

Relačné dotazovacie jazyky

- ISBL ← Peterlee relational test vehicle (IBM Peterlee UK)
 - QUEL INGRES ← algebra
 - QBE (IBM Yorktown Heights) ← n-ticový kalkul
 - SQL (IBM San Jose) ← kalkul
- Aritmetika (nekonečné relácie, vstupná množina)
- Priradovacie a definičné príkazy
- Agregáčn é funkcie
- Administrácia a pomocné funkcie

Relačné jazyky

1

Porovnanie jazykov založených na algebre a jazykov založených na kalkulácii

- $\{ C : (\exists B) (r(A, B) \wedge s(B, C)) \}$ 1
 - $\{ C : (\exists B) (r(A, B) \wedge s(B, C) \wedge (A = a)) \}$ 2
 - $\Pi_C (\Pi_{AC} (\sigma_{A=a} r(A, B)) \bowtie s(B, C))$ 3
 - $\Pi_C (\sigma_{A=a} r(A, B)) \bowtie s(B, C)$ 4
- Algebra špecifikuje výpočet (je čiastočne operačná)
- Kalkul je čiste deklaratívna špecifikácia.
- $q(C) \leftarrow r(a, B), s(B, C)$ (datalog, prolog) 5

Relačné jazyky

2

ISBL - information system base language

Relačná algebra	ISBL
$R \cup S$	$R + S$
$R - S$	$R - S$
$R \wedge S$	$R . S$
$\sigma_F R$	$R : F$
$\Pi_{A_1, \dots, A_n} R$	$R \% A_1, \dots, A_n$
$R \bowtie S$	$R * S$
Premenovanie:	$R \% \dots, A \rightarrow B, \dots$
Priradenie:	=
Odloženie vyhodnotenia:	N!
Tlač:	List

Relačné jazyky

3

QUEL - n-ticovo orientovaný kalkul

Schéma príkazov:

range of μ_1 is R_1

·
·
·

range of μ_k is R_k

[retrieve | delete | append to]

where $\Psi(\mu_1, \dots, \mu_k)$

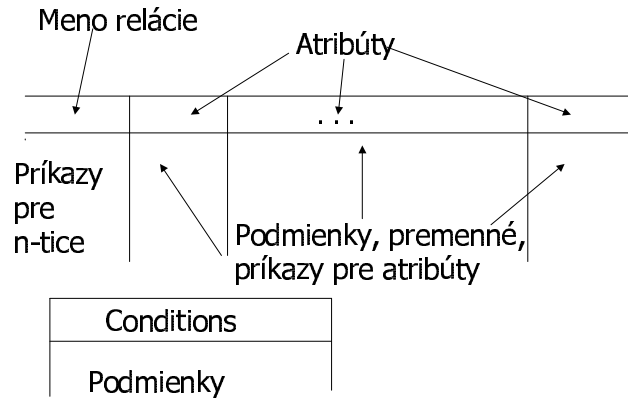
sort

print

Relačné jazyky

4

QBE - query by example



Relačné jazyky

5

SQL - structured query language

Schéma dotazu:

Select [**distinct**] [* | zoznam atribútov]

From [zoznam atribútov a ich aliasov]

[**where** podmienky na atribúty]

[**group by** zoznam atribútov]

[**having** podmienky na agregácie]

[**order by** zoznam]]

Relačné jazyky

6

Ďalšie konštrukcie SQL I

- Aritmetika, reťazcové operácie

+, **-**, *****, **/**, **substring(s, i, j)**, **like 'x%'**

- Agregáčnne funkcie:

count, **sum**, **average**, **min**, **max**

- Dáta definičné príkazy

Create [**table** | **view** | **index**] názov
(deklarácia atribútov)

Drop [**table** | **view** | **index**] názov

Relačné jazyky

7

Ďalšie konštrukcie SQL II

- Dáta manipulačné príkazy

Insert into tabuľka
select

Insert into tabuľka
values

Update tabuľka
set priradenia atribútom
where

Delete from tabuľka
where

Relačné jazyky

8

Administratívne príkazy

- **attach** názov databázy
- **set** transaction read only
- **set** line length
- ukončenie transakcie
commit | rollback
- **disconnect**

Relačné jazyky

9

SQL vložené do hostiteľského jazyka

Príkazy SQL možno vykonávať z hostiteľského jazyka a môžu byť prepojené na hostiteľský jazyk (napr. C, Pascal).

