

Formální sémantika SQL

Tomáš Matoušek

tmd.havit.cz

Obsah

- formalizace SQL: tříhodnotový kalkul E3VPC
- gramatika SQL
- převod SQL na E3VPC
- ekvivalence dotazů SQL

Extended 3-valued tuple predicate calculus (E3VPC)

- rozšíření 2-hodnotového n -ticového kalkulu (2VPC)
- jednoduchá struktura
- snadnější manipulace s dotazy (např. optimalizace)
- schopnost vyjádřit vše, co lze vyjádřit v SQL

Příklad

SQL:

```
SELECT d.manager
FROM dept d
WHERE d.location = ALL
      SELECT e.residence
      FROM emp e
      WHERE e.deptId = d.id
GROUP BY d.manager
HAVING AVG(d.nofEmp) > 500
```

schéma:

```
dept(id, nofEmp, location, manager)
emp(id, deptId, residence)
```

E3VPC:

$\{d \in \text{dept} : \|\ (\forall \{e \in \text{emp} : \| e.\text{deptId} = d.\text{id} \| \}^F) \ d.\text{location} = e.\text{residence} \} \wedge$

$\text{AVG}(d.\text{nofEmp}) \{d \in \text{dept} : \|\ (\forall \{e \in \text{emp} : \| e.\text{deptId} = d.\text{id} \| \}^F) \ d.\text{location} = e.\text{residence} \} \wedge d.\text{manager} \cong d.\uparrow.\text{manager} \| \}^F \} > 500$

$\| \}^F \}$

Jazyk E3VPC

operátory

=, ≠, <, ≤, ... mají obvyklý význam

pokud jeden z operandů je **null**, výsledek je **unknown**

≅ porovnává s hodnotou **null**:

chová se stejně jako =, kromě případu, kdy mají oba operandy hodnotu **null**, pak je výsledek **true**

≅	5	1	N
5	T	F	U
1	F	T	U
N	U	U	T

·↑ je unární operátor vnější vazby

|| · ||^α je unární operátor interpretace, α ∈ {T, F}

výrazy

- E3VPC výraz je

$$\{t(v_1, \dots, v_n): \|P(v_1, \dots, v_n)\|^\alpha\}$$

v_1, \dots, v_n

n -ticové proměnné

$t(v_1, \dots, v_n)$

cílový seznam výrazu ve tvaru

$$v_1 \in R_1, \dots, v_n \in R_n,$$

R_1, \dots, R_n jsou relace

$P(v_1, \dots, v_n)$

formule

$\| \cdot \|^\alpha$

interpretační operátor

$$\alpha \in \{T, F\}$$

termy

- konstanta je term
- v proměnná, a atribut $\Rightarrow v.a, v\uparrow.a$ jsou termy
- S výraz, v proměnná, a atribut, F agregační funkce $\Rightarrow F(v.a)S$ je term

$F \in \{\text{COUNT, AVG, SUM, MIN, MAX, COUNTD, AVGD, SUMD}\}$

xxx D odpovídá v SQL agregační funkci $xxx(\text{DISTINCT})$

formule

- t_1, t_2 termy, θ op. porovnání $\Rightarrow t_1 \theta t_2$ je atomická formule
- T, F, U (**true, false, unknown**) jsou atomické formule
- P atomická formule $\Rightarrow \|P\|^\alpha$ je atomická formule
- atomická formule je formule
- P, Q formule $\Rightarrow \neg P, P \wedge Q, P \vee Q$ jsou formule
- S výraz, Q formule $\Rightarrow \exists S Q, \forall S Q$ jsou formule

Interpretace formulí

- Výsledkem vyhodnocení formule P je jedna ze 3 hodnot
true, false, unknown
- $\llbracket \cdot \rrbracket^T$ interpretuje **unknown** jako **true** (pozitivní interpretace)
- $\llbracket \cdot \rrbracket^F$ interpretuje **unknown** jako **false** (negativní interpretace)
- Výsledkem vyhodnocení formule $\llbracket P \rrbracket^\alpha$ je jedna z hodnot
true, false

