Databázové systémy

Pohledy, rozšíření, slučování a rozdělování

Vilém Vychodil

KMI/DATA1, Přednáška 6

Databázové systémy

Přednáška 6: Přehled

1. Relační proměnné:
   - základní proměnné,
   - pohledy,
   - rekurzivní pohledy,
   - modifikovatelné pohledy,
   - materializované pohledy.
2. Další relační operace:
   - vyjadřování nových skalárních a relačních hodnot,
   - rozšíření relací.
3. Relační operace specifické pro Tutorial D:
   - operace typu `WRAP` / `UNWRAP`,
   - operace typu `GROUP` / `UNGROUP`,
   - role operací jako substituentů za vnější spojení.

Pohledy

motivace:

Snaha vytvořit nové (virtuální) relační proměnné představující abstrakci nad existujícími (základními) proměnnými a jejich typy. Hodnoty (virtuálních) proměnných jsou dány vyhdonocováním dotazů.

virtuální relační proměnná (pohled), angl.: *virtual relvar (view)*

Virtuální relační proměnná (pohled) daného typu je proměnná, která v čase t nabývá hodnoty vzniklé vyhodnocením daného relačního výrazu v čase t.

Tutorial D:

`VAR` ⟨*jméno*⟩ `VIRTUAL` (⟨*dotaz*⟩) ⟨*nepovinný-seznam-klíčů*⟩

SQL:

`CREATE VIEW` ⟨*jméno*⟩ `AS` ⟨*dotaz*⟩

Příklad (Tutorial D: Pohledy)

```
/* base relation variable */
VAR scores BASE
  RELATION {name CHAR, course CHAR, date CHAR, score INT}
  KEY {name, course, date};

/* virtual relation variable */
VAR attendance VIRTUAL (scores {name, course, date})
  KEY {name, course, date};

/* application in a query */
(students JOIN attendance) {ALL BUT date}

/* example with empty key definitions list */
VAR foo VIRTUAL ((students JOIN attendance) {ALL BUT date});
```

Příklad (SQL: Pohledy)

```
/* base relation variable */
CREATE TABLE scores (
  name VARCHAR NOT NULL,
  course VARCHAR NOT NULL,
  date VARCHAR NOT NULL,
  PRIMARY KEY (name, course, date),
  score NUMERIC NOT NULL);

/* virtual relation variable */
CREATE VIEW attendance AS
  SELECT DISTINCT name, course, date FROM scores;

/* application in a query */
SELECT DISTINCT students.*, attendance.course
  FROM attendance NATURAL JOIN students;
```

Příklad (SQL: Tranzitivní uzávěr, Přednáška 5)

```
/* base relation variable */
CREATE TABLE r (
  x NUMERIC NOT NULL,
  y NUMERIC NOT NULL,
  PRIMARY KEY (x, y));

/* virtual relation variable */
CREATE VIEW tr AS
  WITH RECURSIVE
    tr (x, y) AS (
      SELECT * FROM r
        UNION DISTINCT
      SELECT r.x, tr.y FROM r, tr WHERE r.y = tr.x)
    SELECT * FROM tr;
```

Příklad (SQL: Rekurzivní pohledy)

```
/* base relation variable */
CREATE TABLE r (
  x NUMERIC NOT NULL,
  y NUMERIC NOT NULL,
  PRIMARY KEY (x, y));

/* previous view in abbreviated form */
CREATE RECURSIVE VIEW tr (x, y) AS
  SELECT * FROM r
    UNION DISTINCT
  SELECT r.x, tr.y FROM r, tr WHERE r.y = tr.x;

/* application in queries */
SELECT * FROM tr WHERE x > 3;
SELECT * FROM tr WHERE x BETWEEN 1 AND 6;
```

Modifikovatelné pohledy v SQL

modifikovatelný pohled = lze na něm provádět `INSERT`, `UPDATE`, `DELETE`

Pohledy v PostgreSQL jsou modifikovatelné, pokud zároveň platí:

