Databázové rozcvičky

Tomáš Plachetka
Mária Pastorová
Jana Katreniaková
Stanislav Miklík
Marián Sládek

2011

Tento text obsahuje príklady z 5-minútových "rozcvičiek" z cvičení Úvod do databázových systémov, spolu s riešeniami. Text je niekedy rozšírený o variácie pôvodného zadania.

EDB je skratka pre "extensional database" (v relačnom kalkule je to množina predikátov, ktoré tvoria databázu faktov).

Dôležité je naučiť sa písať dotazy najmä v Datalogu a v relačnom kalkule. Dotazy v iných jazykoch sa dajú potom získať automatickým prekladom—aj keď je praktické naučiť sa písať resp. čítať dotazy priamo aj v iných jazykoch. Z tohto dôvodu (a hlavne kvôli lenivosti autorov) v niektorých riešeniach chýba zápis v SQL či v relačnej algebre.

1 Rozcvičky 2011

Príklad 1.1 *Dané sú predikáty clovek(Meno), kope(Kto, Komu, Jama), pada(Meno, Jama). Nájdite mená všetkých ľudí, pre ktorých platí "Kto druhému jamu kope, sám do nej padá."*

Relačný kalkul:
C: clovek(C) $\wedge$ $\forall$J
(
($\exists$C2 kope(C, C2, J)) $\Rightarrow$ pada(C, J)
)

Alebo: C: clovek(C) $\wedge$ $\forall$J$\forall$C2
(
kope(C, C2, J) $\Rightarrow$ pada(C, J)
)

Alebo: C: clovek(C) $\wedge$ $\neg$ $\exists$J
(
($\exists$C2 kope(C, C2, J)) $\wedge$ $\neg$ pada(C, J)
)

2 Rozcvičky 23.9.2009–24.9.2009

Príklad 2.1 *Nájdite množinu párnych čísiel.*
EDB={integer(X), multiply(X, Y, Result)}.

Relačný kalkul:
$\{X: integer(X) \wedge \exists\ Y\ (integer(Y) \wedge multiply(2, Y, X))\}$

Alebo:
$integer(X) \wedge \exists\ Y\ (integer(Y) \wedge multiply(2, Y, X))$

Datalog:
$result(X) \leftarrow integer(X), integer(Y), multiply(2, Y, X).$

SQL:

```
select i1.X
from integer i1, integer i2, multiply m
where multiply.X = 2 and multiply.Y = i2.X and i1.X = multiply.Result
```

Príklad 2.2 *Nájdite dvojice nesúdeliteľných čísiel.*
EDB={integer(X), multiply(X, Y, Result)}.

Relačný kalkul:
$\{[X, Y]: integer(X) \wedge integer(Y) \wedge \forall\ P \forall\ Q \forall\ K$
$((integer(P) \wedge integer(Q) \wedge integer(K) \wedge multiply(K, P, X) \wedge multiply(K, Q, Y))$
$\Rightarrow K{=}1)\}$

Alebo:
$integer(X) \wedge integer(Y) \wedge \forall\ P \forall\ Q \forall\ K$
$((integer(P) \wedge integer(Q) \wedge integer(K) \wedge multiply(K, P, X) \wedge multiply(K, Q, Y))$
$\Rightarrow K{=}1)$

Alebo:
$integer(X) \wedge integer(Y) \wedge \neg\ (\exists\ P \exists\ Q \exists\ K$
$(integer(P) \wedge integer(Q) \wedge integer(K) \wedge multiply(K, P, X) \wedge multiply(K, Q, Y)$
$\wedge\ K \neq 1))$

Datalog:
$sudelitelne(X, Y) \leftarrow integer(X), integer(Y), multiply(K, P, X), multiply(K, Q, Y),$
$K <> 1.$
$result(X, Y) \leftarrow integer(X), integer(Y), not\ sudelitelne(X, Y).$

SQL:

```
create temporary table sudelitelne as select i1.X, i2.X
from integer i1, integer i2, multiply m1, multiply m2
where i1.X = m1.Result and i2.X = m2.Result and m1.X = m2.X and m1.X <> 1

select i1.X, i2.X
from integer i1, integer i2
where not exists (
     select *
     from sudelitelne s
     where i1.X = s.X and i2.X = s.Y
)
```

Príklad 2.3 *Nájdite ľudí, ktorí sú strojcami svojho vlastného šťastia.*
EDB={ludia(Meno), stroji_stastie(Kto, Komu)}.