Interpretace formálně

- $P(x)$ E3VPC formule, $Q(x)$ 2VPC formule.
- $Q(x)$ je **pozitivně interpretovaný** 2-hodnotový ekvivalent $P(x)$, značíme $\|P\|^T$, jestliže pro každé x platí:
 - a) $P(x) \equiv T \Rightarrow Q(x) \equiv T$
 - b) $P(x) \equiv F \Rightarrow Q(x) \equiv F$
 - c) $P(x) \equiv U \Rightarrow Q(x) \equiv \mathbf{T}$
- $Q(x)$ je **negativně interpretovaný** 2-hodnotový ekvivalent $P(x)$, značíme $\|P\|^F$, jestliže pro každé x platí:
 - a) $P(x) \equiv T \Rightarrow Q(x) \equiv T$
 - b) $P(x) \equiv F \Rightarrow Q(x) \equiv F$
 - c) $P(x) \equiv U \Rightarrow Q(x) \equiv \mathbf{F}$

Vnější vazba

- Operátor vnější vazby \uparrow mění rozsah platnosti proměnné
- $v\uparrow$ se odkazuje na proměnnou na nejbližší vyšší úrovni
- Např. ve výrazu

$$\{v \in S: \forall\{v \in T: \|v.a = v\uparrow.b\|^T\} \dots \}$$

je proměnná v dvakrát – jednou jde o proměnnou $v \in T$, podruhé $v \in S$, jelikož je použit operátor \uparrow .

Význam E3VPC výrazu

$$\{v_1 \in R_1, \dots, v_k \in R_k: Q(v_1, \dots, v_k)\}$$

- výsledkem vyhodnocení výrazu je množina n -tic vybraných z relací R_1, \dots, R_k , které po dosazení za příslušné proměnné v_1, \dots, v_k splňují formuli Q
- Q musí být formule 2VPC
- předpokládá se, že n -tice v relacích jsou navzájem různé
- vrací celé n -tice, nelze vrátit jejich části

Kvantifikace

$$U = \forall \{x \in S: P(x)\} Q(x) \equiv (\forall x) x \in S \wedge P(x) \rightarrow Q(x)$$

$$E = \exists \{x \in S: P(x)\} Q(x) \equiv (\exists x) x \in S \wedge P(x) \wedge Q(x)$$

označíme-li $M = \{x \in S: P(x)\}$, pak platí:

$$U \equiv \mathbf{true} \quad \Leftrightarrow \quad (\forall x \in M) Q(x) \equiv \mathbf{true}$$

$$U \equiv \mathbf{false} \quad \Leftrightarrow \quad (\exists x \in M) Q(x) \equiv \mathbf{false}$$

$$U \equiv \mathbf{unknown} \quad \Leftrightarrow \quad \text{jinak}$$

$$E \equiv \mathbf{true} \quad \Leftrightarrow \quad (\exists x \in M) Q(x) \equiv \mathbf{true}$$

$$E \equiv \mathbf{false} \quad \Leftrightarrow \quad (\forall x \in M) Q(x) \equiv \mathbf{false}$$

$$E \equiv \mathbf{unknown} \quad \Leftrightarrow \quad \text{jinak}$$

Gramatika SQL

Cíl

- Syntaxí řízený překlad SQL dotazu na výraz E3VPC.
- Potřebujeme tedy popsat syntax SQL pomocí gramatiky
- Pro přehlednost zjednodušíme popis SQL výrazů gramatikou

Zjednodušení

1. Výrazy „boolean expression of“ nebo „list of“ jsou zkratkou pro množinu zřejmých pravidel.
2. Aritmetické výrazy s atributy nejsou uvažovány.
3. Používání aliasů relací je povinné.
4. Do některých pravidel jsou zavedeny další neterminály, případně jsou některá pravidla rozdělena do více menších.