Problém je, že SQL je množinovo orientovaný a programovacie jazyky pracujú s individuálnymi premennými.

Odvzdávanie premenných:

Cursor: to isté ako view, ale dá sa pristupovať k jednotlivým riadkom pomocou operácie **FETCH** (**next | prior**).

V programe sú vložené príkazy označené **EXEC SQL**.
Premenné zabezpečujúce väzbu začínajú **dvojbodkou**.
Preprocesor preloží vložené príkazy do volania pripravených procedúr.

Relačné jazyky

10

Príklady príkazov embeded SQL

- Exec SQL execute immediate S
 - Exec SQL prepare S from :c
 - Exec SQL execute S using :a₁, ..., :a_k
- } Nie dotaz
- ```
Exec SQL prepare S from :abraka;
Exec SQL declare C cursor for S;
Exec SQL open C;
Exec SQL whenever not found goto NOMORE;
while (true) {Exec SQL Fetch C into :a1, ..., :ak; ... };
NOMORE: Exec SQL close C;
```

Relačné jazyky

11

## Príklady: (pôvodné Coddové)

Schéma (databázy), na ktorú sa vzťahujú príklady:

### Dodávatelia

| Meno | Adresa | Mesto | ČísDod |
|------|--------|-------|--------|
|      |        |       |        |

### Súčiastky

| ČísSúč | Názov | Farba |
|--------|-------|-------|
|        |       |       |

### Dodáva

| ČísDod | ČísSúč |
|--------|--------|
|        |        |

Relačné jazyky

12

### 1. Zoznam dodávateľov (meno, mesto), ktorých každný niečo dodáva.

**kalkul**  $(\exists \text{ adresa, ČísDod, ČísSúč})$   
 Dodávateľia(meno, adresa, mesto, ČísDod)  $\wedge$   
 Dodáva(ČísDod, ČísSúč)

**algebra**  $\Pi_{\text{meno, mesto}} (\text{Dodávateľia} \bowtie \text{Dodáva})$

**SQL**  
**Select distinct** meno, mesto  
**from** Dodávateľia, Dodáva  
**where** Dodávateľia.ČísDod = Dodáva.ČísDod

**QBE**

|             |        |        |        |        |
|-------------|--------|--------|--------|--------|
| Dodávateľia | Meno   | Adresa | Mesto  | ČísDod |
|             | P!     |        | P!     | _123   |
|             | Dodáva | ČísDod | ČísSúč |        |
|             |        | _123   |        |        |

Relačné jazyky 13

### 2. Zoznam dodávateľov (meno, mesto), ktorí nič nedodávajú.

**kalkul**  $(\exists \text{ adresa, ČísDod}) (\forall \text{ ČísSúč})$   
 Dodávateľia(meno, adresa, mesto, ČísDod)  $\wedge \neg$   
 Dodáva(ČísDod, ČísSúč)

**algebra**  $\Pi_{\text{meno, mesto}} (\text{Dodávateľia} \bowtie (\Pi_{\text{ČísDod}} \text{Dodávateľia} - \Pi_{\text{ČísDod}} \text{Dodáva}))$

**SQL**  
**Select distinct** meno, mesto  
**from** Dodávateľia  
**where** ČísDod **not in** ( **select** ČísDod **from** Dodáva )

**QBE**

|             |        |        |        |        |
|-------------|--------|--------|--------|--------|
| Dodávateľia | Meno   | Adresa | Mesto  | ČísDod |
|             | P!     |        | P!     | _123   |
|             | Dodáva | ČísDod | ČísSúč |        |
|             |        | _123   |        |        |

