RM)

otázky:

- definice a modifikace dat (máme vyřešeno), zbývá:
- dotazování – získávání dat z databáze na základě předpisů (dotazů)

Relační dotazování

(zjednodušená) formalizace databáze a dotazů:

instance databáze, angl.: *database instance*

Instance databáze je konečná množina relačních proměnných, jejich aktuálních hodnot a integritních omezení (zatím nepotřebujeme).

dotaz, angl.: *query*

Dotaz je částečná rekurzivní funkce z množiny všech instancí databáze do možiny všech relací (nad relačními schématy).

poznámky:

- z pohledu predikátové logiky: *instance databáze* = *struktury*
- typický přístup: dotaz je popsán v určitém *jazyku* + je dána jeho *interpretace*
- (dotazovací) jazyk je **doménově nezávislý**, pokud výsledky dotazů nezávisí na typech (doménách), ale pouze na hodnotách relačních proměnných

Základní dotazovací systémy:

1. **relační algebra:**
   - specifikuje množinu **operací s relacemi**
   - dotazy = výrazy skládající se ze složených relačních operací
   - interpretace dotazů: postupné vyhodnocení operací
   - elementární operace s relacemi (SŘBD může dobře optimalizovat)
2. **relační kalkuly:**
   - několik typů: **doménový relační kalkul**, **n-ticový relační kalkul**
   - dotazy = formule predikátové logiky (s volnými proměnnými)
   - interpretace dotazů: ohodnocování formulí ve struktuře (instanci databáze)
   - ryze deklarativní, vychází z něj řada jazyků (QUEL, do jisté míry SQL)

poznámky:

- moderní překladače dotazů (obvykle) vytvářejí *plány* používající operace, které jsou blízko operacím relační algebry
- známý mýtus: relační algebra „není deklarativní“ (nedává smysl)

Typy relačních operací

motto:

Množina relačních operací přímo ovlivňuje, jak silný (expresivní) bude dotazovací jazyk, který je na ní založen.

dělení operací relační algebry:

- *základní* (minimální množina operací) / *odvozené*
- *podle počtu operandů* (operace s jednou, dvěma, třemi, … relacemi)
- *podle významu* (množinové operace, protějšky kvantifikátorů, …)

  ⋮

rozumná množina operací:

Množina operací, která zaručuje, že relační algebra je stejně silná jako (doménově nezávislý) doménový relační kalkul.

Množinové operace: Průnik relací

Definice (průnik relací, angl.: *intersection*)

Pro dvě relace $\mathcal{D}_1$ a $\mathcal{D}_2$ na relačním schématu $R \subseteq Y$ zavádíme

$$\mathcal{D}_1 \cap \mathcal{D}_2 = \{r \in \textstyle\prod_{y \in R} D_y \,|\, r \in \mathcal{D}_1 \text{ a zároveň } r \in \mathcal{D}_2\}.$$

Relace $\mathcal{D}_1 \cap \mathcal{D}_2$ na schématu R se nazývá **průnik relací** $\mathcal{D}_1$ a $\mathcal{D}_2$.

Tutorial D:

$\langle \textit{relační-výraz}_1 \rangle$ `INTERSECT` $\langle \textit{relační-výraz}_2 \rangle$

`INTERSECT` `{`$\langle \textit{relační-výraz}_1 \rangle$`,` $\langle \textit{relační-výraz}_2 \rangle$`, ...}`

SQL:

```
SELECT * FROM ⟨jméno₁⟩
  INTERSECT
SELECT * FROM ⟨jméno₂⟩
```

Množinové operace: Sjednocení relací

Definice (sjednocení relací, angl.: *union*)

Pro dvě relace $\mathcal{D}_1$ a $\mathcal{D}_2$ na relačním schématu $R \subseteq Y$ zavádíme

$$\mathcal{D}_1 \cup \mathcal{D}_2 = \{r \in \textstyle\prod_{y \in R} D_y \,|\, r \in \mathcal{D}_1 \text{ a/nebo } r \in \mathcal{D}_2\}.$$

Relace $\mathcal{D}_1 \cup \mathcal{D}_2$ na schématu R se nazývá **sjednocení relací** $\mathcal{D}_1$ a $\mathcal{D}_2$.

Tutorial D:

$\langle \textit{relační-výraz}_1 \rangle$ `UNION` $\langle \textit{relační-výraz}_2 \rangle$

`UNION` {$\langle \textit{relační-výraz}_1 \rangle$, $\langle \textit{relační-výraz}_2 \rangle$, ...}

SQL:

```
SELECT * FROM ⟨jméno₁⟩
  UNION
SELECT * FROM ⟨jméno₂⟩
```

Příklad (SQL: Kvalifikátory ALL a DISTINCT)

```
CREATE TABLE foo (x NUMERIC NOT NULL PRIMARY KEY);
CREATE TABLE bar (x NUMERIC NOT NULL PRIMARY KEY);

INSERT INTO foo VALUES (10);
INSERT INTO foo VALUES (20);
INSERT INTO foo VALUES (30);

INSERT INTO bar VALUES (20);
INSERT INTO bar VALUES (40);

/* implicit qualifier DISTINCT */
SELECT * FROM foo UNION SELECT * FROM bar;
SELECT * FROM foo UNION DISTINCT SELECT * FROM bar;

/* qualifier ALL: value 20 appears multiple times */
SELECT * FROM foo UNION ALL SELECT * FROM bar;
```

