Databázové systémy

Další relační operace

Vilém Vychodil

KMI/DATA1, Přednáška 6

Databázové systémy

Přednáška 6: Přehled

1. Další relační operace:
   - rozšíření,
   - agregace, sumarizace.
2. Relační operace specifické pro Tutorial D:
   - operace typu `WRAP` / `UNWRAP`,
   - operace typu `GROUP` / `UNGROUP`,
   - role operací jako substituentů za vnější spojení.
3. Vnořené dotazy a kvantifikace:
   - použití v SQL,
   - analogie vnořených dotazů v Tutorial D.
4. Relační dělení:
   - obecná definice, vlastnosti,
   - implementace v SQL.

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*⟩`) KEY ...`

SQL:

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

Motivace

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

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

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` `(`{$\langle atribut_1\rangle$, ..., $\langle atribut_n\rangle$}` AS` $\langle$*nový-atribut*$\rangle$`)`

$\langle$*relační výraz*$\rangle$ `WRAP` `(`{$\langle atribut_1\rangle$, ..., $\langle atribut_n\rangle$}` 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 ⟹ ···
```

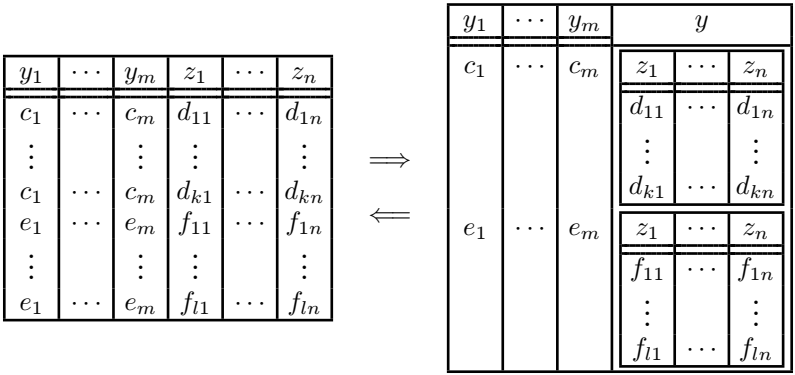
Příklad (Tutorial D: WRAP/UNWRAP jako odvozené operace)

```
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* $y_1, \ldots, y_m$*.*



Definice slučování a rozdělování

Definice (sloučení, angl.: *grouping*)

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

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

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.

Operace GROUP a UNGROUP

Tutorial D:

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

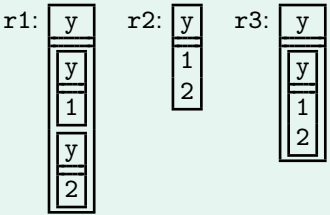
⟨*relační-výraz*⟩ UNGROUP (⟨*atribut*⟩)

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)}})}
```

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

Agregační operace

motivace:

Ze všech hodnot konkrétního atributu nějaké relace chceme vypočíst jedinou (agregovanou) hodnotu, typicky skalárního typu.

agregační funkce, angl.: *aggregation function*

Za agragační funkci považujeme každou funkci, která na základě dané relace a jejího atributu vrací hodnotu, která závisí pouze na hodnotách daného atributu v relaci.

Tutorial D:

- primitivní agregační operátory: `AND`, `AVG` (průměrná hodnota), `COUNT` (vyžaduje právě jeden argument – relaci), `D_UNION`, `EQUIV` (logická ekvivalence), `EXACTLY` (vyžaduje dodatečný argument n typu `INT`, uvedený jako první; výsledek agregace je `TRUE` pokud má specifikovaný atribut právě n hodnot `TRUE`), `INTERSECT`, `MAX`, `MIN`, `OR`, `SUM`, `UNION`, `XOR`, `XUNION`

Příklad (Tutorial D: Agregační operace)

```
/* aggregations of numerical values */
AVG (scores, score)          ⟹ 58.666
MIN (scores, score)          ⟹ 13
MAX (scores, score)          ⟹ 89
SUM (scores, score)          ⟹ 528
SUM (scores {score}, score)  ⟹ 453
COUNT (scores)               ⟹ 9
COUNT (scores {score})       ⟹ 8

