Databázové systémy

Dotazování v relačním modelu a SQL

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

Opakování

- **Relační schéma** je konečná množina $R = \{\langle y_1, \tau_1 \rangle, \ldots, \langle y_n, \tau_n \rangle\}$, kde $y_1, \ldots, y_n$ jsou vzájemně různé atributy z Y a $\tau_1, \ldots, \tau_n$ jsou typy. (Pokud typ vyplývá jednoznačně z kontextu, pak relační schéma ztotožňujeme s konečnými podmnožinami $R \subseteq Y$ a doménu (typ) atributu $y \in Y$ značíme D_y.)

- **Kartézský součin množin** A_i $(i \in I)$ je množina $\prod_{i \in I} A_i$ všech zobrazení $f : I \to \bigcup_{i \in I} A_i$ takových, že $f(i) \in A_i$ platí pro každý index $i \in I$.

- **Relace** $\mathcal{D}$ nad relačním schématem R je libovolná konečná podmnožina $\prod_{y \in R} D_y$.

- Každé zobrazení $f \in \prod_{i \in I} A_i$ se nazývá **n-tice** (angl. tuple).

- **Relační proměnná** daného typu je (perzistentní) proměnná, která může nabývat hodnot, jimiž jsou relace daného typu. (Formalizace jména tabulky v RM.)

Relační dotazování

- **Instance databáze** je konečná množina relační proměnných, jejich aktualních hodnot a integritních omezení. (Zatím nepotřebujeme.)
- **Dotaz** je částečně rekurzivní funkce z množiny všech instancí databáze do množiny všech relací (nad relačními schématy).

Poznámky

- typický přístup: dotaz je popsán v určitém *jazyku* a je dána jeho *intepretace*
- 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**
   - dotaz = výraz skládající se ze složených relačních operací
   - interpretace dotazu = postupné vyhodnocování operací
   - elementární operace s relacemi (SŘBD může dobře optimalizovat)

2. **relační kalkuly**
   - několi typů: **doménový relační kalkul**, **n-ticový relační kalkul**
   - dotaz = formule predikátové logiky (s volnými proměnnými)
   - interpretace dotazů = ohodnocení formulí ve struktuře (instanci databáze)
   - ryze deklarativní, vychází z něj řada jazyků (QUEL, částečně SQL)

Poznámka:

- databáze při zpracování dotazů vytváří **prováděcí plány**, které jsou blízké operacím relační algebry

Typy relačních operací

- Množina relačních operací přímo ovlivňuje, jak silný (expresivní) bude dotazovací jazyk, který je na ní založen.
- dělení operací relační algebry
  - základní (minimální množina operací) vs. odvozené operace
  - podle počtu operandů (operace s jednou, dvěma, ... relacemi)
  - podle významu (množinové, protějšky kvantifikátorů, ...)
- ,,Rozumná množina operací" – taková, že relační algebra je stejně silná jako (doménově nezávislý) doménový relační kalkul.

Množinové operace: Průnik relací

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

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

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

SQL:

```
SELECT * FROM tabulka1
   INTERSECT
SELECT * FROM tabulka2;
```

Množinové operace: Sjednocení relací (1/2)

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

$$\mathcal{D}_1 \cup \mathcal{D}_2 = \{r \in \prod_{y \in R} D_y | r \in \mathcal{D}_1 \text{ nebo } r \in \mathcal{D}_2\}$$

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

SQL:

```
SELECT * FROM tabulka1
   UNION
SELECT * FROM tabulka2;
```

Množinové operace: Sjednocení relací (2/2)

```
CREATE TABLE foo (x INTEGER PRIMARY KEY);
CREATE TABLE bar (x INTEGER PRIMARY KEY);

INSERT INTO foo (x) VALUES (10), (20), (30);
INSERT INTO bar (x) VALUES (30), (40);

/* implicitni modifikator DISTINCT */
SELECT * FROM foo UNION SELECT * FROM bar;
SELECT * FROM foo UNION DISTINCT SELECT * FROM bar;

/* hodnota 30 se ve vysledku objevi 2x -- nerelacni vysledek */
SELECT * FROM foo UNION ALL SELECT * FROM bar;
```