Relačné jazyky 14

### 3. Číslo dodávateľov, ktorí dodávajú súčiastku číslo "15".

**kalkul** Dodáva(ČísDod, "15")

**algebra**  $\Pi_{\text{ČísDod}} \sigma_{\text{ČísSúč}="15"} \text{Dodáva}$

**SQL**  
**Select** ČísDod  
**from** Dodáva  
**where** ČísSúč = "15"

**QBE**

|        |        |        |
|--------|--------|--------|
| Dodáva | ČísDod | ČísSúč |
|        | P!     | "15"   |

Relačné jazyky 15

### 4. Číslo dodávateľov, ktorí dodávajú niečo, čínejšie súčiastka číslo "15".

**kalkul**  $(\exists x) \text{Dodáva}(\text{ČísDod}, x) \wedge (x \neq "15")$

**algebra**  $\Pi_{\text{ČísDod}} \sigma_{\text{ČísSúč} \neq "15"} \text{Dodáva}$

**SQL**  
**Select distinct** ČísDod  
**from** Dodáva  
**where** ČísSúč  $\neq$  "15"

**QBE**

|        |        |             |
|--------|--------|-------------|
| Dodáva | ČísDod | ČísSúč      |
|        | P!     | $\neq$ "15" |

Relačné jazyky 16

### 5. Číslo dodávateľa, ktorým nedodávajú súčiastku číslo "15".

**kalkul**  $(\exists \text{ meno, adresa, mesto})$   
 Dodávateľa(meno, adresa, mesto, ČísDod)  $\wedge$   
 $(\forall \text{ ČísSúč}) \text{ Dodáva}(\text{ČísDod}, \text{ČísSúč}) \Rightarrow \text{ČísSúč} \neq "15"$

**algebra**  $\Pi_{\text{ČísDod}} \text{ Dodávateľa} - \Pi_{\text{ČísDod}} \sigma_{\text{ČísSúč} = "15"} \text{ Dodáva}$

**SQL** **Select** ČísDod **from** Dodávateľa  
**where** ČísDod **not in**  
 ( **select** ČísDod **from** Dodáva **where** ČísSúč = "15" )

**QBE**

| Dodávateľa | Meno | Adresa | Mesto | ČísDod |
|------------|------|--------|-------|--------|
|            |      |        |       | PI_123 |

| Dodáva | ČísDod | ČísSúč |
|--------|--------|--------|
| ¬      | _123   | 15     |

Relačné jazyky

17

### 6. Číslo dodávateľa, ktorým dodávajú aj niečo krem súčiastku číslo "15".

**kalkul**  $(\exists x)( \text{ Dodáva}(\text{ČísDod}, "15") \wedge$   
 $\text{ Dodáva}(\text{ČísDod}, x) \wedge (x \neq "15"))$

**algebra**  $\Pi_{\text{ČísDod}} \sigma_{\text{ČísSúč} = "15"} \text{ Dodáva} \bowtie \Pi_{\text{ČísDod}} \sigma_{\text{ČísSúč} \neq "15"} \text{ Dodáva}$

**SQL** **Select distinct** ČísDod **from** Dodáva Y  
**where** Dodáva.ČísDod = Y.ČísDod **and**  
 Dodáva.ČísSúč = "15" **and** Y.ČísSúč  $\neq$  "15"

**QBE**

| Dodáva | ČísDod | ČísSúč | Dodáva | ČísDod | ČísSúč      |
|--------|--------|--------|--------|--------|-------------|
|        | PI_123 | "15"   |        | _123   | $\neq$ "15" |

