Výpočet rekurzívnych dotazov v relačných databázových systémoch

Jakub Šmahovský
doc. Mgr. Tomáš Plachetka, Dr.

24. júna 2019

Fakulta matematiky, fyziky a informatiky

Predmet jeto práce

- databázové systémy sa skladajú z mnohých odvetví
- zameriavame sa špeciálne na výpočet dotazu
  - máme k dispozícii dotaz a relačné dáta
  - cieľom je vypočítať výsledok dotazu daný sémantikou

Databázové jazyky

Deklaratívne jazyky

- program definuje iba výsledok
- kompilátor zvykne vykonávať optimalizácie
- SQL, Datalog

Procedurálne jazyky

- program definuje postup výpočtu
- interpréter nezvykne vykonávať optimalizácie
- relačná algebra

Relačná algebra

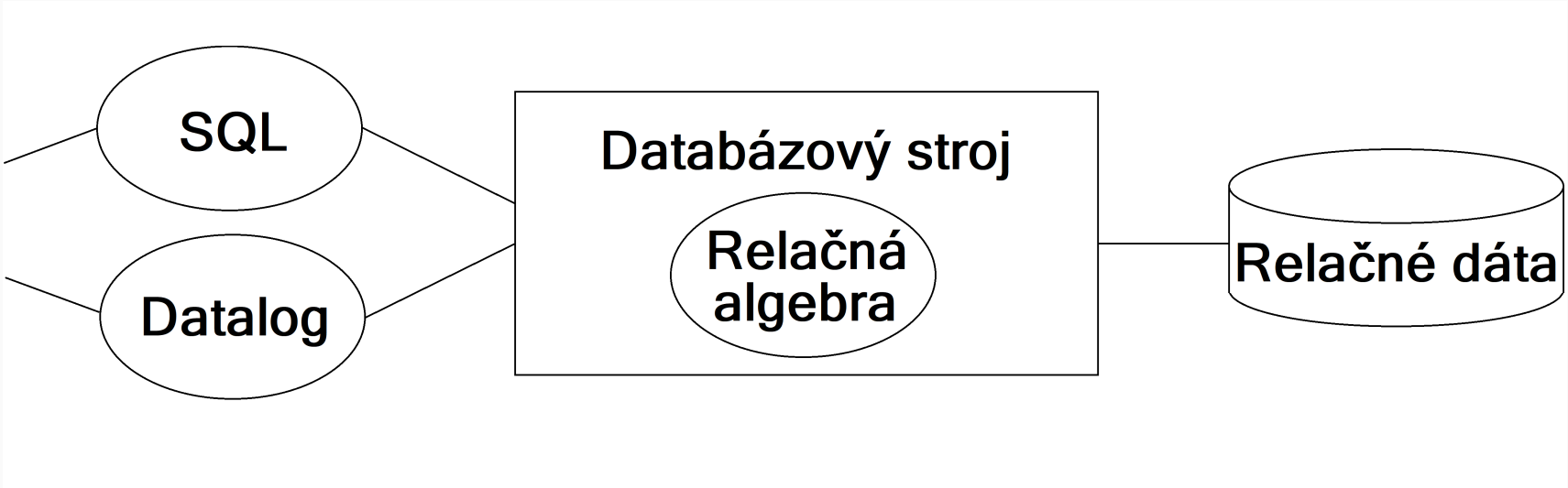
selekcia	σ_C
projekcia	π_C
zjednotenie	$\cup$
join	$\bowtie_C$
antijoin	$\triangleright_C$