Věta (Základní vlastnosti $\cap$ a $\cup$)

Operace $\cap$ a $\cup$ s relacemi na schématu R mají následující vlastnosti:

1. *$\cap$ a $\cup$ jsou asociativní, komutativní, idempotentní a isotonní;*
2. *$\emptyset_R$ (prázdná relace na R) je neutrální prvek operace $\cup$ a anihilátor operace $\cap$;*
3. *$\cap$ a $\cup$ jsou vzájemně distributivní, to jest*
   $\mathcal{D}_1 \cup (\mathcal{D}_2 \cap \mathcal{D}_3) = (\mathcal{D}_1 \cup \mathcal{D}_2) \cap (\mathcal{D}_1 \cup \mathcal{D}_3)$,
   $\mathcal{D}_1 \cap (\mathcal{D}_2 \cup \mathcal{D}_3) = (\mathcal{D}_1 \cap \mathcal{D}_2) \cup (\mathcal{D}_1 \cap \mathcal{D}_3)$;
4. *$\cap$ a $\cup$ jsou vzájemně absorbují, to jest*
   $\mathcal{D}_1 = \mathcal{D}_1 \cap (\mathcal{D}_1 \cup \mathcal{D}_2)$,
   $\mathcal{D}_1 = \mathcal{D}_1 \cup (\mathcal{D}_1 \cap \mathcal{D}_2)$.

Důkaz.

Tvrzení 1–4 plynou z vlastností pravdivostních funkcí logických spojek „konjunkce“ a „disjunkce“. Například pro 4: Pokud $r \in \mathcal{D}_1$, pak i $r \in \mathcal{D}_1 \cup \mathcal{D}_2$ a proto $\mathcal{D}_1 \subseteq \mathcal{D}_1 \cap (\mathcal{D}_1 \cup \mathcal{D}_2)$. Opačně, pokud $r \in \mathcal{D}_1 \cap (\mathcal{D}_1 \cup \mathcal{D}_2)$, pak zřejmě $r \in \mathcal{D}_1$. $\square$

Množinové operace: Rozdíl relací

Definice (rozdíl relací, angl.: *difference/minus*)

Pro dvě relace $\mathcal{D}_1$ a $\mathcal{D}_2$ na relačním schématu $R \subseteq Y$ zavádíme

$$\mathcal{D}_1 \setminus \mathcal{D}_2 = \{r \in \textstyle\prod_{y \in R} D_y \,|\, r \in \mathcal{D}_1 \text{ a zároveň } r \notin \mathcal{D}_2\}.$$

Relace $\mathcal{D}_1 \setminus \mathcal{D}_2$ na schématu R se nazývá **rozdíl relací** $\mathcal{D}_1$ a $\mathcal{D}_2$.

Tutorial D:

$\langle \textit{relační-výraz}_1 \rangle$ `MINUS` $\langle \textit{relační-výraz}_2 \rangle$

SQL:

```
SELECT * FROM ⟨jméno₁⟩
  EXCEPT
SELECT * FROM ⟨jméno₂⟩
```

Věta (Základní vlastnosti $\cap$, $\cup$ a $\setminus$)

Operace $\cap, \cup, \setminus$ *s relacemi na schématu* R *mají následující vlastnosti:*

1. $\setminus$ *je isotonní v prvním argumentu a antitonní v druhém argumentu;*
2. $\cup$ *a* $\setminus$ *jsou adjungované operace:* $\mathcal{D}_1 \setminus \mathcal{D}_2 \subseteq \mathcal{D}_3$ *právě tehdy, když* $\mathcal{D}_1 \subseteq \mathcal{D}_2 \cup \mathcal{D}_3$*;*
3. $\mathcal{D}_1 \subseteq \mathcal{D}_2$ *právě tehdy, když* $\mathcal{D}_1 \setminus \mathcal{D}_2 = \emptyset_R$*;*
4. $(\mathcal{D}_1 \setminus \mathcal{D}_2) \cap (\mathcal{D}_2 \setminus \mathcal{D}_1) = \emptyset_R$*;*
5. $\mathcal{D}_1 \cup \mathcal{D}_2 = \mathcal{D}_1 \cup (\mathcal{D}_2 \setminus \mathcal{D}_1)$*;*
6. $\mathcal{D}_1 \cap \mathcal{D}_2 = \mathcal{D}_2 \setminus (\mathcal{D}_2 \setminus \mathcal{D}_1)$*;*
7. *pokud* $\mathcal{D}_1 \subseteq \mathcal{D}_2$*, pak* $\mathcal{D}_1 = \mathcal{D}_2 \setminus (\mathcal{D}_2 \setminus \mathcal{D}_1)$*;*
8. $\setminus$ *je distributivní (zleva/zprava) vzhledem k* $\cup$ *a* $\cap$ *následovně:*
   $(\mathcal{D}_1 \cup \mathcal{D}_2) \setminus \mathcal{D}_3 = (\mathcal{D}_1 \setminus \mathcal{D}_3) \cup (\mathcal{D}_2 \setminus \mathcal{D}_3)$,
   $(\mathcal{D}_1 \cap \mathcal{D}_2) \setminus \mathcal{D}_3 = (\mathcal{D}_1 \setminus \mathcal{D}_3) \cap (\mathcal{D}_2 \setminus \mathcal{D}_3)$,
   $\mathcal{D}_1 \setminus (\mathcal{D}_2 \cap \mathcal{D}_3) = (\mathcal{D}_1 \setminus \mathcal{D}_2) \cup (\mathcal{D}_1 \setminus \mathcal{D}_3)$,
   $\mathcal{D}_1 \setminus (\mathcal{D}_2 \cup \mathcal{D}_3) = (\mathcal{D}_1 \setminus \mathcal{D}_2) \cap (\mathcal{D}_1 \setminus \mathcal{D}_3)$.