Množinové operace: rozdíl relací

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

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

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

SQL:

```
SELECT * FROM tabulka1
   EXCEPT
SELECT * FROM tabulka2;
```

Množinové operace: některé vybrané vlastnosti

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

1. $\cup$ a $\cap$ jsou asociativní, komutativní, idempotentní.
2. $\emptyset_R$ (prázdná relace na R) je neutrální prvek operace $\cup$ a anihilátor $\cap$
3. $\cup$ a $\cap$ jsou vzájemně distributivní, tj.
   $\mathcal{D}_1 \cup (\mathcal{D}_2 \cap \mathcal{D}_3) = (\mathcal{D}_1 \cup \mathcal{D}_2) \cap (\mathcal{D}_1 \cup \mathcal{D}_3)$
   $\mathcal{D}_1 \cap (\mathcal{D}_2 \cup \mathcal{D}_3) = (\mathcal{D}_1 \cap \mathcal{D}_2) \cup (\mathcal{D}_1 \cap \mathcal{D}_3)$
4. $\cup$ a $\cap$ se vzájemně absorbují, tj.
   $\mathcal{D}_1 = \mathcal{D}_1 \cap (\mathcal{D}_1 \cup \mathcal{D}_2)$
   $\mathcal{D}_1 = \mathcal{D}_1 \cup (\mathcal{D}_1 \cap \mathcal{D}_2)$
5. $\mathcal{D}_1 \subseteq \mathcal{D}_2$ právě tehdy, když $\mathcal{D}_1 \setminus \mathcal{D}_2 = \emptyset_R$
6. $\mathcal{D}_1 \cup \mathcal{D}_2 = \mathcal{D}_1 \cup (\mathcal{D}_2 \setminus \mathcal{D}_1)$
7. $\mathcal{D}_1 \cap \mathcal{D}_2 = \mathcal{D}_2 \setminus (\mathcal{D}_2 \setminus \mathcal{D}_1)$
8. ...

Obecná booleovská operace (1/2)

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

Obecná (booleovská) operace
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 nejvýše výrokové symboly $p_1, \ldots, p_n$. Pak

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

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

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

Obecná booleovská operace (2/2)

Příklad

$$\begin{aligned}
\mathcal{D}_1 \cap \mathcal{D}_2 &= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, p_1 \wedge p_2) \\
\mathcal{D}_1 \cup \mathcal{D}_2 &= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, p_1 \vee p_2) \\
\mathcal{D}_1 \backslash \mathcal{D}_2 &= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, p_1 \wedge \neg p_2) \\
&= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, \neg(p_1 \Rightarrow p_2))
\end{aligned}$$

další operace

$$\begin{aligned}
\mathcal{D}_1 \triangle \mathcal{D}_2 &= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, (p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2)) \\
&= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_1 \cup \mathcal{D}_2, \neg(p_1 \Leftrightarrow p_2)) \\
\mathcal{D}_1 \rightarrow_{\mathcal{D}} \mathcal{D}_2 &= \mathrm{Bool}(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}, p_1 \Rightarrow p_2)
\end{aligned}$$

Operace s n-ticemi

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:

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

Projekce

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

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

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

SQL:

```
SELECT DISTINCT atribut1, atribut2 FROM tabulka1;
```

Projekce (SQL: kvalifikátory ALL a DISTINCT)

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

INSERT INTO foo (x, y, z) VALUES (10, 20, 30);
INSERT INTO foo (x, y, z) VALUES (10, 30, 30);
INSERT INTO foo (x, y, z) VALUES (20, 40, 60);

/* explicitni kvalifikator DISTINCT */
SELECT DISTINCT x, z FROM foo;

/* implicitni kvalifikator ALL, nerelacni vysledek */
SELECT x, z FROM foo;
SELECT ALL x, z FROM foo;
```

Příklad: pracovní data (1/2)