- pohled má jedinou položku za klauzují `FROM`, kterou je jméno tabulky nebo dalšího modifikovatelného pohledu;
- pohled nesmí být výsledkem `UNION`, `INTERSECT` nebo `EXCEPT` nebo obsahovat klauzule `WITH`, `DISTINCT`, `GROUP BY`, `HAVING`, `LIMIT`, nebo `OFFSET`;
- sloupce v pohledu nesmějí být vypočtené hodnoty (viz *rozšíření*).

provedená akce:

- operace `INSERT`, `UPDATE`, `DELETE` se konvertují na operace nad výchozí relací
- pokud *není* pohled modifikovatelný, lze definovat triggery `ON` $\langle akce \rangle$ `DO INSTEAD`

Příklad (SQL: Modifikovatelné pohledy)

```
/* relation view of exam losers */
CREATE VIEW losers AS
  SELECT * FROM scores WHERE score < 20;

/* adding new entry to losers (scores) */
INSERT INTO losers VALUES
  ('Curval', 'KMI/DATA1', '13/02/26', 50);

/* deleting losers */
DELETE FROM losers WHERE ···;

/* setting losers' scores to 0 */
UPDATE losers SET score = 0;
```

poznámka:

- po provedení INSERT nebo UPDATE nemusí být modifikovaná data v pohledu vidět

Příklad (SQL: Materializované pohledy, klauzule MATERIALIZED)

```
/* materialized view of exam losers */
CREATE MATERIALIZED VIEW losers AS
  SELECT * FROM scores WHERE score < 20;

/* error: cannot change materialized view */
INSERT INTO losers VALUES ('Curval', 'KMI/DATA1', '13/01/05', 12);

/* new record added to scores */
INSERT INTO scores VALUES ('Curval', 'KMI/DATA1', '13/01/05', 12);
SELECT * FROM losers WHERE name = 'Curval'; /* no matching tuple */

/* refreshing the view and repeating the query */
REFRESH MATERIALIZED VIEW losers;
SELECT * FROM losers WHERE name = 'Curval';
```

Motivace pro rozšíření

doposud:

- představené relační operace „nevytvářely nové hodnoty“
- význam: hodnoty atributů ve výsledku relační operace jsou vždy některé z hodnot atributů argumentů relační operace

nyní ukážeme:

- relační operace „vytvářející nové hodnoty“
- významné operace: *rozšíření*, *agregace* (další operace odvoditelné)

NAME	STATUS	CHILDREN
Abbe	single	3
Blangis	married	2
Curval	married	3
Durcet	divorced	4

$\Longrightarrow$

NAME	STATUS	CHILDREN	BONUS
Abbe	single	3	3000
Blangis	married	2	2000
Curval	married	3	3000
Durcet	divorced	4	4000

Rozšíření

Definice (rozšíření, angl.: *extension*)

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:

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

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

Tutorial D:

EXTEND ⟨*relační-výraz*⟩ : {⟨*atribut*$_1$⟩ := ⟨*výraz*$_1$⟩, . . . , ⟨*atribut*$_n$⟩ := ⟨*výraz*$_n$⟩}

SQL:

SELECT .*, ⟨*výraz*$_1$⟩ AS ⟨*atribut*$_1$⟩, . . . , ⟨*výraz*$_n$⟩ AS ⟨*atribut*$_n$⟩ FROM ⟨*jméno*⟩

Příklad (Tutorial D: Rošíření n-tic a relací)

```
EXTEND TUPLE {x 10}: {y := x * 20}  ⟹ TUPLE {x 10, y 20}
EXTEND TUPLE {}: {x := 30, y := 40} ⟹ TUPLE {x 30, y 40}
EXTEND TUPLE {x 10}: {x := 20}      /* error, attribute exists */
rl := EXTEND rl: {y := x * 10};     /* error, type collision */

EXTEND RELATION {TUPLE {x 10}, TUPLE {x 20}}: {
  y := x * 2, z := TUPLE {m y + 1}
} ⟹ RELATION {
        TUPLE {x 10, y 20, z TUPLE {m 11}},
        TUPLE {x 20, y 40, z TUPLE {m 21}}}
```