Gramatika SQL (pod)dotazu

vrací *n*-tici:

<QUERY> ::=

SELECT [**ALL** | **DISTINCT**] <SELECT LIST> <FROM CLAUSE>
[<WHERE CLAUSE>][<GROUP BY CLAUSE>][<HAVING CLAUSE>]

vrací hodnotu nebo sloupec:

<SUBQ> ::=

SELECT [**ALL** | **DISTINCT**] <COL OR VAL> <FROM CLAUSE>
[<WHERE CLAUSE>][<GROUP BY CLAUSE>][<HAVING CLAUSE>]

vrací výsledek agregační funkce:

<AF SUBQ> ::=

SELECT [**ALL** | **DISTINCT**] <FUNCTION> <FROM CLAUSE>
[<WHERE CLAUSE>][<GROUP BY CLAUSE>][<HAVING CLAUSE>]

pravidlo <SELECT LIST>

<SELECT LIST> ::= "list of <SELECT ELEMENT>"

<SELECT ELEMENT> ::= <COL OR VAL> | <FUNCTION>

<COL OR VAL> ::= <alias>.<column> | <literal>

<FUNCTION> ::= <COUNT> | <AGGR>

<COUNT> ::= <COUNT D> | **COUNT**(*)

<COUNT D> ::= **COUNT** (**DISTINCT** <alias>.<column>)

<AGGR> ::= <AGGR D> | <AGGR A>

<AGGR D> ::= <AGGR NAME>(**DISTINCT** <alias>.<column>)

<AGGR A> ::= <AGGR NAME>([**ALL**] <alias>.<column>)

<AGGR NAME> ::= **AVG** | **MAX** | **MIN** | **SUM**

- seznam hodnot, sloupců nebo výsledků agregačních funkcí
- nelze vnořit SELECT

pravidla <* CLAUSE>

<FROM CLAUSE> ::= **FROM** "list of <TABLE REFERENCE>"

<TABLE REFERENCE> ::= <table> <alias>

<WHERE CLAUSE> ::= **WHERE** <WHERE CONDITION>

<HAVING CLAUSE> ::= **HAVING** <HAVING CONDITION>

<GROUP BY CLAUSE> ::= **GROUP BY** "list of <alias>.<column>"

- do klauzule FROM nelze vnořit SELECT
- SELECT lze vnořit jen do klauzulí WHERE, HAVING za
[NOT] IN, EXISTS, operátor [SOME | ALL]

pravidlo <WHERE CONDITION>

- podmínka je seznamem jednoduchých a/nebo složených predikátů:

<WHERE CONDITION> ::= "boolean expression of <WHERE PRED>"

<WHERE PRED> ::= <SIMPLE PRED> | <COMPLEX PRED>

<SIMPLE PRED> ::= <COL OR VAL> <comp op> <COL OR VAL>

<COMPLEX PRED> ::=

<SOME QUANTIFIED PRED> | <SOME QUANTIFIED AF PRED> |

<ALL QUANTIFIED PRED> | <ALL QUANTIFIED AF PRED> |

<COMPLEX IN PRED> | <COMPLEX IN AF PRED> |

<COMPLEX NOT IN PRED> | <COMPLEX NOT IN AF PRED> |

<COMPLEX COMP PRED> | <COMPLEX COMP AF PRED> |

<EXISTS PRED>

pravidla <* [AF] PRED>

<SOME Q. PRED> ::= <COL OR VAL> <comp op> **SOME** <SUBQ>

<ALL Q. PRED> ::= <COL OR VAL> <comp op> **ALL** <SUBQ>

<COMPLEX IN PRED> ::= <COL OR VAL> **IN** <SUBQ>

<COMPLEX NOT IN PRED> ::= <COL OR VAL> **NOT IN** <SUBQ>

<COMPLEX COMP PRED> ::= <COL OR VAL> <comp op> <SUBQ>

analogicky s agragační funkcí v poddotazu:

<SOME Q. **AF** PRED> ::= <COL OR VAL> <comp op> **SOME** <**AF** SUBQ>

<ALL Q. **AF** PRED> ::= <COL OR VAL> <comp op> **ALL** <**AF** SUBQ>

<COMPLEX IN **AF** PRED> ::= <COL OR VAL> **IN** <**AF** SUBQ>

<COMPLEX NOT IN **AF** PRED> ::= <COL OR VAL> **NOT IN** <**AF** SUBQ>

<COMPLEX COMP **AF** PRED> ::= <COL OR VAL> <comp op> <**AF** SUBQ>

<EXISTS PRED> ::= **EXISTS** <SUBQ>

pravidlo <HAVING CONDITION>

- podmínka je seznamem jednoduchých resp. složených predikátů, porovnání výsledků agregačních funkce s hodnotou nebo jinou agregační funkcí:

<HAVING CONDITION> ::= "boolean expression of <HAVING PRED>"

<HAVING PRED> ::= <H SIMPLE PRED> | <H COMPLEX PRED> |
<H AF COLUMN PRED> | <H AF FUNCTION PRED> |
<H AF COMPLEX PRED>

<H SIMPLE PRED> ::= <SIMPLE PRED>

<H COMPLEX PRED> ::= <COMPLEX PRED>

<H AF COLUMN PRED> ::= <FUNCTION> <comp op> <COL OR VAL>

<H AF FUNCTION PRED> ::= <FUNCTION> <comp op> <FUNCTION>

pravidlo $\langle H \text{ AF COMPLEX PRED} \rangle$

$\langle H \text{ AF COMPLEX PRED} \rangle ::=$

$\langle \text{AF SOME Q. PRED} \rangle \mid \langle \text{AF SOME Q. AF PRED} \rangle \mid$

$\langle \text{AF ALL Q. PRED} \rangle \mid \langle \text{AF ALL Q. AF PRED} \rangle \mid$

$\langle \text{AF COMPLEX IN PRED} \rangle \mid \langle \text{AF COMPLEX IN AF PRED} \rangle \mid$

$\langle \text{AF COMPLEX NOT IN PRED} \rangle \mid \langle \text{AF COMPLEX IN AF PRED} \rangle \mid$

$\langle \text{AF COMPLEX COMP PRED} \rangle \mid \langle \text{AF COMPLEX COMP AF PRED} \rangle$

pravidla $\langle \text{AF} * [\text{AF}] \text{PRED} \rangle$

$\langle \text{AF SOME Q. PRED} \rangle ::= \langle \text{FUNCTION} \rangle \langle \text{comp op} \rangle \mathbf{SOME} \langle \text{SUBQ} \rangle$

$\langle \text{AF ALL Q. PRED} \rangle ::= \langle \text{FUNCTION} \rangle \langle \text{comp op} \rangle \mathbf{ALL} \langle \text{SUBQ} \rangle$

$\langle \text{AF COMPLEX IN PRED} \rangle ::= \langle \text{FUNCTION} \rangle \mathbf{IN} \langle \text{SUBQ} \rangle$

$\langle \text{AF COMPLEX NOT IN PRED} \rangle ::= \langle \text{FUNCTION} \rangle \mathbf{NOT IN} \langle \text{SUBQ} \rangle$

$\langle \text{AF COMPLEX COMP PRED} \rangle ::= \langle \text{FUNCTION} \rangle \langle \text{comp op} \rangle \langle \text{SUBQ} \rangle$

analogicky s agragační funkcí v poddotazu:

$\langle \text{AF SOME Q. AF PRED} \rangle ::= \langle \text{FUNCTION} \rangle \langle \text{comp op} \rangle \mathbf{SOME} \langle \text{AF SUBQ} \rangle$

$\langle \text{AF ALL Q. AF PRED} \rangle ::= \langle \text{FUNCTION} \rangle \langle \text{comp op} \rangle \mathbf{ALL} \langle \text{AF SUBQ} \rangle$

$\langle \text{AF COMPLEX IN AF PRED} \rangle ::= \langle \text{FUNCTION} \rangle \mathbf{IN} \langle \text{AF SUBQ} \rangle$