```
CREATE TABLE zamestnanci
(jmeno VARCHAR(30) PRIMARY KEY,
 vek INTEGER NOT NULL,
 plat NUMERIC(8, 2) NOT NULL CHECK (plat > 8000),
 uvazek NUMERIC(3,2) DEFAULT 1 NOT NULL CHECK ((uvazek > 0) AND (uvazek <= 2)),
 funkce VARCHAR(20) NOT NULL);

INSERT INTO zamestnanci (jmeno, vek, plat, uvazek, funkce) VALUES
('Alice', 40, 16789.20, 0.80, 'Manager'),
('Bob', 37, 23100.00, 0.50, 'Udrzbar'),
('Chuck', 20, 15000.00, 1.00, 'Kreativec'),
('David', 30, 100000.00, 1.00, 'CEO'),
('Eva', 17, 10000.00, 1.20, 'Manager'),
('Frank', 35, 24000.00, 1.00, 'IT'),
('Gerald', 60, 17000.00, 1.00, 'Udrzbar'),
('Hugh', 45, 42000.00, 0.70, 'IT'),
('Ida', 32, 23000, 1.0, 'Manager');
```

Příklad: pracovní data (2/2)

```
SELECT * FROM zamestnanci;
```

jmeno	vek	plat	uvazek	funkce
Alice	40	16789.20	0.80	Manager
Bob	37	23100.00	0.50	Udrzbar
Chuck	20	15000.00	1.00	Kreativec
David	30	100000.00	1.00	CEO
Eva	17	10000.00	1.20	Manager
Frank	35	24000.00	1.00	IT
Gerald	60	17000.00	1.00	Udrzbar
Hugh	45	42000.00	0.70	IT
Ida	32	23000.00	1.00	Manager

Příklad: projekce

```
SELECT funkce FROM zamestnanci;
```

```
  funkce
-----------
 Manager
 Udrzbar
 Kreativec
 CEO
 Manager
 IT
 Udrzbar
 IT
 Manager
```

```
SELECT DISTINCT funkce FROM zamestnanci;
```

```
  funkce
-----------
 IT
 Manager
 Udrzbar
 Kreativec
 CEO
```

Vlastnosti projekce

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

1. $\pi_R(\mathcal{D}) = \mathcal{D}$
2. $\pi_\emptyset(\mathcal{D}) \begin{cases} \emptyset, & \text{pokud } \mathcal{D} = \emptyset_R, \\ \{\emptyset\}, & \text{jinak.} \end{cases}$
3. $\pi_{S_1}(\pi_{S_2}(\mathcal{D})) = \pi_{S_1}(\mathcal{D})$ pro každé $S_1 \subseteq S_2 \subseteq R$
4. $\pi_S(\mathcal{D}_1 \cup \mathcal{D}_2) = \pi_S(\mathcal{D}_1) \cup \pi_S(\mathcal{D}_2)$

Restrikce

Mějme relaci $\mathcal{D}$ na schématu R a nechť θ je 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 atribůtu v θ jejich hodnotami z r.

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

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

SQL:

```
SELECT * FROM tabulka WHERE podminka;
```

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)

Příklad: restrikce

```
SELECT * FROM zamestnanci WHERE funkce = 'Manager';
```

```
 jmeno | vek |   plat   | uvazek | funkce
-------+-----+----------+--------+---------
 Alice |  40 | 16789.20 |   0.80 | Manager
 Eva   |  17 | 10000.00 |   1.20 | Manager
 Ida   |  32 | 23000.00 |   1.00 | Manager
```