Důkaz.

1, 4, 5 jsou zřejmé; 3 plyne z 2 pro $\mathcal{D}_3 = \emptyset_R$; 7 je speciální případ 6; zbývá tedy ukázat 2, 6 a 8:

Pro dokázání 2 nejprve předpokládejmě, že $\mathcal{D}_1 \setminus \mathcal{D}_2 \subseteq \mathcal{D}_3$ a vezměme $r \in \mathcal{D}_1$. Stačí ukázat, že pokud $r \notin \mathcal{D}_2$, pak $r \in \mathcal{D}_3$; to je ale pravda, protože z $r \notin \mathcal{D}_2$ a $\mathcal{D}_1 \setminus \mathcal{D}_2 \subseteq \mathcal{D}_3$ máme $r \in \mathcal{D}_3$. Obráceně: nechť platí $\mathcal{D}_1 \subseteq \mathcal{D}_2 \cup \mathcal{D}_3$ a vezmeme $r \in \mathcal{D}_1 \setminus \mathcal{D}_2$. To jest, $r \in \mathcal{D}_1$ a $r \notin \mathcal{D}_2$ a z $\mathcal{D}_1 \subseteq \mathcal{D}_2 \cup \mathcal{D}_3$ dostáváme $r \in \mathcal{D}_3$.

Pokud $r \in \mathcal{D}_1 \cap \mathcal{D}_2$, pak zřejmě $r \notin \mathcal{D}_2 \setminus \mathcal{D}_1$. Tím pádem ale $r \in \mathcal{D}_2 \setminus (\mathcal{D}_2 \setminus \mathcal{D}_1)$, protože $r \in \mathcal{D}_2$. Obráceně, pokud $r \in \mathcal{D}_2 \setminus (\mathcal{D}_2 \setminus \mathcal{D}_1)$, pak $r \in \mathcal{D}_2$ a $r \notin \mathcal{D}_2 \setminus \mathcal{D}_1$. Tím pádem ale i $r \in \mathcal{D}_1$, což dohromady ukazuje 6.

Prokážeme první případ 8, ostatní jsou analogické. Předpokládejme, že $r \in (\mathcal{D}_1 \cup \mathcal{D}_2) \setminus \mathcal{D}_3$, to jest platí, že $r \notin \mathcal{D}_3$. Rozlišíme dva případy: buď $r \in \mathcal{D}_1$ nebo $r \in \mathcal{D}_2$. V prvním případě platí, že $r \in \mathcal{D}_1 \setminus \mathcal{D}_3$ a tím spíš $r \in (\mathcal{D}_1 \setminus \mathcal{D}_3) \cup (\mathcal{D}_2 \setminus \mathcal{D}_3)$. Druhý případ je analogický. Opačně, předpokládejme, že $r \in (\mathcal{D}_1 \setminus \mathcal{D}_3) \cup (\mathcal{D}_2 \setminus \mathcal{D}_3)$, pokud $r \in \mathcal{D}_1 \setminus \mathcal{D}_3$, tvrzení plyne z isotonie $\setminus$ v prvním argumentu. Analogicky pro případ, kdy $r \in \mathcal{D}_2 \setminus \mathcal{D}_3$. $\square$

Odvozené množinové operace

otázky související s množinovými operacemi:

- *Které operace vzít jako základní a které jako odvozené?*
- *Jak zavést obecný koncept (booleovské) operace s relacemi?*

Definice (obecná booleovská operace, angl.: *Boolean operation*)