filtrovacie podmienky C

priradenie	$:=$
cyklus	while begin *"telo cyklu"* end

Výpočet dotazov

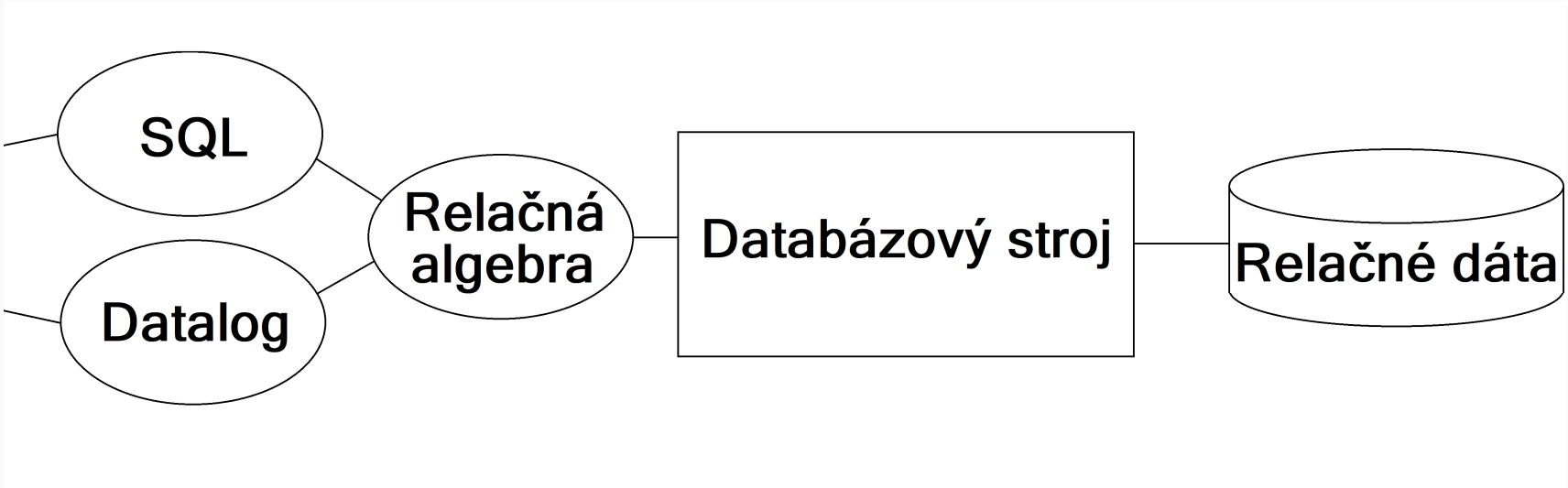
Systémy na výpočet dotazov môžu mať rôznu architektúru



- dotazy v deklaratívnom jazyku môžu byť priamo interpretované databázovým strojom
- databázový stroj iba interne implemntuje obdobu relačnej algerby

Výpočet dotazov

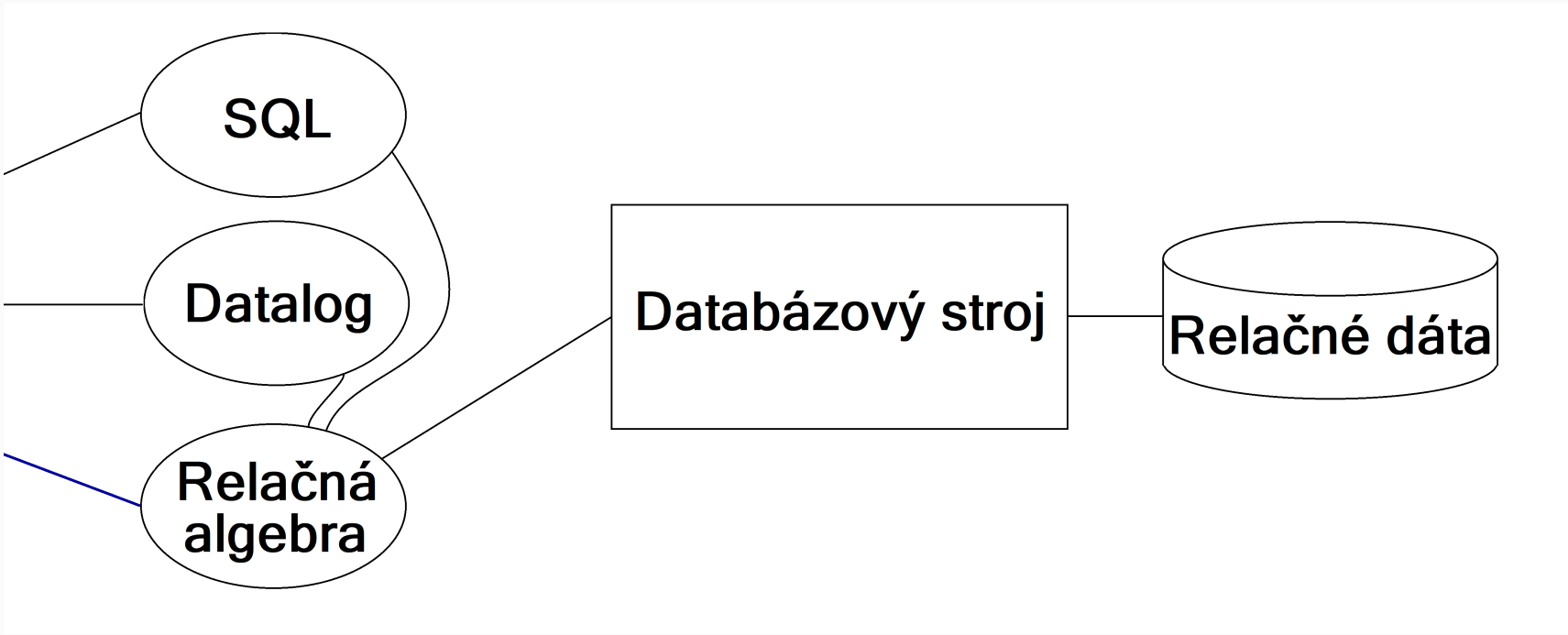
Systémy na výpočet dotazov môžu mať rôznu architektúru