Relačné jazyky

18

### 7. Číslo dodávateľa, ktorým dodávajú len súčiastku číslo "15".

**kalkul**  $(\forall x) \text{ Dodáva}(\text{ČísDod}, x) \Rightarrow (x = "15")$

**algebra**  $\Pi_{\text{ČísDod}} \sigma_{\text{ČísSúč} = "15"} \text{ Dodáva} - \Pi_{\text{ČísDod}} \sigma_{\text{ČísSúč} \neq "15"} \text{ Dodáva}$

**SQL** **Select distinct** ČísDod **from** Dodáva  
**group by** ČísDod  
**having set** ČísSúč **in set** ("15")

**QBE**

| Dodáva | ČísDod | ČísSúč | Dodáva | ČísDod | ČísSúč      |
|--------|--------|--------|--------|--------|-------------|
|        | PI_123 | "15"   | ¬      | _123   | $\neq$ "15" |

Relačné jazyky

19

### 8. Číslo dodávateľa, ktorým dodávajú niečo, ale nedodávajú súčiastku číslo "15".

**kalkul**  $(\forall x) \text{ Dodáva}(\text{ČísDod}, x) \wedge (x \neq "15")$

**algebra**  $\Pi_{\text{ČísDod}} \text{ Dodáva} - \Pi_{\text{ČísDod}} \sigma_{\text{ČísSúč} = "15"} \text{ Dodáva}$

**SQL** **Select distinct** ČísDod **from** Dodáva  
**where** ČísDod **not in**  
 ( **select** ČísDod **from** Dodáva **where** ČísSúč = "15" )

**QBE**

| Dodáva | ČísDod | ČísSúč | Dodáva | ČísDod | ČísSúč |
|--------|--------|--------|--------|--------|--------|
|        | PI_123 |        | ¬      | _123   | "15"   |

Relačné jazyky

20

9. Číslododávateľov, ktorí dodávajú aspoň súčiastky číslo "12", "13", "15".

**kalkul**  $\text{Dodáva}(\text{ČísDod}, "12") \wedge \text{Dodáva}(\text{ČísDod}, "13") \wedge \text{Dodáva}(\text{ČísDod}, "15")$

**algebra**  $\text{Dodáva} : \langle \text{ČísSúč}, \{12, 13, 15\} \rangle$

**SQL** **Select** ČísDod **from** Dodáva **group by** ČísDod **having set** ČísSúč **contains** {12,13,15}

**QBE**

| Dodáva | ČísDod  | ČísSúč |
|--------|---------|--------|
|        | PI _123 | "12"   |
| ^      | _123    | "13"   |
| ^      | _123    | "15"   |

Relačné jazyky

21

10. Číslododávateľov, ktorí dodávajú všetky dodávané súčiastky.

**kalkul**  $(\forall x)(\exists y) \text{Dodáva}(\text{ČísDod}, x) \Rightarrow \text{Dodáva}(y, x)$

**algebra**  $\text{Dodáva}(\text{ČísDod}, \text{ČísSúč}) : \Pi_{\text{ČísSúč}} \text{Dodáva}$

$\Pi_{\text{ČísDod}} \text{Dodáva} -$

$\Pi_{\text{ČísDod}} ((\Pi_{\text{ČísDod}} \text{Dodáva} \times \Pi_{\text{ČísSúč}} \text{Dodáva}) - \text{Dodáva})$

**SQL** **Select** ČísDod **from** Dodáva **group by** ČísDod **having set** ČísSúč **contains** (select ČísSúč from Dodáva)

Relačné jazyky

22

11. Zoznam miestokiaľ prichádza aspoň jedna "červená" súčiastka.

**kalkul**  $(\exists \text{meno, adresa, ČísDod, ČísSúč, názov})$   
 $\text{Dodávateľia}(\text{meno, adresa, mesto, ČísDod}) \wedge$   
 $\text{Dodáva}(\text{ČísDod}, \text{ČísSúč}) \wedge$   
 $\text{Súčiastky}(\text{ČísSúč, názov, "červená"})$

**algebra**  $\Pi_{\text{mesto}} (\text{Dodávateľia} \bowtie \text{Dodáva} \bowtie (\sigma_{\text{farba}="červená"} \text{Súčiastka}))$