Mějme relace $\mathcal{D}_1, \ldots, \mathcal{D}_n$ a $\mathcal{D}$ nad relačním schématem R takové, že platí $\mathcal{D}_i \subseteq \mathcal{D}$ pro každé $i = 1, \ldots, n$. Dál uvažujme výrokovou formuli φ, která obsahuje nejvýš výrokové symboly $p_1, \ldots, p_n$. Pak $\mathrm{Bool}(\mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{D}, \varphi)$ je definovaná

$$\mathrm{Bool}(\mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{D}, \varphi) = \{r \in \mathcal{D} \,|\, e_r(\varphi) = 1\},$$

kde e_r je ohodnocení výrokových symbolů (jednoznačně rozšířené na všechny výrokové formule), splňující

$$e_r(p_i) = \begin{cases} 1, & \text{pokud } r \in \mathcal{D}_i, \\ 0, & \text{jinak.} \end{cases}$$

Příklad (Příklady obecných množinových operací)

průnik, sjednocení a rozdíl:

$$\mathcal{D}_1 \cap \mathcal{D}_2 = \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, p_1 \curlywedge p_2)$$

$$\mathcal{D}_1 \cup \mathcal{D}_2 = \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, p_1 \curlyvee p_2)$$

$$\begin{aligned} \mathcal{D}_1 \setminus \mathcal{D}_2 &= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, p_1 \curlywedge \neg p_2) \\ &= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, \neg(p_1 \Rightarrow p_2)) \end{aligned}$$

příklady dalších operací:

$$\begin{aligned} \mathcal{D}_1 \bigtriangleup \mathcal{D}_2 &= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, (p_1 \curlywedge \neg p_2) \curlyvee (\neg p_1 \curlywedge p_2)) \\ &= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, \neg(p_1 \Leftrightarrow p_2)) \end{aligned}$$

$$\mathcal{D}_1 \rightarrow_{\mathcal{D}} \mathcal{D}_2 = \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}, p_1 \Rightarrow p_2)$$

Věta (Vyjádření obecných množinových operací)

Pro každou výrokovou formuli φ *lze* $\mathrm{Bool}(\mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{D}, \varphi)$ *vyjádřit pouze pomocí relací* $\mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{D}$ *a relačního rozdílu* $\setminus$.

Důkaz.

Vezměme libovolnou výrokovou formuli φ, která obsahuje nejvýš výrokové symboly $p_1, \ldots, p_n$. Dále zavedeme následující logické spojky: nulární spojka $\mathbb{1}$ (konstanta pro pravdivostní hodnotu „pravda“) a binární spojky $\setminus\!\!\setminus$ (abjunkce). Pro každé ohodnocení e položíme:

$$e(\mathbb{1}) = 1, \qquad\qquad e(\varphi \setminus\!\!\setminus \psi) = \begin{cases} 1, & \text{pokud } e(\varphi) = 1 \text{ a } e(\psi) = 0, \\ 0, & \text{jinak.} \end{cases}$$

Zřejmě $e(\varphi \Rightarrow \psi) = e(\mathbb{1} \setminus\!\!\setminus (\varphi \setminus\!\!\setminus \psi))$ a $e(\neg\varphi) = e(\mathbb{1} \setminus\!\!\setminus \varphi)$. To jest, ke každé výrokové formuli φ existuje formule $\varphi^\bullet$ obsahující pouze spojky $\mathbb{1}$ a $\setminus\!\!\setminus$, která je sémanticky ekvivalentní s φ. Hledané vyjádření $\mathrm{Bool}(\mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{D}, \varphi)$ získáme z $\varphi^\bullet$ tím, že $\setminus\!\!\setminus$ nahradíme $\setminus$, $\mathbb{1}$ nahradíme $\mathcal{D}$ a každý výrokový symbol p_i nahradíme $\mathcal{D}_i$. $\square$

Příklad (Vyjádření základních binárních logických spojek)

$\neg(p \Rightarrow p)$	$\neg(p \Rightarrow p)$	$p \setminus p$
$p \wedge q$	$\neg(p \Rightarrow \neg q)$	$p \setminus (p \setminus q)$
$\neg(p \Rightarrow q)$	$\neg(p \Rightarrow q)$	$p \setminus q$
p	p	p
$\neg(q \Rightarrow p)$	$\neg(q \Rightarrow p)$	$q \setminus p$
q	q	q
$\neg(p \Leftrightarrow q)$	$(p \Rightarrow q) \Rightarrow \neg(q \Rightarrow p)$	$\mathbb{1} \setminus ((\mathbb{1} \setminus (p \setminus q)) \setminus (q \setminus p))$
$p \vee q$	$\neg p \Rightarrow q$	$\mathbb{1} \setminus ((\mathbb{1} \setminus p) \setminus q)$
$\neg(p \vee q)$	$\neg(\neg p \Rightarrow q)$	$(\mathbb{1} \setminus p) \setminus q$
$p \Leftrightarrow q$	$\neg((p \Rightarrow q) \Rightarrow \neg(q \Rightarrow p))$	$(\mathbb{1} \setminus (p \setminus q)) \setminus (q \setminus p)$
$\neg q$	$\neg q$	$\mathbb{1} \setminus q$
$q \Rightarrow p$	$q \Rightarrow p$	$\mathbb{1} \setminus (q \setminus p)$
$\neg p$	$\neg p$	$\mathbb{1} \setminus p$
$p \Rightarrow q$	$p \Rightarrow q$	$\mathbb{1} \setminus (p \setminus q)$
$\neg(p \wedge q)$	$p \Rightarrow \neg q$	$\mathbb{1} \setminus (p \setminus (p \setminus q))$
$p \Rightarrow p$	$p \Rightarrow p$	$\mathbb{1}$