Relačný kalkul:
{M: ludia(M) ∧ stroji_stastie(M, M)}

Alebo:
ludia(M) ∧ stroji_stastie(M, M)

Datalog:
result(M) ← ludia(M), stroji_stastie(M, M).

SQL:
```
select l.Meno
from ludia l, stroji_stastie ss
where l.Meno = ss.Kto and l.Meno = ss.Komu
```

Relačná algebra:
ludia $\bowtie_{Meno=Kto}$ ($\sigma_{Kto=Komu}$(stroji_stastie))

Príklad 2.4 *Každý (človek) je strojcom svojho vlastného šťastia.*
EDB={ludia(Meno), stroji_stastie(Kto, Komu)}.

Relačný kalkul:
∀ M (ludia(M) ⇒ stroji_stastie(M, M))

Alebo (nie je pravda, že existuje nejaký človek, ktorý nestrojí šťastie sám sebe):
¬ (∃ M (ludia(M) ∧ ¬ stroji_stastie(M, M)))

Datalog:
```
existuje_clovek_ktory_nestroji ← ludia(M), not stroji_stastie(M, M).
result ← not existuje_clovek_ktory_nestroji.
```

SQL (trochu pritiahnuté za vlasy, keďže výsledkom je jednoriadková tabuľka, ktorá obsahuje TRUE, resp. FALSE):
```
create temporary table existuje_clovek_ktory_nestroji as
select exists (
    select *
    from ludia l
    where not exists (
        select *
        from stroji_stastie ss
        where l.Meno = ss.Kto and l.Meno = ss.Komu
    )
)

select not (
    select *
    from existuje_clovek_ktory_nestroji eckn)
)
```

Príklad 2.5 *Všetci politici sú nečestní.*
EDB={politik(Meno), cestny(Meno)}.

Relačný kalkul:
$\forall$ M (politik(M) $\Rightarrow$ ($\neg$ cestny(M)))

Alebo (neexistuje čestný politik):
$\neg$ ($\exists$ M (politik(M) $\wedge$ cestny(M)))

Datalog:
existuje_cestny_politik $\leftarrow$ politik(M), cestny(M).
result $\leftarrow$ not existuje_cestny_politik.

SQL (trochu pritiahnuté za vlasy, keďže výsledkom je jednoriadková tabuľka, ktorá obsahuje TRUE, resp. FALSE):

```
create temporary table existuje_cestny_politik as
select exists (
    select *
    from politik p, cestny c
    where p.Meno = c.Meno )

select not (
    select *
    from existuje_cestny_politik ecp
)
```

Príklad 2.6 *Nájdite veci, ktoré sa blyštia, ale pritom nie sú zlaté.*
EDB={blyští(Vec), zlaté(Vec)}.

Relačný kalkul:
{Vec: blysti(Vec) $\wedge$ $\neg$ zlate(Vec)}

Alebo:
blysti(Vec) $\wedge$ $\neg$ zlate(Vec)

Datalog:
result(V) $\leftarrow$ blysti(V), not zlate(V).

SQL:

```
select b.Vec
from blysti b
where not exists (
    select *
    from zlate z
    where b.Vec = z.Vec
)
```

Príklad 2.7 *Nie je všetko zlato, čo sa blyští.*
EDB={blyští(Vec), zlaté(Vec)}.

Relačný kalkul (existuje vec, ktorá sa blyští a nie je zlatá):
$\exists$ Vec (blysti(Vec) $\wedge$ $\neg$ zlate(Vec))

Datalog:
result $\leftarrow$ blysti(V), not zlate(V).

SQL (trochu pritiahnuté za vlasy, keďže výsledkom je jednoriadková tabuľka, ktorá obsahuje TRUE, resp. FALSE):

```
select exists (
    select *
    from blysti b
    where not exists (
        select *
        from zlate z
        where b.Vec = z.Vec
    )
)
```

3 Rozcvičky 30.9.2009–1.10.2009

Príklad 3.1 *Vypočítajte 1+1.*
EDB={add(X, Y, Result)}.

Relačný kalkul:
{R: add(1, 1, R)}

Alebo:
add(1, 1, R)

Datalog:
result(R) ← add(1, 1, R).