**SQL** **Select** mesto **from** Dodávateľia, Dodáva, Súčiastka **where** farba = "červená"

Relačné jazyky

23

## Delenie v SQL

**Select** ČísDod **from** Dodáva **group by** ČísDod **having set** ČísSúč **contains** (select ČísSúč from Dodáva)

**Select** ČísDod **from** Dodáva **where** ČísDod **not in** **select** Dodáva.ČísDod **from** Dodáva X **where** Dodáva.ČísDod, X.ČísSúč **not in** **select** \* **from** Dodáva

Relačné jazyky

24

## Datalóg – logika a kódový model

- Základný princíp definuje sa najprv pohľad – program. Potom dotaz – dotaz na čiastočnú zhodu na niektorý predikát definovaného pohľadu.
- Syntax datalógu je podobná syntaxy jazyku logického programovania – prológu.

Príklad:

šéf(Vedúci, Zamestnanec) :- vedie (Vedúci, Zamestnanec).

šéf(Vedúci, Zamestnanec) :-

šéf(Vedúci, Z), vedie(Z, Zamestnanec).

? – šéf(X, peter).

- Premenné
- Databázové – extenzionálne predikáty
- Definované – intencionálne predikáty
- Zabudované predikáty (=, <, >, ...)

Relačné jazyky

25

## Terminológia – klauzuly a formuly

Fakt = (pozitívna) **atomická formula**

- $p(a_1, a_2, \dots, a_n)$
- $p(X_1, X_2, \dots, X_n)$

Literál = **atomická formula** alebo **negovaná atomická formula** ( $\neg p, p$ )

Klauzula = **logický súčet** literálov (literály pospájané logickou spojkou  $\vee$  (**or**)).

Hornova klauzula = klauzula obsahujúca **najviac jednu** (negovanú) **atomickú formulu**.

Hornove klauzuly:

- len 1 atomická formula – fakt
- zjednotenie negovaných atomických formúl – podmienka
- práve jedna negovaná atomická formula

$$P_0 \vee P_1 \vee \dots \vee P_k \equiv P_1 \wedge \dots \wedge P_k \Rightarrow P_0 \equiv P_0 :- P_1, \dots, P_k$$

Relačné jazyky

26

## Príklady použitia klauzálnej logiky

Databázová relácia *rodič(x, y)*

*súrodenci(x, y) :- rodič(z, x), rodič(z, y), x ≠ y.*

*bratraci(x, y) :- rodič(u, x), rodič(v, y), súrodenci(u, v).*

*bratraci(x, y) :- rodič(u, x), rodič(v, y), bratraci(u, v).*

*príbuzní(x, y) :- súrodenci(x, y).*

*príbuzní(x, y) :- príbuzní(x, z), rodič(z, y).*

*príbuzní(x, y) :- príbuzní(z, y), rodič(z, x).*

Implementácia aritmetiky zabudovaných funkcií

$p(x, z) :- q(x, u, v), z = (u+v)/5$ . alebo presnejšie

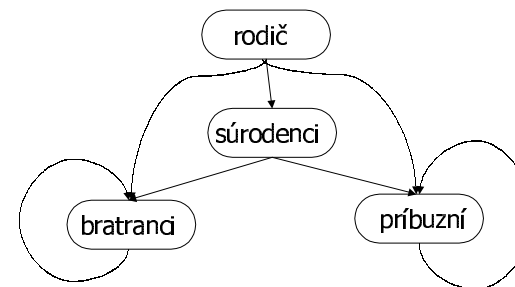
$p(x, z) :- q(x, u, v), add(u, v, y), div(y, 5, z).$

V zabudovaných funkciách predikátoch treba „strážit“ vstupné množiny.