Příklad (Tutorial D: Další množinové operace)

```
VAR foo BASE RELATION {x INTEGER}
  INIT (RELATION {TUPLE {x 10}, TUPLE {x 20}, TUPLE {x 30}})
  KEY {x};

/* union vs. union of disjoint relations */
foo UNION RELATION {TUPLE {x 20}}      ⟼ ···
foo D_UNION RELATION {TUPLE {x 20}}  /* error */

/* difference vs. included difference */
foo MINUS RELATION {TUPLE {x 40}}      ⟼ ···
foo I_MINUS RELATION {TUPLE {x 40}}  /* error */

/* symmetric difference */
foo XUNION RELATION {TUPLE {x 40}, TUPLE {x 20}}
  ⟼ RELATION {TUPLE {x 10}, TUPLE {x 30}, TUPLE {x 40}}
```

Intermezzo: Operace s n-ticemi

sjednocení (zřetězení) n-tic, angl.: *concatenation/union*

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) n-tice, angl.: *projection*

Mějme n-tici $r \in \prod_{y \in R} D_y$ a pak $S \subseteq R$ 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 ($rs = sr$), asociativní ($r(st) = (rs)t$), idempotentní ($rr = r$), neutrální vzhledem k $\emptyset$ ($r\emptyset = \emptyset r = r$)
- sjednocení n-tic $r \cup s$ je množinově-teoretické sjednocení zobrazení, odtud:

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

Příklad (Tutorial D: Sjednocení a projekce n-tic)

```
TUPLE {x 10, y 20} UNION TUPLE {z 30, a "foo"}
  ⟾ TUPLE {x 10, y 20, z 30, a "foo"}

TUPLE {x 10, y 20} UNION TUPLE {z 30, y 20}
  ⟾ TUPLE {x 10, y 20, z 30}

TUPLE {x 10, y 20} UNION TUPLE {z 30, y 666}  /* error */

TUPLE {w 0, x 10, y 20, z 30} {x, z}
  ⟾ TUPLE {x 10, z 30}

TUPLE {w 0, x 10, y 20, z 30} {ALL BUT x, z}
  ⟾ TUPLE {w 0, y 20}

TUPLE {w 0, x 10, y 20, z 30} {}
  ⟾ TUPLE {}
```

Projekce

Definice (projekce, angl.: *projection*)

Mějme relaci $\mathcal{D}$ na schématu R. Pri libovolné $S \subseteq R$ položíme:

$$\pi_S(\mathcal{D}) = \{s \in \textstyle\prod_{y \in S} D_y \,|\, \text{existuje } t \in \prod_{y \in R \setminus S} D_y \text{ tak, že } st \in \mathcal{D}\}.$$

Relace $\pi_S(\mathcal{D})$ se nazývá **projekce $\mathcal{D}$ na schéma S**.

Tutorial D:

$\langle \textit{relační-výraz} \rangle$ `{`$\langle \textit{atribut}_1 \rangle, \ldots, \langle \textit{atribut}_n \rangle$`}`

$\langle \textit{relační-výraz} \rangle$ `{ALL BUT` $\langle \textit{atribut}_1 \rangle, \ldots, \langle \textit{atribut}_n \rangle$`}`

SQL:

`SELECT DISTINCT` $\langle \textit{atribut}_1 \rangle, \ldots, \langle \textit{atribut}_n \rangle$ `FROM` $\langle \textit{jméno} \rangle$

Příklad (SQL: Kvalifikátory ALL a DISTINCT)

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

INSERT INTO foo VALUES (10, 20, 30);
INSERT INTO foo VALUES (10, 30, 30);
INSERT INTO foo VALUES (20, 40, 60);
```

x	y	z
10	20	30
10	30	30
20	40	60

```
/* explicit qualifier DISTINCT */
SELECT DISTINCT x, z FROM foo;