/*  aggregations of boolean values */
EXACTLY (6, EXTEND scores: {foo := score > 50}, foo) ⟹ TRUE

/*  aggregations of relational values */
UNION (scores GROUP ({date, score} AS result), result)     ⟹ ···
INTERSECT (scores GROUP ({date, score} AS result), result) ⟹ ···
```

Příklad (Tutorial D: Aplikace agregačních operací, vyjádření UNGROUP)

```
/* new virtual variable */
VAR goo VIRTUAL (scores GROUP ({date, score} AS result))
  KEY {name, course};

/* ungrouping of the result attribute */
goo UNGROUP (result)

/* can equivalently be expressed by */
((EXTEND UNION (goo, result): {
    res := RELATION {TUPLE {score score, date date}}
  }) TIMES goo WHERE res <= result) {ALL BUT res, result}

/* alternatively, using WRAP, UNWRAP, and IN */
((UNION (goo, result) WRAP ({score, date} AS res)) TIMES goo
 WHERE res IN result) {ALL BUT result} UNWRAP (res)
```

Sumarizace

agregace × sumarizace:

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

Tutorial D:

- obecné výrazy `SUMMARIZE` (viz dále)

SQL:

```
SELECT DISTINCT ⟨výraz_1⟩ AS ⟨nový-atribut_1⟩, ...
  FROM ⟨jméno⟩
  WHERE ⟨podmínka⟩
  GROUP BY ⟨atribut_1⟩, ..., ⟨atribut_n⟩
  HAVING ⟨agregační-podmínka⟩
```

Sumarizace v Tutorial D: Syntax

možné tvary použití:

SUMMARIZE ⟨*relační-výraz*⟩ : {⟨*nový-atribut*$_1$⟩ := ⟨*výraz*$_1$⟩, . . .}

SUMMARIZE ⟨*relační-výraz*⟩ BY {⟨*atribut*$_1$⟩, . . .} : {⟨*nový-atr*$_1$⟩ := ⟨*výraz*$_1$⟩, . . .}

SUMMARIZE ⟨*relační-výraz*⟩ PER (⟨*univerzum*⟩) : {⟨*nový-atr*$_1$⟩ := ⟨*výraz*$_1$⟩, . . .}

role argumentů:

- hodnotou ⟨*relační-výraz*⟩ je relace, kterou chceme sumarizovat
- ⟨*nový-atribut*$_i$⟩ jsou atributy nevyskytující se ve schématu relačního výrazu
- ⟨*výraz*$_i$⟩ jsou výrazy obsahující *sumarizační funkce*: AND, AVG, COUNT, D_UNION, EQUIV, EXACTLY, INTERSECT, MAX, MIN, OR, SUM, UNION, XOR, XUNION
- schéma relačního výrazu ⟨*univerzum*⟩ je podmnožinou schémtu ⟨*relační-výraz*⟩
- v druhém případě je možné použít i notaci BY {ALL BUT ⟨*atribut*$_1$⟩, . . .}

Sumarizace v Tutorial D: Sémantika

ekvivalentní vyjádření v SUMMARIZE:

BY {⟨*atribut*$_1$⟩,...} ≡ PER (⟨*relační-výraz*⟩ {⟨*atribut*$_1$⟩,...})

vynechání PER i BY je ekvivalentní uvedení PER (TABLE_DEE)

interpretace sumarizace ve tvaru:

SUMMARIZE ⟨*relační-výraz*⟩ PER (⟨*univerzum*⟩) : {⟨*nový-atr*$_1$⟩ := ⟨*výraz*$_1$⟩,...}

je vyjádřena v následujících krocích:

1. nechť ⟨*relační-výraz*⟩ má hodnotu $\mathcal{D}$ (na schématu R)
   a ⟨*univerzum*⟩ má hodnotu $\mathcal{D}_U$ (na schématu $S \subseteq R$);
2. pro každou $s \in \mathcal{D}_U$ vypočti $\{r \in \mathcal{D} \mid s \subseteq r\}$;
3. vyhodnoť všechny ⟨*výraz*$_i$⟩ za předpokladu, že atributy $y \in R \setminus S$ odkazují na množiny hodnot atributů z $\mathcal{D}_y$ (a musejí být agregovány); n-tici výsledků označ t;
4. vlož $s \cup t$ do výsledku

Příklad (Tutorial D: Sumarizace)