$\langle \text{AF COMPLEX NOT IN AF PRED} \rangle ::= \langle \text{FUNCTION} \rangle \mathbf{NOT IN} \langle \text{AF SUBQ} \rangle$

$\langle \text{AF COMPLEX COMP AF PRED} \rangle ::= \langle \text{FUNCTION} \rangle \langle \text{comp op} \rangle \langle \text{AF SUBQ} \rangle$

$\langle \text{EXISTS PRED} \rangle ::= \mathbf{EXISTS} \langle \text{SUBQ} \rangle$

Překlad SQL => E3VPC

Pravidla překladu

- Výsledek překladu symbolu w je řetězec symbolů Θ_w .
- S některými pravidly gramatiky SQL jsou spojena překladová pravidla.
- Překladové pravidlo pro symbol w definuje Θ_w .
- Některá syntaktická pravidla mohou mít i více pravidel. Rozlišíme je indexem: Θ_i
- Použití závisí na kontextu.

Překlad (ne)terminálů

- Je-li w terminál, pak $\Theta w ::= w$.
- Význam přeložených SQL terminálů v E3VPC:

<alias>	proměnná
<column>	atribut
<table>	relace
<literal>	konstanta
<comp op>	operátor porovnání

Implicitní překlad

- Bud' $LHS ::= RHS$ syntaktické pravidlo, w_1, \dots, w_k neterminály na jeho pravé straně v pořadí zleva doprava.
- Necht' neexistuje pravidlo překladu pro $LHS ::= RHS$.
- Pak $\Theta LHS ::= \Theta w_1 \dots \Theta w_k$

syntaktické pravidlo:

$\langle \text{WHERE CLAUSE} \rangle ::= \mathbf{WHERE} \langle \text{WHERE CONDITION} \rangle$

překlad:

$\Theta \langle \text{WHERE CLAUSE} \rangle ::= \Theta \langle \text{WHERE CONDITION} \rangle$

Pravidla popsaná metajazykem

Θ_1 <FUNCTION>

::= odpovídající jméno agregační funkce a v závorkách její parametr

Θ_2 <FUNCTION>

::= hodnota odpovídající funkce pro prázdnou množinu

Θ "boolean expression of <X>"

::= odpovídající výraz v E3VPC, v němž jsou predikáty nahrazeny svými překlady

Θ "list of <X>"

::= seznam přeložených neterminálů oddělených čárkami

Θ <COL OR VAL>

::= konstanta nebo proměnná.atribut

⊖ <QUERY>

syntaktické pravidlo:

<QUERY> ::=

SELECT [**ALL** | **DISTINCT**] <SELECT LIST><FROM CLAUSE>
[<WHERE CLAUSE>][<GROUP BY CLAUSE>][<HAVING CLAUSE>]

pravidlo překladu:

⊖ <QUERY> ::=

{⊖ <FROM CLAUSE>: || ⊖ <WHERE CLAUSE> ^ ⊖ <HAVING CLAUSE> ||^F}

- negativní interpretace
- <GROUP BY CLAUSE> se projeví až při překladu <HAVING CLAUSE> a při překladu agregačních funkcí
- <SELECT LIST> není použit – jedná se o vnější SELECT

⊖ <* CLAUSE>

<FROM CLAUSE> ::= **FROM** "list of <TABLE REFERENCE>"

<TABLE REFERENCE> ::= <table> <alias>

<WHERE CLAUSE> ::= **WHERE** <WHERE CONDITION>

<HAVING CLAUSE> ::= **HAVING** <HAVING CONDITION>

<GROUP BY CLAUSE> ::= **GROUP BY** "list of <alias>.<column>"

pravidla překladač spojená s těmito syntaktickými pravidly:

⊖ <TABLE REFERENCE> ::= <alias> ∈ <table>

⊖ <GROUP BY CLAUSE> ::= <alias>.<column> ≅ <alias>↑.<column>

[^ <alias>.<column> ≅ <alias>↑.<column>] ...

zbývá definovat:

⊖ <WHERE CONDITION>, ⊖ <HAVING CONDITION>

⊕ <* CONDITION>

implicitní překlad a překlad popsany metajazykem:

<WHERE CONDITION> ::= "boolean expression of <WHERE PRED>"

<WHERE PRED> ::= <SIMPLE PRED> | <COMPLEX PRED>

<SIMPLE PRED> ::= <COL OR VAL> <comp op> <COL OR VAL>

<COMPLEX PRED> ::= ... | <ALL QUANTIFIED PRED> | ...