/* implicit qualifier ALL, nonrelational operation */
SELECT x, z FROM foo;
SELECT ALL x, z FROM foo;
```

Věta (Základní vlastnosti projekce)

Pro relaci $\mathcal{D}$ na schématu R platí:

1. $\pi_R(\mathcal{D}) = \mathcal{D}$;
2. $\pi_\emptyset(\mathcal{D}) = \begin{cases} \emptyset, & \textit{pokud } \mathcal{D} = \emptyset_R, \\ \{\emptyset\}, & \textit{jinak}; \end{cases}$
3. $\pi_S(\mathcal{D}) = \{r(S) \,|\, r \in \mathcal{D}\}$;
4. $\pi_{S_1}(\pi_{S_2}(\mathcal{D})) = \pi_{S_1}(\mathcal{D})$ *pro každé* $S_1 \subseteq S_2 \subseteq R$;
5. $\pi_S(\mathcal{D}_1 \cup \mathcal{D}_2) = \pi_S(\mathcal{D}_1) \cup \pi_S(\mathcal{D}_2)$.

Důkaz.

1 je zřejmá; 2 plyne z toho, jak vypadají relace nad prázdným schématem; 3 plyne z toho, že pokud $s \in \prod_{y \in S} D_y$, pak existuje $t \in \prod_{y \in R \setminus S} D_y$ tak, že $st \in \mathcal{D}$ p. k. $s = r(S)$ pro nějakou $r \in \mathcal{D}$; pro 4 je $\pi_{S_1}(\pi_{S_2}(\mathcal{D})) = \{s_2(S_1) \,|\, s_2 \in \pi_{S_2}(\mathcal{D})\} = \{r(S_2)(S_1) \,|\, r \in \mathcal{D}\} = \{r(S_1) \,|\, r \in \mathcal{D}\}$; 5 $\pi_S(\mathcal{D}_1 \cup \mathcal{D}_2) = \{r(S) \,|\, r \in \mathcal{D}_1 \cup \mathcal{D}_2\} = \{r(S) \,|\, r \in \mathcal{D}_1\} \cup \{r(S) \,|\, r \in \mathcal{D}_2\} = \pi_S(\mathcal{D}_1) \cup \pi_S(\mathcal{D}_2)$. □

Restrikce

Definice (restrikce, angl.: *restriction*)

Mějme relaci $\mathcal{D}$ na schématu R a nechť θ je skalární výraz typu „pravdivostní hodnota“, který může obsahovat jména atributů z R. Řekneme, že $r \in \mathcal{D}$ **splňuje** (podmínku danou výrazem) θ, pokud má θ hodnotu „pravda“ za předpokladu, že jsme nahradili jména atributů v θ jejich hodnotami z r. Položíme

$$\sigma_\theta(\mathcal{D}) = \{r \in \mathcal{D} \,|\, r \text{ splňuje } \theta\}$$

Relace $\sigma_\theta(\mathcal{D})$ se nazývá **restrikce $\mathcal{D}$** splňující θ.

poznámky:

- **restrikce na rovnost** – restrikce tvaru $\sigma_{y=d}(\mathcal{D})$
- terminologie: *restrikce* = *selekce* (nezaměňovat se `SELECT` z SQL, **!!**)
- restrikce (*zmenšení velikosti* relace) $\times$ projekce (*zmenšení stupně* relace)

Restrikce v Tutorial D a SQL

Tutorial D:

$\langle \textit{relační-výraz} \rangle$ `WHERE` $\langle \textit{podmínka} \rangle$

SQL:

`SELECT * FROM` $\langle \textit{jméno} \rangle$ `WHERE` $\langle \textit{podmínka} \rangle$

Poznámka o výrazech v restrikcích

Výraz $\langle \textit{podmínka} \rangle$ chápeme jako obecný výraz, který lze formulovat v daném dotazovacím jazyku. Pro zjednodušení dalších úvah budeme předpokládat, že $\langle \textit{podmínka} \rangle$ je výraz, který se chová z hlediska své interpretace *funkcionálně* (nemá vedlejší efekty); lze jej chápat jako zobrazení $\theta\colon \prod_{y \in R} D_y \to \{0, 1\}$, kde $\theta(r) = 1$ znamená, že r splňuje θ.

důsledek: pokud je $\prod_{y \in R} D_y$ konečná, $\sigma_\theta(\mathcal{D})$ lze chápat jako průnik relací (**!!**)

Věta (Základní vlastnosti restrikce)

Pro relaci $\mathcal{D}$ na schématu R platí:

1. $\sigma_{\theta_1}(\sigma_{\theta_2}(\mathcal{D})) = \sigma_{\theta_2}(\sigma_{\theta_1}(\mathcal{D}))$
2. $\sigma_{\theta}(\sigma_{\theta}(\mathcal{D})) = \sigma_{\theta}(\mathcal{D})$

Důkaz.