- dotazy môžu byť prekladané do relačnej algebry v samostatnom module systému
- databázový stroj interpretuje iba dotazy v relačnej algebre

Výpočet dotazov

Systémy na výpočet dotazov môžu mať rôznu architektúru

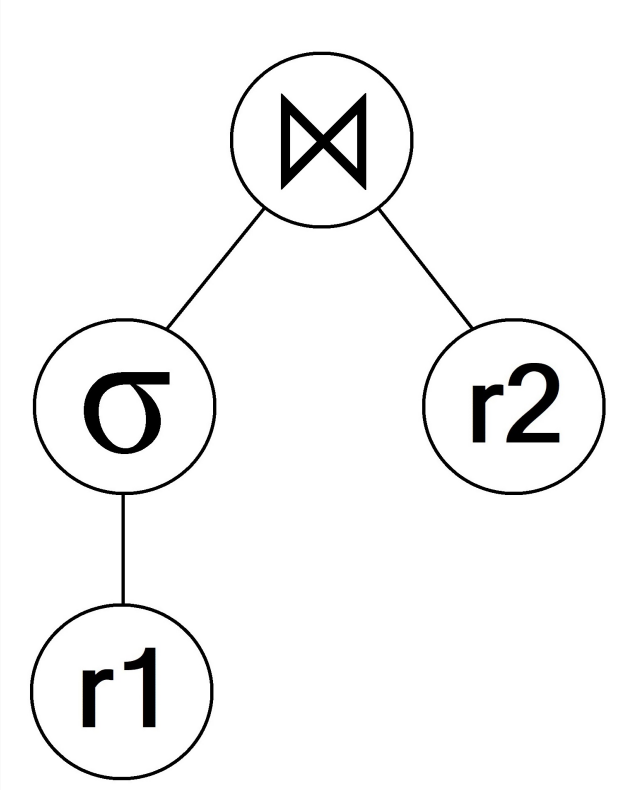


- chceli sme používateľovi umožniť písať dotazy priamo v jazyku relačnej algebry
- zároveň sme chceli zachovať aj možnosť použiť deklaratívny jazyk