```
/* aggregation vs. summarization */
AVG (scores, score) ⟹ 58.666

SUMMARIZE scores: {average := AVG (score)}
  ⟹ RELATION {TUPLE {average 58.666}}

SUMMARIZE scores BY {name}: {m := MAX (score), cnt := COUNT ()}

/* allows to have zero counts */
SUMMARIZE scores PER (people {name}): {cnt := COUNT ()}

/* additional condition */
SUMMARIZE scores PER (people {name}): {test_cnt := COUNT ()}
  WHERE test_cnt <= 1;

/* sumarizace sjednocenim (!!) */
SUMMARIZE (scores GROUP ({bar, baz} AS new)) {new}: {a := UNION (
   new)}
```

Příklad (Tutorial D: Sumarizace jako odvozená operace)

```
/* summarization */
SUMMARIZE foo PER (baz {qux}): {
  total := COUNT (),
  conj := AND (b)
}

/* can be expressed as extension */
EXTEND baz {qux}: {
  total := COUNT (foo COMPOSE RELATION {TUPLE {qux qux}}),
  conj := AND (foo COMPOSE RELATION {TUPLE {qux qux}}, b)
}
```

foo:

QUX	B
10	FALSE
10	TRUE
20	FALSE

baz:

QUX	NAME
10	Abbe
20	Blangis
30	Curval

result:

QUX	TOTAL	CONJ
10	2	FALSE
20	1	FALSE
30	0	TRUE

Příklad (SQL: Sumarizace)

```
SELECT count (*) FROM scores;
SELECT count (name) FROM scores;

SELECT avg (score), min (score), max (score) FROM scores;
SELECT avg (score) AS average FROM scores;
SELECT name, avg (score) AS average FROM scores GROUP BY name;

SELECT name, avg (score) AS average FROM scores
  WHERE date LIKE '13%'
  GROUP BY name;

SELECT name, avg (score) AS average FROM scores
  WHERE date LIKE '13%'
  GROUP BY name
  HAVING min (score) < 50;
```

Vnořené dotazy a kvantifikace v SQL

vnořený dotaz ($\langle dotaz \rangle$ musí vracet relaci s jedním atributem):

```
SELECT * FROM ⟨jméno⟩ WHERE ⟨atribut⟩ IN (⟨dotaz⟩)
```

vnořený dotaz ($\langle dotaz \rangle$ musí vracet singleton):

```
SELECT * FROM ⟨jméno⟩ WHERE ⟨atribut⟩ = (⟨dotaz⟩)
SELECT * FROM ⟨jméno⟩ WHERE ⟨atribut⟩ <= (⟨dotaz⟩)
```

existenční a univerzální kvantifikace podmínky:

```
SELECT * FROM ⟨jméno⟩ WHERE ⟨atribut⟩ <= SOME (⟨dotaz⟩)
SELECT * FROM ⟨jméno⟩ WHERE ⟨atribut⟩ <= ALL (⟨dotaz⟩)
```

vnořený dotaz na úrovni atributu:

```
SELECT *, (⟨dotaz⟩) AS ⟨nový-atribut⟩ FROM ⟨jméno⟩
```

vnořený dotaz na úrovni jména tabulky:

```
SELECT * FROM (⟨dotaz⟩) AS ⟨jméno-vnořeného-dotazu⟩
```

Příklad (SQL: Vnořené dotazy)

```
/* condition with general quantifier */
SELECT * FROM foo WHERE x > ALL (SELECT x FROM bar);

/* condition of presence in query result */
SELECT * FROM foo WHERE x IN (SELECT x FROM bar);

/* nested query new_bar */
SELECT * FROM (SELECT x FROM bar) AS new_bar WHERE x < 20;
SELECT x FROM bar WHERE x < 20;

/* nontrivial application of the nested query */
SELECT * FROM
  (SELECT x FROM bar WHERE x > 30) AS new_bar
  NATURAL JOIN baz;