<HAVING CONDITION> ::= "boolean expression of <HAVING PRED>"

<HAVING PRED> ::= ... | <H AF COLUMN PRED> | ...

zbývá tedy přeložit:

<ALL QUANTIFIED PRED>, <H AF COLUMN PRED>, ...

Θ <ALL QUANTIFIED PRED>, ...

<ALL QUANTIFIED PRED> ::= <COL OR VAL> <comp op> **ALL** <SUBQ>

$\forall \Theta_1 \langle \text{SUBQ} \rangle \langle \text{COL OR VAL} \rangle \langle \text{comp op} \rangle \Theta_2 \langle \text{SUBQ} \rangle$

<SOME QUANTIFIED PRED> ::= <COL OR VAL> <comp op> **SOME** <SUBQ>

$\exists \Theta_1 \langle \text{SUBQ} \rangle \langle \text{COL OR VAL} \rangle \langle \text{comp op} \rangle \Theta_2 \langle \text{SUBQ} \rangle$

<COMPLEX IN PRED> ::= <COL OR VAL> **IN** <SUBQ>

$\forall \Theta_1 \langle \text{SUBQ} \rangle \langle \text{COL OR VAL} \rangle = \Theta_2 \langle \text{SUBQ} \rangle$

<COMPLEX NOT IN PRED> ::= <COL OR VAL> **NOT IN** <SUBQ>

$\forall \Theta_1 \langle \text{SUBQ} \rangle \langle \text{COL OR VAL} \rangle \neq \Theta_2 \langle \text{SUBQ} \rangle$

<EXISTS PRED> ::= **EXISTS** <SUBQ>

$\exists \Theta_1 \langle \text{SUBQ} \rangle$

...

Θ <SUBQ>

<SUBQ> ::=

SELECT [**ALL** | **DISTINCT**] <COL OR VAL> <FROM CLAUSE>
[<WHERE CLAUSE>][<GROUP BY CLAUSE>][<HAVING CLAUSE>]

výsledek poddotazu může být množina hodnot:

Θ_1 <SUBQ> =

$\{\Theta$ <FROM CLAUSE>: || Θ <WHERE CLAUSE> \wedge Θ <HAVING CLAUSE>||^F}

výsledek poddotazuje musí být jenom jedna hodnota:

Θ_2 <SUBQ> = <COL OR VAL>

Příklad

syntaktické pravidlo:

$\langle \text{ALL QUANTIFIED PRED} \rangle ::= \langle \text{COL OR VAL} \rangle \langle \text{comp op} \rangle \mathbf{ALL} \langle \text{SUBQ} \rangle$

překlad:

$\forall \Theta_1 \langle \text{SUBQ} \rangle \langle \text{COL OR VAL} \rangle \langle \text{comp op} \rangle \Theta_2 \langle \text{SUBQ} \rangle$

WHERE d.location = **ALL**

SELECT e.residence

FROM emp e

WHERE e.deptId = d.id

$\forall \{ e \in \text{emp} : \| e.\text{deptId} = d.\text{id} \| ^F \} d.\text{location} = e.\text{residence}$

Θ <H AF COLUMN PRED>, ...