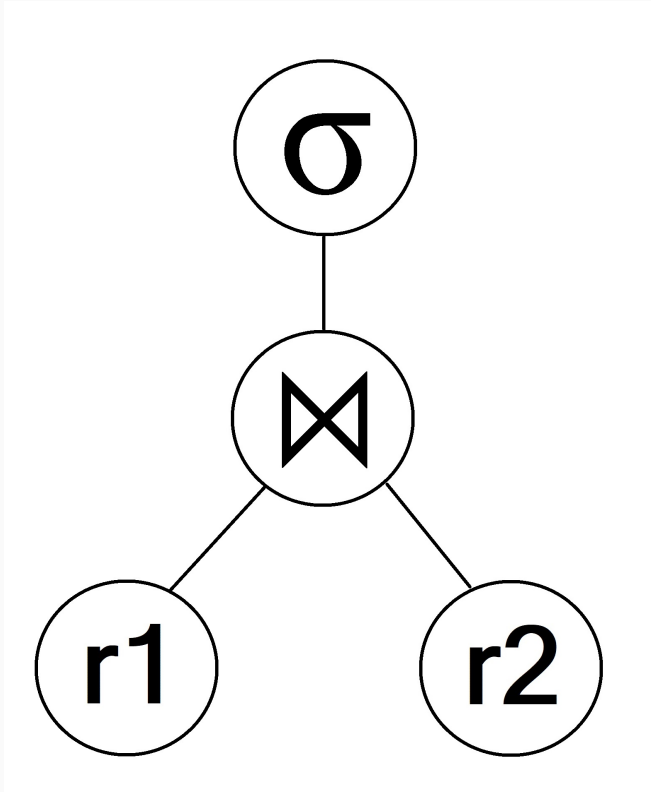
Optimalizácie

Niektoré optimalizácie vieme robiť všeobecne

lepšie



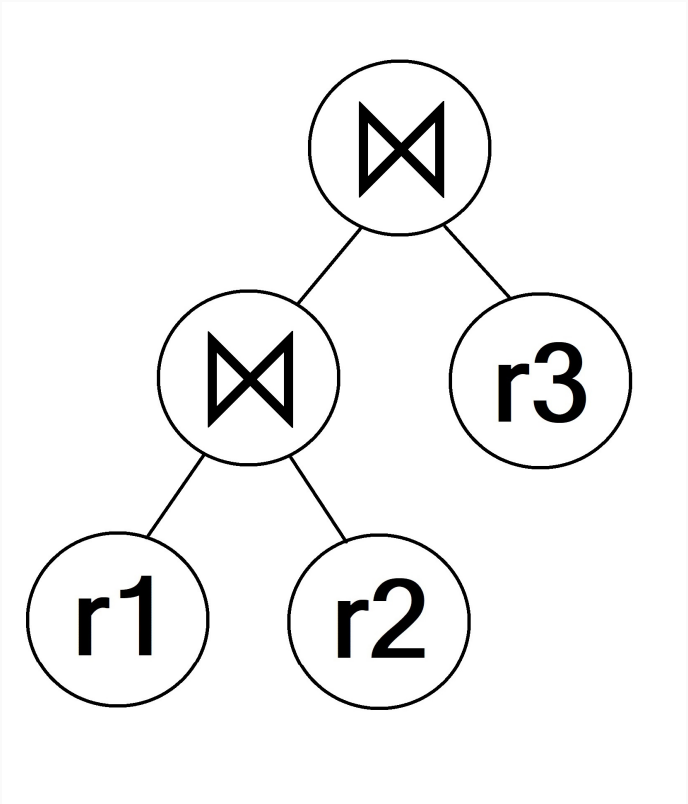
horšie



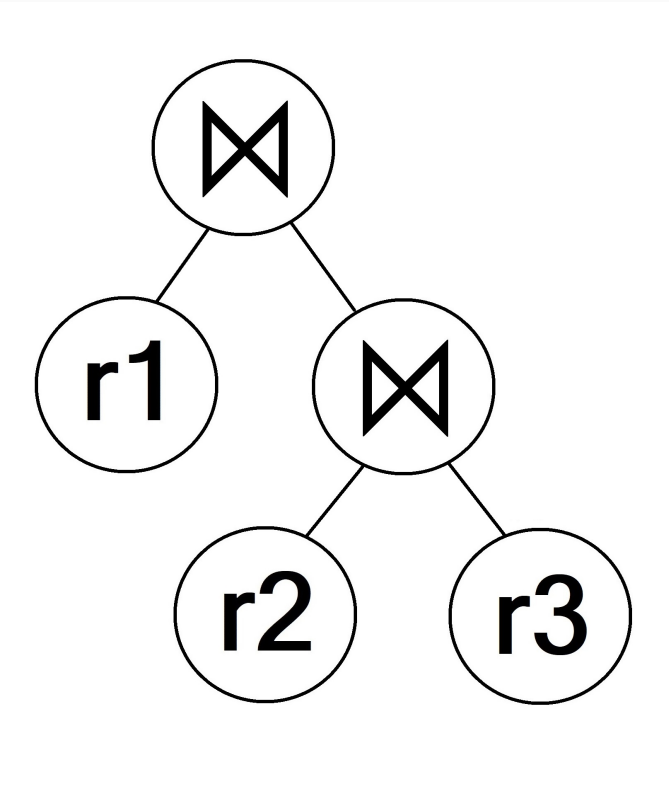
Optimalizácie

Niektoré optimalizácie sa všeobecne robia ťažko

???



???



Výsledok práce

- súhrn znalostí pre preklad a počítanie rekurzívnych dotazov v Datalogu a relačnej algebre
- návrh a implementácia minimalistického databázového systému, ktorý počíta dotazy v oboch týchto jazykoch
  - operácie relačnej algebry
  - kompilátor datalogu
  - interpréter relačnej algebry
- testy demonštrujúce vybrané možnosti optimalizácie
  - poradie operácií join
  - poradie vstupov pre join
  - vynútenie materializácie

Materializácia versus iterácia

- zasebou nadväzujúce operácie
- operácie si posúvajú záznamy relácií po jednom
- materializácia medzivýsledku môže zaberať priveľa pamäte
- ukladanie a načítavanie môže zaberať viac času ako opätovné počítanie

Implementácia iteratívnych operácií relačnej algebry

- rekurzívne volať next na odstraňovanie duplikátov je nerealistické

```
interface Operator {
    public Tuple next();
    public Operator instance();
    public void reset();
}
```

- riešením je interne používať novú metódu

```
Tuple nonDistinctNext();
```

- niekedy je materializácia žiadaná, napríklad na rekurziu
- priradenie do relácie = materializácia

Výpočet Datalogu podľa well-founded sémantiky

- programom udávame význam podľa takzvanej well-founded sémantiky
- dobre odzrkadľuje to, čo mal pravdepodobne programátor na mysli