Příklad (SQL: Rozšíření)

```
SELECT *, y AS 20 FROM tbl;
SELECT *, y AS x * 20 FROM tbl;
SELECT *, y1 AS x * 20, y2 AS x + 1 FROM tbl;
```

Příklad (Tutorial D: Opakování, 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: zatím jsme neukázali jako relační přiřazení (PŘEDNÁŠA 3)

Příklad (Tutorial D: UPDATE jako relační přiřazení)

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

/* equivalently as the relational assignment */
person := (person WHERE NOT (salary >= 12000)) UNION
  (EXTEND person WHERE (salary >= 12000): {
    s := (salary * 120) / 100,
    b := bonus + 2000
   }) {name, s, b} RENAME {s AS salary, b AS bonus};
```

obecně: pro $\mathcal{D}$ na schématu $R \cup S$, kde $R \cap S = \emptyset$ a $S = \{y_1, \ldots, y_n\}$ lze vyjádřit

$$\sigma_{\neg\theta}(\mathcal{D}) \cup \rho_{y_1 \leftarrow y'_n, \ldots, y_n \leftarrow y'_n}(\pi_{R \cup \{y'_1, \ldots, y'_n\}}(\epsilon_{y'_1 \leftarrow \theta_1, \ldots, y'_n \leftarrow \theta_n}(\sigma_\theta(\mathcal{D}))))$$

Pořadí rozšíření a dalších operací v SQL

relačním dotazům v SQL ve tvaru:

`SELECT` ⟨*výraz*⟩ `AS` ⟨*atribut*⟩ `FROM` ⟨*jméno*⟩ `WHERE` ⟨*podmínka*⟩

odpovídají dotazy v Tutorial D ve tvaru:

`EXTEND` ⟨*jméno*⟩ `WHERE` ⟨*podmínka*⟩`: {`⟨*atribut*⟩ `:=` ⟨*výraz*⟩`}`

Příklad (Důsledky pro dotazy v SQL)

```
((EXTEND tbl: {b := a * 2}) WHERE b > 500) {b, c}
```

se v SQL nahrazuje:

```
SELECT a * 2 AS b, c FROM tbl WHERE a * 2 > 500;
```

nelze tedy psát:

```
SELECT a * 2 AS b, c FROM tbl WHERE b > 500;
```

Slučování dat do n-tic a rozdělování n-tic

motivace:

V n-ticích nahradíme několik atributů a jejich hodnot jediným atributem, jehož hodnota je n-tice výchozích hodnot; naopak, jeden atribut, jehož hodnotouje n-tice, nahradíme atributy a jejich hodnotami z této n-tice.

Tutorial D (slučování do n-tic):

$\langle n\text{-ticový-výraz}\rangle$ `WRAP` `{`$\langle atribut_1\rangle$`,` $\ldots$ `,`$\langle atribut_n\rangle$`}` `AS` $\langle$*nový-atribut*$\rangle$

$\langle$*relační výraz*$\rangle$ `WRAP` `{`$\langle atribut_1\rangle$`,` $\ldots$ `,`$\langle atribut_n\rangle$`}` `AS` $\langle$*nový-atribut*$\rangle$

Tutorial D (rozdělování n-tic):

$\langle n\text{-ticový-výraz}\rangle$ `UNWRAP` $\langle atribut\rangle$

$\langle$*relační-výraz*$\rangle$ `UNWRAP` $\langle atribut\rangle$

poznámky:

- jako relační operace jsou `WRAP`/`UNWRAP` aplikovány na všechny n-tice relací

Příklad (Tutorial D: Slučování dat do n-tic a rozdělování n-tic)

```
/* wrapping values into tuples */
TUPLE {x 10, y 20, z 30} WRAP {} AS foo
  ⟹ TUPLE {x 10, y 20, z 30, foo TUPLE {}}