<H AF COLUMN PRED> ::= <FUNCTION> <comp op> <COL OR VAL>

Θ_1 <FUNCTION>

{ Θ <FROM CLAUSE> : || Θ <WHERE CLAUSE> \wedge Θ <GROUP BY CLAUSE> ||^F}

<comp op>

<COL OR VAL>

<H AF FUNCTION PRED> ::= <FUNCTION> <comp op> <FUNCTION>

Θ_1 <FUNCTION>

{ Θ <FROM CLAUSE> : || Θ <WHERE CLAUSE> \wedge Θ <GROUP BY CLAUSE> ||^F}

<comp op>

Θ_1 <FUNCTION>

{ Θ <FROM CLAUSE> : || Θ <WHERE CLAUSE> \wedge Θ <GROUP BY CLAUSE> ||^F}

<H SIMPLE PRED> ::= <SIMPLE PRED>

\forall { Θ <FROM CLAUSE> : || Θ <WHERE CLAUSE> \wedge Θ <GROUP BY CLAUSE> ||^F}

Θ <SIMPLE PRED>

...

Příklad

syntaktické pravidlo:

$\langle \text{H AF COLUMN PRED} \rangle ::= \langle \text{FUNCTION} \rangle \langle \text{comp op} \rangle \langle \text{COL OR VAL} \rangle$

překlad:

$\Theta_1 \langle \text{FUNCTION} \rangle$

$\{ \Theta \langle \text{FROM CLAUSE} \rangle : \{ \Theta \langle \text{WHERE CLAUSE} \rangle \wedge \Theta \langle \text{GROUP BY CLAUSE} \rangle \}^F \}$

$\langle \text{comp op} \rangle$

$\langle \text{COL OR VAL} \rangle$

HAVING **AVG** (*d.nofEmp*) > 500

AVG(d.nofEmp)

$\{ \Theta \langle \text{FROM CLAUSE} \rangle : \{ \Theta \langle \text{WHERE CLAUSE} \rangle \wedge \Theta \langle \text{GROUP BY CLAUSE} \rangle \}^F \}$

> 500

ještě je třeba použít dříve získaných výsledků ...

- překlady získané dříve (na vyšší úrovni):

$\Theta\langle\text{FROM CLAUSE}\rangle ::= d \in \text{dept}$

$\Theta\langle\text{WH. CLAUSE}\rangle ::= \forall \{e \in \text{emp} : \parallel e.\text{deptId} = d.\text{id} \parallel^F\} d.\text{location} = e.\text{residence}$

$\Theta\langle\text{GROUP BY CLAUSE}\rangle ::= d.\text{manager} \cong d^\uparrow.\text{manager}$

- SQL:

SELECT d.manager

FROM dept d

WHERE d.location = **ALL**

SELECT e.residence **FROM** emp e **WHERE** e.deptId = d.id

GROUP BY d.manager

HAVING AVG (d.nofEmp) > 500

- celkový překlad neterminálu $\langle\text{H AF COLUMN PRED}\rangle$:

$\text{AVG}(d.\text{nofEmp})\{d \in \text{dept} :$

$\parallel (\forall \{e \in \text{emp} : \parallel e.\text{deptId} = d.\text{id} \parallel^F\} d.\text{location} = e.\text{residence}) \wedge$

$d.\text{manager} \cong d^\uparrow.\text{manager} \parallel^F\} > 500$

Dokončení příkladu

SQL:

```
SELECT d.manager  
FROM dept d  
WHERE d.location = ALL  
  SELECT e.residence  
  FROM emp e  
  WHERE e.deptId = d.id  
GROUP BY d.manager  
HAVING AVG (d.nofEmp) > 500
```

schéma:

```
dept(id, nofEmp, location, manager)  
emp(id, deptId, residence)
```

E3VPC:

$\{d \in \text{dept} : \|\ (\forall \{e \in \text{emp} : \| e.\text{deptId} = d.\text{id} \| ^F\} d.\text{location} = e.\text{residence}) \wedge$

$\text{AVG}(d.\text{nofEmp})\{d \in \text{dept} :$

$\|\ (\forall \{e \in \text{emp} : \| e.\text{deptId} = d.\text{id} \| ^F\} d.\text{location} = e.\text{residence}) \wedge$
 $d.\text{manager} \cong d \uparrow .\text{manager} \| ^F\} > 500$

$\| ^F\}$

Ekvivalence SQL dotazů

Cíl

- Zjistit, zda dva dané SQL výrazy jsou ekvivalentní.
- Použijeme k tomu E3VPC
- Oba dotazy přeložíme a zjistíme ekvivalenci E3VPC výrazů.
- Jak?