SQL:
```
select a.R
from add a
where a.X = 1 and a.Y = 1
```

Príklad 3.2 *Nájdite množinu ľudí, o ktorých platí—o každom z nich osobitne—že nevstúpili dvakrát do rovnakej rieky.*
EDB={clovek(Meno), vstupil(Id_vstupu, Meno, Rieka)}.

Inak povedané: chceme vybrať ľudí, pre ktorých neplatí, že pri aspoň dvoch rôznych vstupoch vstúpili do tej istej rieky.

Relačný kalkul:
{M: clovek(M) ∧ ¬ (∃ R ∃ Id1 ∃ Id2 (Id1 ≠ Id2 ∧ vstupil(Id1, M, R) ∧ vstupil(Id2, M, R)))}

Alebo:
clovek(M) ∧ ¬ (∃ R ∃ Id1 ∃ Id2 (Id1 ≠ Id2 ∧ vstupil(Id1, M, R) ∧ vstupil(Id2, M, R)))

Datalog:
result(M) ← clovek(M), not vstupil_dvakrat(M).
vstupil_dvakrat(M) ← clovek(M), vstupil(Id1, M, R), vstupil(Id2, M, R), not Id1 = Id2.

SQL:
```
create temporary table vstupil_dvakrat as
select c.Meno
from clovek c, vstupil v1, vstupil v2
```

where c.Meno = v1.Meno and c.Meno = v2.Meno and v1.Rieka = v2.Rieka and v1.Id_vstupu <> v2.Id_vstupu

```
select c.Meno
from clovek c
where not exists (
      select *
      from vstupil_dvakrat v2
      where c.Meno = v2.Meno
)
```

Príklad 3.3 *Nájdite množinu ľudí, o ktorých platí—o každom z nich osobitne—že majú niekoho radi, alebo ich má niekto rád.*
EDB={ludia(Meno), rad(Kto, Koho)}.

Relačný kalkul:
{X: clovek(X) ∧ ∃ Y (rad(X, Y) ∨ rad(Y, X))}

Alebo:
clovek(X) ∧ ∃ Y (rad(X, Y) ∨ rad(Y, X))

Datalog:
result(X) ← clovek(M), rad(X, _).
result(X) ← clovek(M), rad(_, X).

SQL:

```
(select c.Meno
from clovek c, rad r
where c.Meno = rad.Kto
)
union
(select c.Meno
from clovek c, rad r
where c.Meno = rad.Koho
)
```

4 Rozcvičky 7.10.2009–8.10.2009 a 14.10–15.10.2009

Príklad 4.1 *Nájdite pijanov, ktorí ľúbia len rum.*
EDB={lubi(Pijan, Alkohol)}.

Relačný kalkul:
{P: lubi(P, rum) ∧ ∀ A (lubi(P, A) ⇒ A=rum)}

Alebo:
lubi(P, rum) ∧ ∀ A (lubi(P, A) ⇒ A=rum)

Datalog:
Ekvivalentná špecifikácia: Pre ktorého pijana neplatí, že ten pijan ľúbi nejaký alkohol rôzny od rumu?
lubi_nie_rum(P) ← lubi(P, A), not A = rum.
result(P) ← lubi(P, rum), not lubi_nie_rum(P).

SQL:

```
create temporary table lubi_nie_rum as
select l.Pijan
from lubi l
where l.Alkohol <> 'rum'

select l.pijan
from lubi l
where l.Alkohol = 'rum' and not exists (
    select *
    from lubi_nie_rum lnr
    where lnr.Pijan = l.Pijan
)
```

Príklad 4.2 *Nájdite pijanov, ktorí vypili aspoň dva rôzne alkoholy.*
EDB={lubi(Pijan, Alkohol), capuje(Krcma, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo)}.

Datalog:

answer(P) ← navstivil(I1, P, _), vypil(I1, A1, _), navstivil(I2, P, _), vypil(I2, A2, _), not A1=A2.

SQL:

```
select n1.Pijan
from navstivil n1, vypil v1, navstivil n2, vypil v2
where n1.Idn = v1.Idn and n2.Idn = v2.Idn and v1.Alkohol <> v2.Alkohol
```

Príklad 4.3 *Nájdite pijanov štamgastov, ktorí doteraz navštívili práve 1 krčmu.*
EDB={lubi(Pijan, Alkohol), capuje(Krcma, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo)}.