TUPLE {x 10, y 20, z 30} WRAP {x} AS foo
  ⟹ TUPLE {y 20, z 30, foo TUPLE {x 10}}

TUPLE {x 10, y 20, z 30} WRAP {x, y} AS foo
  ⟹ TUPLE {z 30, foo TUPLE {x 10, y 20}}

TUPLE {x 10, y 20, z 30} WRAP {x, y, z} AS foo
  ⟹ TUPLE {foo TUPLE {x 10, y 20, z 30}}

/* unwrapping tuples */
TUPLE {z 30, foo TUPLE {x 10, y 20}} UNWRAP foo
  ⟹ TUPLE {x 10, y 20, z 30}
```

Příklad (Tutorial D: Aplikace WRAP/UNWRAP na relace)

```
VAR person BASE
  RELATION {name CHAR, salary INTEGER, bonus INTEGER}
  KEY {name}; ···

person WRAP {salary, bonus} AS wages
  ⟹ RELATION {
    TUPLE {name "Abbe", wages TUPLE {salary 15000, bonus 0}},
    TUPLE {name "Blangis", wages TUPLE {salary 10000, bonus 0}},
      ···}

/* query for the original relvar */
person WHERE salary >= 15000 ⟹ ···

/* analogous query in case of the wrapped data */
(person WRAP {salary, bonus} AS wages)
  WHERE (salary FROM wages) >= 15000 ⟹ ···
```

Příklad (Tutorial D: WRAP/UNWRAP jako odvozené operace)

```
person WRAP {salary, bonus} AS wages ⟹ ···

(EXTEND person: {
  wages := TUPLE {salary salary, bonus bonus}
 }) {ALL BUT salary, bonus} ⟹ ···

VAR wrapped BASE
  RELATION {name CHAR, wages TUPLE {salary INT, bonus INT}}
  KEY {name};