```
SELECT jmeno, vek FROM zamestnanci WHERE funkce = 'Manager';
```

```
 jmeno | vek
-------+-----
 Alice |  40
 Eva   |  17
 Ida   |  32
```

Vlastnosti restrikce

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

- $\sigma_{\theta_1}(\sigma_{\theta_2}(\mathcal{D})) = \sigma_{\theta_2}(\sigma_{\theta_1}(\mathcal{D}))$
- $\sigma_{\theta}(\sigma_{\theta}(\mathcal{D})) = \sigma_{\theta}(\mathcal{D})$

Restrikce a projekce

- Mějme relaci $\mathcal{D}$ na schématu R. Pokud jsou všechna jména atributů z výrazu θ obsažena v $S \subseteq R$, pak $\pi_S(\sigma_\theta(\mathcal{D})) = \sigma_\theta(\pi_S(\mathcal{D}))$.
- Pozor: Pokud je v θ obsaženo jméno některého atributu, který není v $S \subseteq R$, pak výraz $\sigma_\theta(\pi_S(\mathcal{D}))$ nedává smysl.
- V případě SQL chápeme, že projekce následuje za restrikcí.

Poznámka

- Podmínku θ lze chápat jako zobrazení $\theta : \prod_{y \in R} D_y \to \{0, 1\}$, kde $\theta(r) = 1$ znamená, že r splňuje θ.
- Pokud je $\prod_{y \in R} D_y$ konečná, lze $\sigma_\theta(\mathcal{D})$ chápat jako průnik relací.

Přejmenování

Přejmenování n-tice
Mějme n-tici $r \in \prod_{y \in R} D_y$ a injektivní zobrazení $h : R \to Y$. Pak složené zobrazení $\rho_h(r) = h^{-1} \circ r$ nazveme přejmenování n-tice r podle h.
$\rho_h(r)$ je n-tice nad schématem $h(R)$, kde $(\rho_h(r))(h(y)) = r(y)$, pro každé $y \in R$.

Přejmenování relace
Mějme relaci $\mathcal{D}$ na schématu R a injektivní zobrazení $h : R \to Y$. Položíme:

$$\rho_h(\mathcal{D}) = \{\rho_h(r) | r \in \mathcal{D}\}.$$

Relace $\rho_h(\mathcal{D})$ se nazývá **přejmenování** $\mathcal{D}$ **podle** h. Pro jednoduchost se někdy značí $\rho_{y'_1 \leftarrow y_1, \ldots, y'_n \leftarrow y_n}(\mathcal{D})$ pokud $h(y_i) = y'_i$ $(i = 1, \ldots, n)$ a $h(y) = y$ jinak.

SQL:

```
SELECT stare_jmeno AS nove_jmeno FROM tabulka;
SELECT stare_jmeno nove_jmeno FROM tabulka;
```

Příklad: přejmenování

```
SELECT jmeno AS krestni_jmeno, vek, plat, uvazek, funkce FROM zamestnanci;
```

krestni_jmeno	vek	plat	uvazek	funkce
Alice	40	16789.20	0.80	Manager
Bob	37	23100.00	0.50	Udrzbar
Chuck	20	15000.00	1.00	Kreativec
David	30	100000.00	1.00	CEO
Eva	17	10000.00	1.20	Manager
Frank	35	24000.00	1.00	IT
Gerald	60	17000.00	1.00	Udrzbar
Hugh	45	42000.00	0.70	IT
Ida	32	23000.00	1.00	Manager

Rozšíření

Mějme relaci $\mathcal{D}$ na schématu R a uvažujme po dvou různé atributy $y_1, \ldots, y_n$, které nejsou v R. Dále uvažujme výrazy $\theta_1, \ldots \theta_n$, které mohou obsahovat jména atributů z R. Pro každou n-tici $r \in \mathcal{D}$ označíme r^{θ_i} hodnotu výrazu θ_i za předpokladu, že byly výskyty atributů $y \in R$ v θ_i nahrazeny hodnotami $r(y)$. Dále položíme:

$$\epsilon_{y_1 \leftarrow \theta_1, \ldots, y_n \leftarrow \theta_n}(\mathcal{D}) = \{r \cup \{\langle y_1, r^{\theta_1} \rangle, \ldots, \langle y_n, r^{\theta_n} \} | r \in \mathcal{D}\}.$$

Relace $\epsilon_{y_1 \leftarrow \theta_1, \ldots, y_n \leftarrow \theta_n}(\mathcal{D})$ se nazývá **rozšíření** $\mathcal{D}$ o atributy $y_1, \ldots, y_n$.

SQL:

```
SELECT *, 20 AS y FROM tabulka;
SELECT *, x + 20 AS y FROM tabulka;
SELECT *, x + 20 AS y, x * 20 AS z FROM tabulka;
```

Příklad: rozšíření

```
SELECT *, plat * 0.19 AS dan FROM zamestnanci;
```

jmeno	vek	plat	uvazek	funkce	dan
Alice	40	16789.20	0.80	Manager	3189.9480
Bob	37	23100.00	0.50	Udrzbar	4389.0000
Chuck	20	15000.00	1.00	Kreativec	2850.0000
David	30	100000.00	1.00	CEO	19000.0000
Eva	17	10000.00	1.20	Manager	1900.0000
Frank	35	24000.00	1.00	IT	4560.0000
Gerald	60	17000.00	1.00	Udrzbar	3230.0000
Hugh	45	42000.00	0.70	IT	7980.0000
Ida	32	23000.00	1.00	Manager	4370.0000

Agregace a sumarizace (1/2)

- **aggregace** – jediná hodnota stanovená ze všech hodnot atributů dané relace
- **sumarizace** – výsledkem je relace obsahující hodnoty vypočtené z (částí) hodnot (některých) atributů v dané relaci

Agregační funkce

- `AVG()` – aritmetický průměr
- `SUM()` – součet
- `COUNT()` – počet hodnot
- `COUNT(DISTINCT )` – počet unikátních hodnot
- `MIN()` – minimum
- `MAX()` – maximum

Agregace a sumarizace (2/2)

```
SELECT COUNT(*) AS cnt FROM tabulka;
SELECT COUNT(name) AS cnt FROM tabulka;
SELECT COUNT(DISTINCT name) AS cnt FROM tabulka;
SELECT MAX(x) AS y FROM tabulka;
SELECT AVG(x) AS y, SUM(x) AS z FROM tabulka;
```

Sumarizace

- klauzule `GROUP BY` umožňuje určit, podle kterých atributů proběhne sumarizace
- na výsledky sumarizace je možné použít restrikci klauzulí `HAVING`

```
SELECT funkce, MAX(plat) AS nejvyssi_plat FROM zamestnanci
   GROUP BY funkce;

SELECT funkce, MAX(plat) FROM zamestnanci
   GROUP BY funkce
   HAVING MAX(plat) > 20000;
```

Příklad: agregace a sumarizace

```
SELECT COUNT(*) AS zamestnancu, COUNT(DISTINCT funkce) AS profesi,
       AVG(plat) prumerny_plat FROM zamestnanci;
```

```
 zamestnancu | profesi |   prumerny_plat
-------------+---------+--------------------
           9 |       5 | 30098.800000000000
```

```
SELECT funkce, COUNT(*) AS zamestnancu, AVG(plat) prumerny_plat
   FROM zamestnanci GROUP BY funkce;
```

```
  funkce    | zamestnancu |      prumerny_plat
------------+-------------+-------------------------
 IT         |           2 |     33000.000000000000
 Manager    |           3 |     16596.400000000000
 Udrzbar    |           2 |     20050.000000000000
 Kreativec  |           1 | 15000.0000000000000000
 CEO        |           1 |    100000.000000000000
```

SQL: nerelační operace

Řazení výsledků

- klauzule `ORDER BY atribut1, atribut2, ...`
- za každým atributem je možné uvést `ASC` nebo `DESC` podle toho, zda se mají hodnoty řadit vzestupně, či sestupně
- operace nemá oporu v RM

Omezení velikosti výsledků

- řada SŘDB umožňuje omezit množství vrácených řádků
- implementace/řešení se často liší podle konkrétní SŘBD
- nutné používat společně s `ORDER BY` (jinak nemá smysl, jelikož pořadí řádků není nikde definováno!!!)
- typicky klauzule `LIMIT n` vracející prvních `n` řádků
- typicky klauzule `OFFSET n` ignorujících prvních `n` řádků

SQL: nerelační operace (1/2)