- dovoľuje nám počítať programy s akoukoľvek rekurziou
- náročnejší výpočet
- relačnú algebru definujeme tak, aby umožňovala vyjadriť programy s rovnakým významom

Výpočet Datalogu podľa well-founded sémantiky

- ako počítame takýto program (s reláciou $e = \{(1)\}$)?

  $r(X) \leftarrow \neg s(X), e(X).$

  $s(X) \leftarrow \neg r(X), e(X).$
- v dvojhodnotovej logike nedostaneme uspokojivý výsledok
  - intuícia: $r(1) \vee s(1)$
  - označiť obe za nepravdivé je sporné
  - žiadnu z iných možností nevieme dokázať
- uvažujeme v trojhodnotovej logike
  - oba výrazy označíme za neznáme (unknown)
  - na otzázky $r(1)$? $s(1)$? $\neg r(1)$? $\neg s(1)$? odpovedáme false
  - odpovede sú iba také, ktoré vieme odvodiť zo zdrojových dát

Výpočet Datalogu podľa well-founded sémantiky

- každé pravidlo preložíme na priradenie

  $r := e \triangleright_{\hat{1}=\hat{2}} s$

  $s := e \triangleright_{\hat{1}=\hat{2}} r$
- vnútorný cyklus: pravá strana antijoinu z predošlého pevného bodu, hľadáme nový pevný bod
- vonkajší cyklus: hľadáme alternujúci pevný bod

pevný bod	1.	2.	3.	4.	5.
r(1)	false	true	false	true	...
s(1)	false	true	false	true	...

po tretej iterácii vieme, že sme našli výsledok

Zhrnutie

- venovali sme sa Datalogu a relačnej algebre
- zhrnuli sme v práci potrebné znalosti
- implementovali sme operácie relačnej algebry, základný kompilátor datalogu a interpréter relačne algebry
- vykonali sme a zdokumentovali demonštračné testy, ktoré ukazujú ako sa dá takýto systém využiť