wrapped UNWRAP wages ⟹ ···

(EXTEND wrapped: {
   salary := salary FROM wages,
   bonus := bonus FROM wages
 }) {ALL BUT wages} ⟹ ···
```

Slučování dat do relací a rozdělování

motivace:

Chceme na základě dané relace nad schématem $\{y_1, \ldots, y_m, z_1, \ldots, z_n\}$ *zkonstruovat relaci, která bude mít místo atributů* $z_1, \ldots, z_n$ *jeden nový atribut, jehož hodnoty budou relace nad schématem* $\{z_1, \ldots, z_n\}$ *skládající se ze všech* n*-tic výchozí tabulky mající stejné hodnoty na atributech* $y_1, \ldots, y_m$*.*

y_1	$\cdots$	y_m	z_1	$\cdots$	z_n
c_1	$\cdots$	c_m	d_{11}	$\cdots$	d_{1n}
$\vdots$		$\vdots$	$\vdots$		$\vdots$
c_1	$\cdots$	c_m	d_{k1}	$\cdots$	d_{kn}
e_1	$\cdots$	e_m	f_{11}	$\cdots$	f_{1n}
$\vdots$		$\vdots$	$\vdots$		$\vdots$
e_1	$\cdots$	e_m	f_{l1}	$\cdots$	f_{ln}

$\Longrightarrow$

$\Longleftarrow$

y_1	$\cdots$	y_m	y
c_1	$\cdots$	c_m	z_1 $\cdots$ z_n d_{11} $\cdots$ d_{1n} $\vdots$ $\vdots$ d_{k1} $\cdots$ d_{kn}
e_1	$\cdots$	e_m	z_1 $\cdots$ z_n f_{11} $\cdots$ f_{1n} $\vdots$ $\vdots$ f_{l1} $\cdots$ f_{ln}

Definice slučování a rozdělování

Definice (sloučení, angl.: *grouping*)

Mějme relaci $\mathcal{D}$ na schématu R a uvažujme $S \subseteq R$ a $y \notin R \setminus S$. Položme:

$$\gamma_{y \leftarrow S}(\mathcal{D}) = \big\{ r(R \setminus S) \cup \{ \langle y, \{ r(R \setminus S) \} \circ \mathcal{D} \rangle \} \,|\, r \in \mathcal{D} \big\}.$$

Relace $\gamma_{y \leftarrow S}(\mathcal{D})$ nad schématem $(R \setminus S) \cup \{y\}$ se nazývá relace vzniklá z $\mathcal{D}$ **sloučením atributů** z S do atributu y relačního typu S.

Definice (rozdělování, angl.: *ungrouping*)

Mějme relaci $\mathcal{D}$ na schématu R a $y \in R$ tak, že typem y je množina relací na schématu S, pro který $(R \setminus \{y\}) \cap S = \emptyset$. Položme:

$$\wr_y(\mathcal{D}) = \bigcup \big\{ \{ r(R \setminus \{y\}) \} \bowtie r(y) \,|\, r \in \mathcal{D} \big\}.$$

Relace $\wr_y(\mathcal{D})$ na schématu $(R \setminus y) \cup S$ se nazývá relace vzniklá z relace $\mathcal{D}$ **rozdělením atributu** y relačního typu S na jednotlivé atributy.

Jak číst výrazy v definici sloučení a rozdělování

$$\gamma_{y \leftarrow S}(\mathcal{D}) = \Big\{ \underbrace{\underbrace{r(R \setminus S)}_{\text{projekce } n\text{-tice}} \cup \underbrace{\{\langle y, \overbrace{\{r(R \setminus S)\}}^{\text{relace}} \circ \overbrace{\mathcal{D}}^{\text{komozice relací}} \rangle\}}_{n\text{-tice s jediným atributem } y} y}_{\text{spojení svou } n\text{-tic}} \mid r \in \mathcal{D} \Big\}$$

$$\mathcal{L}_y(\mathcal{D}) = \overbrace{\bigcup \{ \overbrace{\{r(R \setminus \{y\})\}}^{\substack{\text{relace obsahující} \\ n\text{-tici } r \text{ bez } y}} \bowtie r(y) \mid r \in \mathcal{D} \}}^{\substack{\text{spojení dvou relací} \\ \text{sjednocení konečně mnoha relací}}}$$

Operace GROUP a UNGROUP

Tutorial D:

⟨*relační-výraz*⟩ GROUP {⟨$atribut_1$⟩, . . . ,⟨$atribut_n$⟩} AS ⟨*nový-atribut*⟩

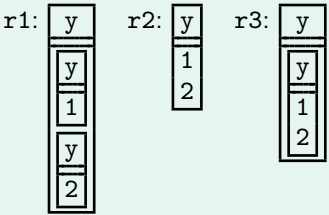
⟨*relační-výraz*⟩ UNGROUP ⟨*atribut*⟩

poznámky:

- narozdíl od WRAP/UNWRAP nemá smysl používat s n-ticovými výrazy

Příklad (Aplikace UNGROUP a potom GROUP nemusí dát výchozí relaci)

```
r1 UNGROUP y = r2        ⟹ TRUE
r2 GROUP {y} AS y = r3   ⟹ TRUE
```



Příklad (Tutorial D: Příklady slučování do relací)

NAME	COURSE	DATE	SCORE
Abbe	KMI/DATA1	13/01/05	56
Abbe	KMI/DATA1	13/01/08	83
Abbe	KMI/PAPR1	12/04/13	65
Blangis	KMI/PAPR1	12/04/14	34
Blangis	KMI/DATA2	13/01/08	13
Curval	KMI/PAPR1	12/04/14	75
Durcet	KMI/PAPR1	12/04/14	75
Durcet	KMI/DATA1	13/02/23	38
Durcet	KMI/DATA1	13/02/26	89

NAME	RESULT		
Abbe	COURSE	DATE	SCORE
	KMI/DATA1	13/01/05	56
	KMI/DATA1	13/01/08	83
	KMI/PAPR1	12/04/13	65
Blangis	COURSE	DATE	SCORE
	KMI/DATA2	13/01/08	13
	KMI/PAPR1	12/04/14	34
Curval	COURSE	DATE	SCORE
	KMI/PAPR1	12/04/14	75
Durcet	COURSE	DATE	SCORE
	KMI/DATA1	13/02/23	38
	KMI/DATA1	13/02/26	89
	KMI/PAPR1	12/04/14	75