Bod 1 plyne z komutativity konjunkce, konkrétně $r \in \sigma_{\theta_1}(\sigma_{\theta_2}(\mathcal{D}))$ p. k. r splňuje θ_1 a náleží do $\sigma_{\theta_2}(\mathcal{D})$ což platí p. k. r splňuje θ_1 a r splňuje θ_2 a náleží do $\mathcal{D}$, což je p. k. r splňuje θ_2 a r splňuje θ_1 a náleží do $\mathcal{D}$ (viz poznámku o funkcionálním charakteru podmínek), to jest $r \in \sigma_{\theta_2}(\sigma_{\theta_1}(\mathcal{D}))$. Bod 2 plyne analogicky z idempotence konjunkce (opět důležitý předpoklad z předchízí poznámky). $\square$

značení:

- pro podmínky $\theta_1, \ldots, \theta_n$ píšeme $\sigma_{\theta_1,\ldots,\theta_n}(\mathcal{D})$ místo $\sigma_{\theta_1}(\sigma_{\theta_2}(\cdots(\sigma_{\theta_n}(\mathcal{D}))\cdots))$
- alternativní značení: $\sigma_{\theta_1 \curlywedge \cdots \curlywedge \theta_n}(\mathcal{D})$ ($\curlywedge$ je symbol pro konjunkci)

Věta (Vztah restrikce a projekce)

Mějme relaci $\mathcal{D}$ na schématu R. Pokud jsou všechna jména atributů z výrazu θ obsažena v $S \subseteq R$, pak $\pi_S(\sigma_\theta(\mathcal{D})) = \sigma_\theta(\pi_S(\mathcal{D}))$.

Důkaz.

Platí, že $s \in \pi_S(\sigma_\theta(\mathcal{D}))$ právě tehdy, když existuje t tak, že $st \in \sigma_\theta(\mathcal{D})$. To platí p. k. existuje t tak, že $st \in \mathcal{D}$ a st splňue θ. Platnost θ však nezávisí na t, protože všechna jména atributů z θ jsou v s, takže předchozí podmínka je ekvivalentní podmínce: existuje t tak, že $st \in \mathcal{D}$ a s splňuje θ. To jest, s splňuje θ a navíc existuje t tak, že $st \in \mathcal{D}$, což znamená $s \in \sigma_\theta(\pi_S(\mathcal{D}))$. $\square$

pozor:

- pokud je v θ obsaženo jméno některého atributu, který není v $S \subseteq R$, pak $\sigma_\theta(\pi_S(\mathcal{D}))$ nedává smysl (pravá strana rovnosti předchozí věty není definovaná)
- příkaz `SELECT` v SQL chápeme tak, že *projekce následuje za restrikcí*

Věta (Vztah restrikce a množinových operací)

Pro relace $\mathcal{D}_1$ a $\mathcal{D}_2$ na stejném relačním schématu platí:

1. $\sigma_\theta(\mathcal{D}_1 \cup \mathcal{D}_2) = \sigma_\theta(\mathcal{D}_1) \cup \sigma_\theta(\mathcal{D}_2)$,
2. $\sigma_\theta(\mathcal{D}_1 \cap \mathcal{D}_2) = \sigma_\theta(\mathcal{D}_1) \cap \mathcal{D}_2 = \mathcal{D}_1 \cap \sigma_\theta(\mathcal{D}_2) = \sigma_\theta(\mathcal{D}_1) \cap \sigma_\theta(\mathcal{D}_2)$,
3. $\sigma_\theta(\mathcal{D}_1 \setminus \mathcal{D}_2) = \sigma_\theta(\mathcal{D}_1) \setminus \mathcal{D}_2 = \sigma_\theta(\mathcal{D}_1) \setminus \sigma_\theta(\mathcal{D}_2)$.

Důkaz.

Bod 1 se dokáže následovně: $r \in \sigma_\theta(\mathcal{D}_1 \cup \mathcal{D}_2)$ p. k. r splňuje θ a buď $r \in \mathcal{D}_1$ nebo $r \in \mathcal{D}_2$. To znamená, že buď r splňuje θ a $r \in \mathcal{D}_1$ nebo r splňuje θ a $r \in \mathcal{D}_2$, to jest $r \in \sigma_\theta(\mathcal{D}_1) \cup \sigma_\theta(\mathcal{D}_2)$. Bod 2 plyne analogicky z toho, že $r \in \sigma_\theta(\mathcal{D}_1 \cap \mathcal{D}_2)$ p. k. r splňuje θ a $r \in \mathcal{D}_1$ a $r \in \mathcal{D}_2$ (dále použijem idempotenci, asociativitu a komutativitu konjunkce). Analogicky dokážeme 3 z toho, že $r \in \sigma_\theta(\mathcal{D}_1 \setminus \mathcal{D}_2)$ p. k. r splňuje θ a platí, že $r \in \mathcal{D}_1$ a $r \notin \mathcal{D}_2$. $\square$

poznámka: $\sigma_\theta(\mathcal{D}_1 \setminus \mathcal{D}_2) \subseteq \mathcal{D}_1 \setminus \sigma_\theta(\mathcal{D}_2)$ ale obecně ne obráceně (**!!**)

Příklad (Tutorial D: Modifikace dat, příkaz INSERT)