Kanonická forma E3VPC výrazu

E3VPC výraz je *kanonický*, pokud

1. interpretační operátor je aplikován na každou atomickou formuli
2. interpretační operátor není aplikován na jinou než atomickou formuli
3. neobsahuje zkrácené kvantifikované výrazy:

$$\forall \{x \in S: P(x)\} Q(x), \exists \{x \in S: P(x)\} Q(x)$$

Pozorování

- S kanonickým výrazem lze pracovat ve 2VPC.
- Ve 2VPC umíme zjistit ekvivalenci dvou výrazů.
- Má každý E3VPC výraz svůj kanonický tvar?
- Ano – následuje lemma, jejichž násobnou aplikací dostaneme kanonickou formu výrazu

Lemma 1

1. $\|P(x) \vee Q(x)\|^\alpha \Leftrightarrow \|P(x)\|^\alpha \vee \|Q(x)\|^\alpha$
2. $\|P(x) \wedge Q(x)\|^\alpha \Leftrightarrow \|P(x)\|^\alpha \wedge \|Q(x)\|^\alpha$
3. $\neg\|P(x)\|^\alpha \Leftrightarrow \|\neg P(x)\|^{\neg\alpha}$
4. $\|\|P(x)\|^\alpha\|^\beta \Leftrightarrow \|P(x)\|^\alpha$
5. $\|(\exists x \in S): P(x)\|^\alpha \Leftrightarrow (\exists x \in S): \|P(x)\|^\alpha$
6. $\|(\forall x \in S): P(x)\|^\alpha \Leftrightarrow (\forall x \in S): \|P(x)\|^\alpha,$
kde $P(x), Q(x)$ jsou formule, S relace, $\alpha, \beta \in \{T, F\}$

Lemma 2

1. $\|\exists\{x \in S: \|P(x)\|^\alpha\} Q(x)\|^\beta \Leftrightarrow (\exists x \in S): \|P(x)\|^\alpha \wedge \|Q(x)\|^\beta$
 2. $\|\forall\{x \in S: \|P(x)\|^\alpha\} Q(x)\|^\beta \Leftrightarrow (\forall x \in S): \|\neg P(x)\|^\alpha \vee \|Q(x)\|^\beta$,
- kde $P(x)$, $Q(x)$ jsou formule, S relace, $\alpha, \beta \in \{T, F\}$

důkaz 1: $\|\exists\{x \in S: \|P(x)\|^\alpha\} Q(x)\|^\beta \Leftrightarrow^{\text{PC}}$
 $\|(\exists x \in S): \|P(x)\|^\alpha \wedge Q(x)\|^\beta \Leftrightarrow^5$
 $(\exists x \in S): \| \|P(x)\|^\alpha \wedge Q(x)\|^\beta \Leftrightarrow^2$
 $(\exists x \in S): \| \|P(x)\|^\alpha \|^\beta \wedge \|Q(x)\|^\beta \Leftrightarrow^4$
 $(\exists x \in S): \|P(x)\|^\alpha \wedge \|Q(x)\|^\beta$

qed.

Závěr

- Definovali jsme jazyk E3VPC.
- Ukázali jsme si gramatiku SQL (zjednodušeně).
- Víme, že lze sestavit syntaxí řízený překladač
SQL \Rightarrow E3VPC.
- Viděli jsme překlad některých částí SQL dotazů.
- Umíme rozhodnout, zda jsou dva SQL dotazy ekvivalentní.

Literatura

M. Negri, G. Pelagatti, L. Sbattella:
Formal Semantics of SQL Queries