```
scores GROUP {course, date, score} AS result
```

Příklad (Tutorial D: Příklady slučování do relací)

NAME	COURSE	DATE	SCORE
Abbe	KMI/DATA1	13/01/05	56
Abbe	KMI/DATA1	13/01/08	83
Abbe	KMI/PAPR1	12/04/13	65
Blangis	KMI/PAPR1	12/04/14	34
Blangis	KMI/DATA2	13/01/08	13
Curval	KMI/PAPR1	12/04/14	75
Durcet	KMI/PAPR1	12/04/14	75
Durcet	KMI/DATA1	13/02/23	38
Durcet	KMI/DATA1	13/02/26	89

DATE	RESULT		
13/01/05	NAME	COURSE	SCORE
	Abbe	KMI/DATA1	56
13/01/08	NAME	COURSE	SCORE
	Abbe	KMI/DATA1	83
	Blangis	KMI/DATA2	13
12/04/13	NAME	COURSE	SCORE
	Abbe	KMI/PAPR1	65
12/04/14	NAME	COURSE	SCORE
	Blangis	KMI/PAPR1	34
	Curval	KMI/PAPR1	75
	Durcet	KMI/PAPR1	75
13/02/23	NAME	COURSE	SCORE
	Durcet	KMI/DATA1	38
13/02/26	NAME	COURSE	SCORE
	Durcet	KMI/DATA1	89

```
scores GROUP {name, course, score} AS result
```

Příklad (Tutorial D: Příklady slučování do relací)

NAME	COURSE	DATE	SCORE
Abbe	KMI/DATA1	13/01/05	56
Abbe	KMI/DATA1	13/01/08	83
Abbe	KMI/PAPR1	12/04/13	65
Blangis	KMI/PAPR1	12/04/14	34
Blangis	KMI/DATA2	13/01/08	13
Curval	KMI/PAPR1	12/04/14	75
Durcet	KMI/PAPR1	12/04/14	75
Durcet	KMI/DATA1	13/02/23	38
Durcet	KMI/DATA1	13/02/26	89

```
scores GROUP {name, score} AS result
```

COURSE	DATE	RESULT	
KMI/DATA1	13/01/05	NAME	SCORE
		Abbe	56
KMI/DATA1	13/01/08	NAME	SCORE
		Abbe	83
KMI/PAPR1	12/04/13	NAME	SCORE
		Abbe	65
KMI/DATA2	13/01/08	NAME	SCORE
		Blangis	13
KMI/PAPR1	12/04/14	NAME	SCORE
		Blangis	34
		Curval	75
		Durcet	75
KMI/DATA1	13/02/23	NAME	SCORE
		Durcet	38
KMI/DATA1	13/02/26	NAME	SCORE
		Durcet	89

Příklad (Tutorial D: Příklady dotazů používajících GROUP)

```
IS_EMPTY (RELATION {foo INT, bar CHAR, baz INT} {}) ⟹ TRUE
IS_EMPTY (TABLE_DUM)                                ⟹ TRUE

IS_EMPTY (TABLE_DEE)                ⟹ FALSE
IS_EMPTY (RELATION {TUPLE {}})      ⟹ FALSE
IS_EMPTY (RELATION {TUPLE {x 10}}) ⟹ FALSE

/* show results of groups including "Abbe" */
(scores GROUP {name, course, score} AS result)
  WHERE NOT (IS_EMPTY (result WHERE name = "Abbe")) ⟹ ···

/* show results of groups composed only of "Abbe" */
(scores GROUP {name, course, score} AS result)
  WHERE (result WHERE name = "Abbe") = result ⟹ ···