```
SELECT * FROM tabulka ORDER BY x, y;
SELECT * FROM tabulka ORDER BY x ASC, b;
SELECT * FROM tabulka ORDER BY x ASC, b DESC;

SELECT * FROM tabulka ORDER BY x LIMIT 10;
SELECT * FROM tabulka ORDER BY x OFFSET 5 LIMIT 10;

SELECT * FROM zamestnanci ORDER BY plat LIMIT 5;
```

jmeno	vek	plat	uvazek	funkce
Eva	17	10000.00	1.20	Manager
Chuck	20	15000.00	1.00	Kreativec
Alice	40	16789.20	0.80	Manager
Gerald	60	17000.00	1.00	Udrzbar
Ida	32	23000.00	1.00	Manager

SQL: Úplný dotaz

```
SELECT atribut1, atribut2, ... FROM tabulka
   WHERE podminka
   GROUP BY atributy
   HAVING podminka
   ORDER BY sloupce
   OFFSET radek
   LIMIT pocet
```

1. pro každý řádek jsou vyhodnoceny výrazy (část hned za `SELECT`)
2. jsou vybrány řádky splňující podmínku `WHERE`
3. dojde k sumarizaci hodnot pomocí `GROUP BY`
4. jsou vybrány řádky splňující podmínku `HAVING`
5. řádky jsou setříděny podle `ORDER BY`
6. jsou odfiltrovany řádky podle `OFFSET`
7. je vybrán počet řádků daný zapomocí `LIMIT`

Vnořené dotazy

vnořený dotaz (dotaz musí vracet sloupec)

```
SELECT * FROM tabulka WHERE atribut IN (dotaz);
```

vnořený dotaz (dotaz musí vracet singleton)

```
SELECT * FROM tabulka WHERE atribut = (dotaz);
SELECT * FROM tabulka WHERE atribut > (dotaz);
```

vnořený dotaz (existenční a kvantifikační podmínky)

```
SELECT * FROM tabulka WHERE atribut <= SOME (dotaz);
SELECT * FROM tabulka WHERE atribut <= ALL (dotaz);
```

vnořený dotaz na úrovni atributu

```
SELECT *, (dotaz) AS novy_atribut FROM tabulka;
```

Vnořené dotazy: příklady (1/2)

```
SELECT * FROM zamestnanci
   WHERE jmeno IN (SELECT jmeno FROM zamestnanci WHERE funkce = 'IT');

SELECT * FROM zamestnanci
   WHERE plat > (SELECT AVG(plat) FROM zamestnanci);

SELECT * FROM zamestnanci
   WHERE plat >= SOME (SELECT plat FROM zamestnanci
                                   WHERE funkce = 'Udrzbar');

SELECT * FROM zamestnanci
   WHERE plat >= ALL (SELECT plat FROM zamestnanci
                                   WHERE funkce = 'Udrzbar');
```

Vnořené dotazy: příklady (2/2)

```
SELECT *, (SELECT COUNT(*) FROM zamestnanci AS z1
                  WHERE z1.plat > z2.plat)
          AS rank
   FROM zamestnanci AS z2
   ORDER BY rank;
```

jmeno	vek	plat	uvazek	funkce	rank
David	30	100000.00	1.00	CEO	0
Hugh	45	42000.00	0.70	IT	1
Frank	35	24000.00	1.00	IT	2
Bob	37	23100.00	0.50	Udrzbar	3
Ida	32	23000.00	1.00	Manager	4
Gerald	60	17000.00	1.00	Udrzbar	5
Alice	40	16789.20	0.80	Manager	6
Chuck	20	15000.00	1.00	Kreativec	7
Eva	17	10000.00	1.20	Manager	8