Relačné jazyky

27

## Graf závislosti predikátov



Uzly sú predikáty. Hrana z uzla  $p_1$  do uzla  $p_2$ , ak predikát  $p_2$  je definovaný pomocou predikátu  $p_1$ . Existuje pravidlo s hlavou  $p_1$  ktorého telo obsahuje  $p_2$ .

Relačné jazyky

28

## Bezpečné pravidlá – konečné premenné

- Premenná je konečná ak sa vyskytuje v obyčajnom (nenegovanom, nezabudovanom predikáte) tele pravidla alebo v predikáte  $x = a$ .
- Ak v predikáte  $x = y$  jedna premenná je konečná, potom aj druhá je konečná.

Pravidlo je bezpečné, ak všetky jeho premenné sú konečné.  
Dôsledok: Každá premenná v hlave sa musí vyskytovať aj v tele pravidla.

Pravidlo je **rekurzívne**, keď predikát hlavy sa vyskytuje aj v tele pravidla. Program je rekurzívny, keď graf závislosti predikátov obsahuje cyklus.

## Výpočet nerekurzívnych programov

Každé pravidlo prerobíme na relačný výraz: náhradou čiarok za prirodzené spojenia.  
Problémom je, keď viac pravidiel má ten istý predikát v hlave. V takomto prípade musíme vytvoriť „rektifikované“ pravidlá:

Predikáty v hlave obsahujú tie isté premenné a iba premenné.

Premenovanie, zavedenie premennej

$\rho(x, a) :- \dots$

$\rho(x, y) :- \dots, y = a.$

Výsledný výraz pre predikát  $\rho$  je zjednotenie výrazov, pre telá rektifikovaných pravidiel s hlavou  $\rho$ .

Poradie výpočtu predikátov je dané topologickým utriedením grafu závislosti predikátov.

Rektifikáciu a úpravu na algebraické výrazy môžeme urobiť aj pre rekurzívne programy – **systém rovníc v relačnej algebre**.

## Tarského veta o pevnom bode

Úplný zväz:  $S = \langle D, \sqsubseteq, \sqcup, \sqcap, \perp, \top \rangle$

$\sqsubseteq$  čiastočné usporiadanie

$\sqcup$  l.u.b., sup, join       $\perp$  dolník

$\sqcap$  g.u.b., inf, meet       $\top$  horník

Každá neprázdna množina má l.u.b. (suprémum).

Veta: Nech  $F$  je zobrazenie z úplného zväzu  $S$  do  $S$  také, že  $x \sqsubseteq y \Rightarrow F(x) \sqsubseteq F(y)$ . Potom  $F$  má aspoň jeden pevný bod.

Dôkaz: Nech  $U = \{x : x \sqsubseteq F(x)\}$ . Množina  $U$  je neprázdna,  $\perp \in U$ .

Označme:  $x^0 = \perp$ . Pre každé  $x \in U$  platí  $x \sqsubseteq F(x) \sqsubseteq x_0$ .

Preto aj  $x_0 \sqsubseteq F(x_0)$  t.j.  $x_0 \in U$ . Preto aj  $F(x_0) \sqsubseteq F(F(x_0))$  a

$F(x_0) \in U$ . Z toho ale plynie  $F(x_0) \sqsubseteq x_0$ . Teda  $x_0 = F(x_0)$ .

## Výpočet rekurzívnych programov bez negácie

Systém rovníc:  $\vec{P} = \vec{E}(\vec{P}, \vec{R})$ , kde  $\vec{P}$  sú intencionálne a  $\vec{R}$  extenzionálne predikáty.

Riešenie: najnovou iteráciou. Začneme  $\vec{P}_0 = \vec{0}$ . Postupne počítame postupne  $P_1, P_2, \dots$ , podľa vzorca:

$$\vec{P}_i = \vec{E}(\vec{P}_{i-1}, \vec{R}),$$

pokiaľ  $\vec{P}_i \neq \vec{P}_{i+1}$ .

Podľa Tarského vety o pevnom bode tento proces vždy skončí (konverguje). Stačí overiť, že prirodzené spojenie, zjednotenie, projekcia a premenovanie sú „neklesajúce“ operácie.