/* show ... composed only of students from some_relvar */
(scores GROUP {name, course, score} AS result)
  WHERE (result {name} <= some_relvar {name}) ⟹ ···
```

Příklad (Tutorial D: Mezní případy slučování)

```
/* grouping of the entire schema */
scores GROUP {score, name, course, date} AS x ⟹ ···

/* grouping of the empty subschema */
scores GROUP {} AS x ⟹ ···

/* in particular, for DUM and DEE */
DUM GROUP {} AS x ⟹ RELATION {x RELATION {}} {}
DEE GROUP {} AS x ⟹ RELATION {TUPLE {x RELATION {TUPLE {}}}}
```

poznámky:

- v případě slučování celého schématu je výsledkem relace s jediným atributem a jedinou hodnotou – *celou výchozí relací*
- v případě slučování prázdné množiny atributů je výsledkem tabulka s přidaným atributem, jehož hodnota je v každé n-tici rovna TABLE_DEE

Příklad (Tutorial D: GROUP jako odvozená operace)

```
/* relation group query */
scores GROUP {date, score} AS result ⟹ ···

/* can be expressed by */
EXTEND scores {ALL BUT date, score}: {
  result := (scores RENAME {name AS new_name, course AS new_course}
              WHERE new_name = name AND new_course = course)
            {date, score}
} ⟹ ···

/* shortly, using relational composition */
EXTEND scores {ALL BUT date, score}: {
  result := (scores COMPOSE
             RELATION {TUPLE {name name, course course}})
} ⟹ ···
```

Výhody a nevýhody GROUP/UNGROUP

- možnost mít přehledněji strukturové výsledky dotazů
- možnost mít základní relační proměnné s atributy relačních typů (méně časté)
- začínají se objevovat některé nešvary z hierarchických modelů

Příklad (Tutorial D: Asymetrie dotazů)

```
VAR foo VIRTUAL (scores GROUP {course, date, score} AS result)
  KEY {name};

(scores WHERE name = "Blangis") {date}
(scores WHERE date = "12/04/14") {name}

((foo WHERE name = "Blangis") UNGROUP result) {date}
(foo UNGROUP result WHERE date = "12/04/14") {name}
```

Příklad (Tutorial D: Nahrazení jednostranných vnějších spojení)

id	name
44	Abbe
55	Blangis
66	Curval
77	Durcet

id	course	year	score
44	KMI/DATA1	2012	56
44	KMI/DATA1	2013	83
44	KMI/PAPR1	2013	65
55	KMI/PAPR1	2012	34
55	KMI/PAPR1	2013	95

id	name	result		
44	Abbe	course	year	score
		KMI/DATA1	2012	56
		KMI/DATA1	2013	83
		KMI/PAPR1	2013	65
55	Blangis	course	year	score
		KMI/PAPR1	2012	34
		KMI/PAPR1	2013	95
66	Curval	course	year	score
77	Durcet	course	year	score

```
(students JOIN exams) GROUP {course, year, score} AS result
  UNION
(EXTEND students NOT MATCHING exams: {
   result := RELATION {course CHAR, year INT, score INT} {}})
```

Příklad (Metoda nahrazení oboustranných vnějších spojení)

$\mathcal{D}_1$

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

$\mathcal{D}_2$

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

oboustranné

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

pomocí relací jako hodnot

r1

foo	bar	baz
444	ghi	103
555	ghi	103

foo	bar	baz
555	def	102

foo	bar	baz
666	abc	101

foo	bar	baz

r2

bar	baz	qux
ghi	103	ttt
ghi	103	uuu
ghi	103	vvv

bar	baz	qux
def	102	www
def	102	yyy

bar	baz	qux

bar	baz	qux
abc	111	zzz
ghX	103	xxx

Příklad (Tutorial D: Nahrazení oboustranných vnějších spojení)