/* the following nested query can introduce NULLs! */
SELECT x, (SELECT x FROM bar WHERE x = foo.x) AS y FROM foo;
```

Příklad (Tutorial D: Analogie „vnořených dotazů“ ze SQL)

```
/* analogy of generally quantified SQL statement */
foo WHERE IS_EMPTY (bar RENAME {x AS nx} WHERE NOT (x > nx))

/* analogy of existentially quantified SQL statement */
foo WHERE NOT IS_EMPTY (bar RENAME {x AS nx} WHERE (x > nx))

/* analogy of IN-clause in SQL statement */
foo WHERE NOT IS_EMPTY (bar RENAME {x AS nx} WHERE (x = nx))

/* analogy of the nested restrict-before-join query in SQL */
(bar WHERE x > 30) JOIN baz

/* analogy of nested query in place of an attribute */
EXTEND foo: {
  /* add new attribute of relational type */
  y := bar RENAME {x AS nx} WHERE (x = nx)
}
```

Intermezzo: Nerelační modifikace výpisu dotazu

nerelační operace třídění:

- výsledek reprezentuje pořád stejnou relaci
- stojí mimo RM (přidáním do modelu bychom neudrželi 1NF)
- „pořadí“ n-tic je pouze věcí výpisu, tj. externí prezentace relace

Tutorial D:

$\langle$*relační-výraz*$\rangle$ `ORDER (DESC` $\langle atribut_1 \rangle$`, ASC` $\langle atribut_2 \rangle$`, ...)`

SQL:

`SELECT * FROM` $\langle$*jméno*$\rangle$ `ORDER BY` $\langle výraz_1 \rangle$ `DESC,` $\langle výraz_1 \rangle$ `ASC, ...`

poznámky:

- implicitní je `ASC` (vzestupně), volitelné `DESC` (sestupně)
- další klauzule v SQL: `LIMIT`, `OFFSET` (viz cvičení)

Motivace pro relační dělení

dotazy tvaru „některá φ jsou ψ“:

„Studenti, kteří mají zapsaný nějaký předmět.“
„Díly, které patří k některým součástkám.“
„Dodavatelé, kteří dodávají některé výrobky.“
⋮

dotazy tvaru „všechna φ jsou ψ“:

„Studenti, kteří mají zapsaný všechny (vyžadované) předměty.“
„Součástky, jejichž všechny díly jsou na skladě.“
„Dodavatelé, kteří dodávají všechny výrobky.“
⋮

vyjadřování dotazů **existenčního** a **všeobecného** chrakteru:

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

Dělení

Definice (dělení, angl.: *division*)

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. Položme:

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

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

Tutorial D:

$\langle$*relační-výraz*$_1\rangle$ `DIVIDEBY` $\langle$*relační-výraz*$_2\rangle$ `PER` ($\langle$*relační-výraz*$_3\rangle$)

poznámka:

- formulace se zjednodušuje, pokud R, S a T splňují dodatečné podmínky

Speciální případy dělení

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

se zjednoduší pokud platí $R \cap S = \emptyset$:

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

pokud navíc $T = R \cup S$, pak se předchozí zjednoduší:

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

interpretace pro $\mathcal{D}_1$ (výrobci), $\mathcal{D}_2$ (výrobky), $\mathcal{D}_3$ (kdo vyrábí co):

- $\mathcal{D}_1 \div_{\mathcal{D}_3} \mathcal{D}_2$ *výrobci vyrábějící všechny výrobky*
- $\mathcal{D}_2 \div_{\mathcal{D}_3} \mathcal{D}_1$ *výrobky vyráběné všemi výrobci*

Příklad (Vstupní data 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

Příklad (Tutorial D: Relační dělení)