Datalog:

answer(P) ← navstivil(_, P, _), not navstivil_rozne(P).

navstivil_rozne(P) ← navstivil(_, P, K1), navstivil(_, P, K2), not K1=K2.

SQL:

```
create temporary table navstivil_rozne as
select n1.Pijan
from navstivil n1, navstivil n2
where n1.Pijan = n2.Pijan and n1.Krcma <> n2.Krcma
```

Príklad 4.4 *Nájdite trojice [Alkohol, Pijan, Krcma] také, že ten Pijan niekedy v tej Krčme vypil naraz viacej než 0.1 l toho alkoholu.*
EDB={lubi(Pijan, Alkohol), capuje(Krcma, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo)}.

Datalog:

answer(A, P, K) ← navstivil(I, P, K), vypil(I, A, M), M >= 0.1.

SQL:

```
select v.Alkohol, n.Pijan, n.Krcma
from navstivil n, vypil v
where n.Idn = v.Idn and Mnozstvo >= 0.1
```

Príklad 4.5 *Nájdite pijanov, ktorí nikdy nedolali rumu (t.j. vypili pri každej návšteve krčmy, ktorá čapuje rum).*
EDB={lubi(Pijan, Alkohol), capuje(Krcma, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo)}.

Datalog:

```
pijani(P) ← lubi(P, _).
pijani(P) ← navstivil(_, P, _).

answer(P) ← pijani(P), not odolal_rumu(P).

odolal_rumu(P) ← navstivil(I, P, K), capuje(K, rum), not vypil(I, rum, _).
```

SQL:

```
create temporary table pijani as
select l.Pijan
from lubi l
union select n.Pijan
from navstivil n

create temporary table odolal_rumu as
select n.Pijan
from navstivil n, capuje c
where n.Krcma = c.Krcma and c.Alkohol = 'rum' and not exists (
    select *
    from vypil v
    where v.Idn = n.Idn and v.Alkohol = 'rum'
)

select p.Pijan
from pijani p
where not exists (
    select *
    from odolal_rumu or
    where p.Pijan = or.Pijan
)
```

Príklad 4.6 *Nájdite ľudí, ktorí nikoho nemajú radi.*
EDB={clovek(Meno), rad(Kto, Koho)}.

Relačný kalkul:

clovek(X) ∧¬ (∃ Y (rad(X, Y))

Datalog:

```
rad_niekoho(X) ← clovek(X), rad(X, _).

result(X) ← clovek(X), not rad_niekoho(X).
```

SQL:

```
select c.Meno
from clovek c
where not exists (
    select *
    from rad r
```

```
    where r.Kto = c.Meno
)
```

5 Rozcvičky 21.10.2009–22.10.2009

Príklad 5.1 *Nájdite krčmy, kde sa pilo len pivo (t.j. okrem piva nič iné).*
EDB={lubi(Pijan, Alkohol), capuje(Krcma, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo)}.

Datalog:

```
result(K) ← navstivil(I, _, K), vypil(I, pivo, _), not pilo_sa_ine(K).

pilo_sa_ine(K) ← navstivil(I, _, K), vypil(I, A, _), not A=pivo.
```

SQL:

```
create temporary table pilo_sa_ine as
select n.Krcma
from navstivil n, vypil v
where n.Idn = v.Idn and v.Alkohol <> 'pivo'

select n.krcma
from navstivil n, vypil v
where n.Idn = v.Idn and v.Alkohol='pivo' and not exists (
    select *
    from pilo_sa_ine psi
    where psi.Krcma = n.Krcma
)
```

Príklad 5.2 *Nájdite pijanov, ktorí vypili maximálne množstvo vodky pri jednej návšteve krčmy. (Vypili na posedenie aspoň toľko vodky ako ktorýkoľvek iný pijan.)*
EDB={lubi(Pijan, Alkohol), capuje(Krcma, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo)}.