```
UNION { /* joinable tuples */
        (((EXTEND r1: { new_bar := bar, new_baz := baz } JOIN r2)
           GROUP {foo, bar, baz} AS r1)
         RENAME {new_bar AS bar, new_baz AS baz})
       GROUP {bar, baz, qux} AS r2,
       /* dangling tuples from r1 */
       ((r1 NOT MATCHING r2) GROUP {foo, bar, baz} AS r1)
         TIMES
       (RELATION {TUPLE {r2 (r2 WHERE FALSE)}}),
       /* dangling tuples from r2 */
       ((r2 NOT MATCHING r1) GROUP {bar, baz, qux} AS r2)
         TIMES
       (RELATION {TUPLE {r1 (r1 WHERE FALSE)}})}
```

Příklad (Tutorial D: Mezifáze v předchozím dotazu)

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

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

r1			new_bar	new_baz	qux
foo	bar	baz	ghi	103	ttt
444	ghi	103			
555	ghi	103			
foo	bar	baz	ghi	103	uuu
444	ghi	103			
555	ghi	103			
foo	bar	baz	ghi	103	vvv
444	ghi	103			
555	ghi	103			
foo	bar	baz	def	102	www
555	def	102			
foo	bar	baz	def	102	yyy
555	def	102			

```
(EXTEND r1: {
   new_bar := bar,
   new_baz := baz
 } JOIN r2) GROUP {foo, bar, baz} AS r1
```

SQL: Čím nahradit vnořené tabulky

motivace:

Čím v SQL obejít možnost používat atributy relačních typu?

transformace $\mathcal{D}$ nad schématem R do několika tabulek s atributy skalárních typů:

1. pokud jsou všechny atributy schématu R skalární, pak jsme hotovi (vrátíme $\mathcal{D}$);
2. pokud je $y \in R$ atribut relačního typu S, pak pro $\{r(y) \mid r \in \mathcal{D}\} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ uvažujeme následující relace:

$$\mathcal{D}' = \{r(R \setminus \{y\}) \cup \{\langle y', i\rangle\} \mid r \in \mathcal{D} \text{ a } r(y) = \mathcal{D}_i\},$$
$$\mathcal{D}'' = \bigcup_{i=1}^{n} \big(\mathcal{D}_i \bowtie \{\{\langle y', i\rangle\}\}\big),$$

kde $y' \notin R$ a $y' \notin S$. Bod 1 zopakujeme pro $\mathcal{D}'$ a $\mathcal{D}''$.

poznámky:

- předchozí transformace je daná *rekurzivním předpisem* (y' je vždy nový atribut)
- stejný výsledek jako `UNGROUP` výchozích dat získáme násobnou kompozicí

Příklad (Nahrazování atributů relačních typů)

NAME	RESULT		
Abbe	COURSE	DATE	SCORE
	KMI/DATA1	13/01/05	56
	KMI/DATA1	13/01/08	83
	KMI/PAPR1	12/04/13	65
Blangis	COURSE	DATE	SCORE
	KMI/DATA2	13/01/08	13
	KMI/PAPR1	12/04/14	34
Curval	COURSE	DATE	SCORE
	KMI/PAPR1	12/04/14	75
Durcet	COURSE	DATE	SCORE
	KMI/DATA1	13/02/23	38
	KMI/DATA1	13/02/26	89
	KMI/PAPR1	12/04/14	75

NAME	y
Abbe	1
Blangis	2
Curval	3
Durcet	4

y	COURSE	DATE	SCORE
1	KMI/DATA1	13/01/05	56
1	KMI/DATA1	13/01/08	83
1	KMI/PAPR1	12/04/13	65
2	KMI/DATA2	13/01/08	13
2	KMI/PAPR1	12/04/14	34
3	KMI/PAPR1	12/04/14	75
4	KMI/DATA1	13/02/23	38
4	KMI/DATA1	13/02/26	89
4	KMI/PAPR1	12/04/14	75

Přednáška 6: Závěr

pojmy k zapamatování:

- virtuální relační proměnná, pohled
- rozšíření, slučování a rozdělování n-tic
- slučování do relací, rozdělování relací
- nahrazení vnějších spojení pomocí slučování

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