```
students DIVIDEBY courses PER (enrolled)
  ⟹ RELATION {TUPLE {name "Blangis"}, TUPLE {name "Curval"}}

courses DIVIDEBY students PER (enrolled) ⟹ 0
  ⟹ RELATION {name CHAR} {}

students DIVIDEBY DUM PER (enrolled) = students ⟹ TRUE
courses DIVIDEBY DUM PER (enrolled) = courses   ⟹ TRUE
```

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

Věta (Vyjádření $\div$ pomocí π, $\setminus$ a $\bowtie$)

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

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

Důkaz.

Platí, že $r(R\cap T)$ náleží do výrazu na pravé straně rovnosti p. k. existuje r tak, že $r(R\cap T) \in \pi_{R\cap T}(\mathcal{D}_1)$ a $r(R\cap T) \notin \pi_{R\cap T}((\pi_{R\cap T}(\mathcal{D}_1) \bowtie \pi_{S\cap T}(\mathcal{D}_2)) \setminus \pi_{(R\cup S)\cap T}(\mathcal{D}_3))$. Dále platí, že $r'(R\cap T) \in \pi_{R\cap T}((\pi_{R\cap T}(\mathcal{D}_1) \bowtie \pi_{S\cap T}(\mathcal{D}_2)) \setminus \pi_{(R\cup S)\cap T}(\mathcal{D}_3))$ p. k. $r' \in \mathcal{D}_1$ a existuje $s' \in \mathcal{D}_2$ tak, že $r'(R\cap T)$ je spojitelná s $s'(S\cap T)$, to jest $r'(R\cap S\cap T) = s'(R\cap S\cap T)$ a navíc $r'(R\cap T)s'(S\cap T) \notin \pi_{(R\cup S)\cap T}(\mathcal{D}_3)$. To jest pro předpokládané r neexistuje žádné $s \in \mathcal{D}_2$ splňující $r(R\cap S\cap T) = s(R\cap S\cap T)$ a $r(R\cap T)s(S\cap T) \notin \pi_{(R\cup S)\cap T}(\mathcal{D}_3)$. To jest, pro $r \in \mathcal{D}_1$ a každé $s \in \mathcal{D}_2$ platí, že pokud $r(R\cap S\cap T) = s(R\cap S\cap T)$, pak $r(R\cap T)s(S\cap T) \in \pi_{(R\cup S)\cap T}(\mathcal{D}_3)$. To znamená, že $r(R\cap T) \in \mathcal{D}_1 \div_{\mathcal{D}_3} \mathcal{D}_2$. □

odtud: pokud $R\cap S = \emptyset$ a $R\cup S = T$, pak $\mathcal{D}_1 \div_{\mathcal{D}_3} \mathcal{D}_2 = \mathcal{D}_1 \setminus \pi_R\big((\mathcal{D}_1 \bowtie \mathcal{D}_2) \setminus \mathcal{D}_3\big)$

Příklad (Tutorial D: Relační dělení jako odvozená operace)

```
/* base relations */
VAR r1 BASE RELATION {r INT, x INT, a1 INT} KEY {r, x, a1};
VAR r2 BASE RELATION {s INT, x INT, a2 INT} KEY {s, x, a2};
VAR r3 BASE RELATION {t INT, a1 INT, a2 INT} KEY {t, a1, a2};

/* relational division */
r1 DIVIDEBY r2 PER (r3)

/* equivalently using projections, joins, and difference */
r1 {a1}
    MINUS
((r1 {a1} JOIN r2 {a2})
     MINUS
 r3 {a1, a2}) {a1}
```

Příklad (SQL: Vyjádření dělení v SQL)

```
/* analogously as in the previous example */
CREATE TABLE r1 (r NUMERIC, x NUMERIC, a1 NUMERIC);
CREATE TABLE r2 (s NUMERIC, x NUMERIC, a2 NUMERIC);
CREATE TABLE r3 (t NUMERIC, a1 NUMERIC, a2 NUMERIC);

/* division as nested query */
SELECT DISTINCT a1 FROM r1
  EXCEPT
SELECT DISTINCT a1 FROM
  (SELECT DISTINCT * FROM
      (SELECT DISTINCT a1 FROM r1) AS new1
         NATURAL JOIN
      (SELECT DISTINCT a2 FROM r2) AS new2
    EXCEPT
   SELECT DISTINCT a1, a2 FROM r3) AS new3;
```

Přednáška 6: Závěr

pojmy k zapamatování:

- virtuální relační proměnná, pohled
- rozšíření, slučování a rozdělování n-tic, slučování do relací
- agregace, sumarizace, vnořený dotaz
- nerelační modifikace dotazu, relační dělení

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