Datalog:

```
pv(P, M) ← navstivil(I, P, _), vypil(I, vodka, M).

vypilo_sa_viac(M) ← pv(_, M), pv(_, M2), M2 > M.

result(P) ← pv(P, M), not vypilo_sa_viac(M).
```

SQL:

```
create temporary table pv as
select n.Pijan, v.Mnozstvo
from navstivil n, vypil v
where n.Idn = V.idn and V.alkohol = 'vodka'

create temporary table vypilo_sa_viac as
select pv1.Mnozstvo
from pv pv1, pv pv2
where pv2.Mnozstvo > pv1.Mnozstvo

select p.Pijan
from pv p
```

```
where not exists (
    select *
    from vypilo_sa_viac vsv
    where vsv.Mnozstvo = p.Mnozstvo
)
```

Príklad 5.3 *Nájdite exkluzívne akcie, ktoré sa ponúkajú len na jednom mieste a niekto ich ľúbi.*
EDB={ponuka(Miesto, Akcia), lubi(Meno, Akcia)}.

Relačný kalkul:
ponuka(M, A) ∧ (∃ X lubi(X, A)) ∧ ¬ (∃ M2 (ponuka(M2, A) ∧ M2≠M))

Datalog:
non_exlusive(A) ← ponuka(M1, A), ponuka(M2, A), not M1=M2.

result(A) ← ponuka(_, A), lubi(_, A), not non_exclusive(A).

SQL (pre zmenu s "not in" namiesto "not exists"):

```
create temporary table non_exclusive as
select p.Akcia
from ponuka p1, ponuka p2
where p1.Akcia=p2.Akcia and p2.Miesto<>p.Miesto

select p.Akcia
from ponuka p, lubi l
where l.Akcia=p.Akcia and p.Akcia not in (
    select ne.Akcia from non_exclusive ne)
```

6 Ďalšie rozcvičky

Príklad 6.1 *Nájdite pijanov, ktorí ľubia práve jeden alkohol*
EDB={lubi(Pijan, Alkohol), capuje(Krcma, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo)}.

Relačný kalkul:
∃ A (lubi(P, A) ∧ ¬ (∃ A2 (A ≠ A2 ∧ lubi(P, A2))))

Datalog:
answer(P) ← lubi(P, A), not lubi_iny(P, A).
lubi_iny(P, A) ← lubi(P, A), lubi(P, A2), not A=A2.

SQL:

```
create temporary table lubi iny as
select l1.Pijan, l1.Alkohol
from lubi l1, lubi l2
where l1.Pijan = l2.Pijan and l1.Alkohol <> l2.Alkohol

select l.Pijan
from lubi l
where not exists (
    select *
    from lubi_iny li
    where l.Alkohol = li.Alkohol)
```

Príklad 6.2 *Nájdite dvojice [Alkohol, Suma], ktoré hovoria, aké množstvo ktorého alkoholu sa vypilo celkovo v krčme Carlton (netreba nájsť alkoholy, ktoré sa v Carlton nikdy nepili).*
EDB={lubi(Pijan, Alkohol), capuje(Krcma, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo)}.

Relačný kalkul:
[A, S]: ♡ I, S=sum(M) (∃ P (navstivil(I, P, carlton) ∧ vypil(I, A, M)))

Datalog:
vypil_carlton(I, A, M) ← navstivil(I, _, carlton), vypil(I, A, M).
answer(A, S) ← vypil_carlton(_, A, _), subtotal(vypil_carlton(_, A, M), [A], [sum(M, S)]).

SQL:
```
select v.Alkohol, sum(v.Mnozstvo) as Suma
from navstivil n, vypil v
where n.Krcma = 'carlton' and n.Idn = v.Idn
group by v.Alkohol
```

Relačná algebra:
$\Gamma_{\text{Alkohol, Suma=sum(Mnozstvo)}}\ (\sigma_{\text{Krcma='carlton'}}(\text{navstivil}) \bowtie \text{vypil})$

Príklad 6.3 *Nájdite pijanov, ktorí ľúbia aspoň 5 (rôznych) alkoholov.*
EDB={lubi(Pijan, Alkohol), capuje(Krcma, Alkohol), navstivil(Idn, Pijan, Krcma), vypil(Idn, Alkohol, Mnozstvo)}.

Relačný kalkul:
∃ X (X >=5 ∧ ♡ P, X=count(A) lubi(P, A))

Datalog:
answer(P) ← lubi(P, _), subtotal(lubi(P, A), [P], [count(A, X)]), X >= 5.

SQL:
```
select l.Pijan
from lubi l
group by l.Pijan
having count(l.Alkohol) >= 5
```

Relačná algebra:
$\Pi_{\text{Pijan}}(\sigma_{\text{X>=5}}(\Gamma_{\text{Pijan, X=count(Alkohol)}}\ (\text{lubi})))$