Vypočítané riešenie je najmenší pevný bod uvedeného systému rovníc. Z vlastností implikácií plynie, že každé riešenie uvedeného systému rovníc musí obsahovať vety vypočítané našim algoritmom.



## Inkrementálny výpočet

$$P_i = P_{i-1} \cup \Delta_i$$

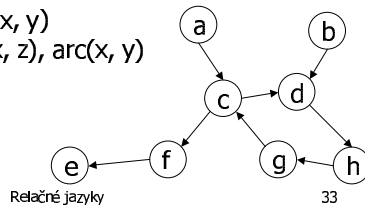
Pri naivnej iterácii sa vyhodnocuje opakovane v každom cykle.

Spojenie:

$$(P \cup \Delta) \bowtie (R \cup \Gamma) = P \bowtie R \cup R \bowtie \Delta \cup P \bowtie \Gamma \cup \Delta \bowtie \Gamma$$

Seminaivná iterácia: Všetky relácie sa v každom iteračnom kroku rozdelia na nové a staré. Počítajú sa len prirodzené spojenia obsahujúce aspoň jeden prírastkový operand.

Príklad:  $tc(x, y) :- arc(x, y)$   
 $tc(x, y) :- tc(x, z), arc(x, y)$



Relačné jazyky

33

## Negácia – stratifikované programy

Rozdiel, negácia nie sú neklesajúce operácie. Vo všeobecnosti programy s negáciou nemusia mať pevný bod.

**Stratifikované programy:** Ak definícia predikátu  $p$  obsahuje negáciu predikátu  $q$ , potom v grafe závislosti predikátov neexistuje cesta od  $p$  ku  $q$ .

**Podmienky stratifikácie:**

- Každému predikátu je priradené celé číslo stratum (vrstva)
- Ak pravidlo  $p :- \dots q \dots$ . Potom  $S(p) \geq S(q)$ .
- Ak pravidlo  $p :- \dots \neg q \dots$ . Potom  $S(p) > S(q)$ .

**Postup stratifikácie:** Na začiatku, priradíme všetkým predikátom stratum 1 a spočítame predikáty -  $n$ . Prezeráme postupne pravidlá. Ak je porušená niektorá s podmienok stratifikácie, zvýšime stratum predikátu hlavy na najmenšie prirodzené číslo, ktoré ju spĺňa. Ak stratum väčšie ako  $n$ , program sa nedá stratifikovať. Opakuj zakiaľ sa niečo mení.

Relačné jazyky

34

## Princípbyrokracie predpokladuzavretéhosveta.

Striktne vzaté z Hornových formúl sa nedá odvodiť negácia žiadnej atomickej formuly (negatívny fakt). Aby sme prípadne mohli odvodiť negatívny fakt aplikujeme princíp uzavretého sveta:

**Čo sa nedá dokázať z bázy dát neplatí.**

**Quod non est in datis, non est in mundi.**

Pôvodne: **actis**

Princíp uzavretého sveta vedie k nekonzistencii pri aktualizácii bázy dát. Logika negácii nie je monotónna.

Relačné jazyky

35

## Irépokusyonegáciu

Negácia pomocou protirečivosti (negation as inconsistency)

$\neg p(x)$  vtedy, keď  $db \vee p(x)$  je nekonzistentné.

Toto funguje, ale množina Hornových formúl je vždy neprotirečivá. Potrebujeme nejaké negatívne fakty v báze dát.

$p^+$ ,  $p^-$  Intencia je  $p^+$  pre pozitívne fakty a  $p^-$  pre negatívne fakty. Podmienka  $\neg p^+ \vee \neg p^-$ .

Takáto negácia je konzistentná, ale dosť slabá. (Matematicky ekvivaentná intuicionistickej logike). Všetko sa dá nasimulovať v datalógu pomocou „rozdvojenia“ predikátov.

Relačné jazyky

36