```
VAR person BASE
  INIT (RELATION {
          TUPLE {name "Abbe", salary 15000, bonus 0},
          TUPLE {name "Blangis", salary 10000, bonus 0}})
  KEY {name};

INSERT person RELATION {
    TUPLE {name "Curval", salary 12000, bonus 500},
    TUPLE {name "Durcet", salary 11000, bonus 1500}};

/* equivalently using relational assignment */
person := person UNION
  RELATION {
    TUPLE {name "Curval", salary 12000, bonus 500},
    TUPLE {name "Durcet", salary 11000, bonus 1500}};
```

Příklad (Tutorial D: Modifikace dat, příkaz DELETE)

```
VAR person BASE
  INIT (RELATION {
          TUPLE {name "Abbe", salary 15000, bonus 0},
          TUPLE {name "Blangis", salary 10000, bonus 0},
          TUPLE {name "Curval", salary 12000, bonus 500},
          TUPLE {name "Durcet", salary 11000, bonus 1500}})
  KEY {name};

DELETE person WHERE salary < 12000;
DELETE person;  /* delete all tuples */

/* equivalently using relational assignment */
person := person WHERE NOT (salary < 12000);
person := person MINUS (person WHERE salary < 12000);
person := person WHERE FALSE;
```

Příklad (Tutorial D: Modifikace dat, příkaz UPDATE)

```
VAR person BASE
  INIT (RELATION {
          TUPLE {name "Abbe", salary 15000, bonus 0},
          TUPLE {name "Blangis", salary 10000, bonus 0},
          TUPLE {name "Curval", salary 12000, bonus 500},
          TUPLE {name "Durcet", salary 11000, bonus 1500}})
  KEY {name};

UPDATE person WHERE salary >= 12000: {
  salary := (salary * 120) / 100,
  bonus := bonus + 2000
};

UPDATE person: {bonus := 0};
```

poznámka: UPDATE lze také vyjádřit jako relační přiřazení (PŘEDNÁŠA 6)

Problémy fyzické vrstvy: Otázky efektivity

připomenutí:

- **logická vrstva** databázového systému:
  - abstrahuje od fyzického uložení dat
- **fyzická vrstva** databázového systému:
  - nejnižší vrstva, zabývá se fyzickým (efektivním a perzistentním) uložením dat
  - zajímavá z pohledu implementace DB systému, pro uživatele (téměř) nezajímavá

doposud: pouze úvahy o logické (a externí) vrstvě

otázky fyzické vrstvy:

- jak efektivně organizovat data na disku
- efektivní organizace z pohledu: odezvy dotazů / modifikace dat
- jaké algoritmy používat pro vyhodnocování dotazů
- jak odstínit uživatele od aspektů fyzické vrstvy (automatická indexace, ...)

Efektivní vyhodnocování relačních operací

otázka:

Jak efektivně počítat výsledky relačních operací typu sjednocení, průnik, rozdíl, projekce a restrikce?

naivní vyhodnocování:

- σ: iterace přes všechny prvky tabulky
- $\cap$, $\cup$, $\setminus$, π: iterace ve vnořené smyčce

optimalizované vyhodnocování:

- využití dodatečných struktur (indexů) umožňujících rychlé vyhledávání n-tic
- zjednodušování dotazů na základě vlastností operací (viz předchozí tvrzení)

podpora:

- SQL – explicitní podpora pro vytváření indexů
- Tutorial D – nedefinuje indexy ani jiné koncepty související s fyzickou vrstvou

Základní metody implementace indexů

dva základní typy struktur pro indexy:

1. indexy reprezentují **uspořádané množiny**:
   - organizace indexu: perzistentní B-strom nebo jeho modifikace (B^+-strom)
   - rychlé nalezení hodnot splňující podmínky s $<$, $\leq$, $=$, $\geq$, $>$
2. indexy reprezentují **asociační pole**:
   - organizace indexu: perzistentní (a rozšiřitelná) hashovací tabulka
   - rychlé nalezení hodnot splňující podmínky s $=$

další vlastnosti indexů:

- **jednosloupcové** / **vícesloupcové**
- **husté** / **řídké** (indexovány jsou první hodnoty bloků, sekvenční dohledávání)
- mohou být **unikátní** (vytvořené při definici integritního omezení typu „`UNIQUE`“)
- **úplné** / **částečné** (indexují pouze podmnožinu tabulky)

Příklad (SQL: Vytváření indexů)

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

/* creating unique multicolumn B-tree index explicitly */
CREATE UNIQUE INDEX foo_yz_idx ON foo (y, z);

DROP INDEX foo_yz_idx;

/* creating implicit index by imposing constraint */
CREATE TABLE bar (
  x NUMERIC NOT NULL PRIMARY KEY,
  y NUMERIC NOT NULL,
  z NUMERIC NOT NULL,
  UNIQUE (y, z));
```

Příklad (SQL: Vytváření indexů)

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

/* hash index */
CREATE INDEX foo_y_idx ON foo USING hash (y);

/* partial index */
CREATE INDEX foo_z_idx ON foo (z) WHERE z >= 1000;

/* index on results of expression */
CREATE INDEX foo_abs_idx ON foo (abs (y + z));

/* application in query */
SELECT * FROM foo WHERE abs (y + z) >= 1000;
```

Přednáška 3: Závěr

pojmy k zapamatování:

- relační model dat, atributy, relační schémata, typy (domény)
- typy v RM: skalární, n-ticové, relační
- kartézské součiny, relace, první normální forma
- relační proměnné, klíče a jejich realizace v jazycích Tutorial D a SQL

použité zdroje:

Date C. J.: *Database in Depth: Relational Theory for Practitioners*
O'Reilly Media 2005, ISBN 978–0596100124

Date C. J., Darwen H.: *Databases, Types and the Relational Model*
Addison Wesley 2006, ISBN 978–0321399427

Maier D: *Theory of Relational Databases*
Computer Science Press 1983, ISBN 978–0914894421