

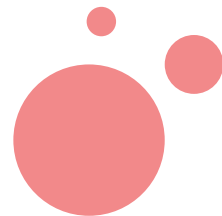


# Projektový seminár 1

FMFI, UK, Pavel Rajčok

**Školiteľ**

Ing. Alexander Šimko, PhD.



## **Anotácia diplomovej práce**

Aby databázový systém fungoval efektívne, je ho potrebné nakonfigurovať. Súčasťou tejto konfigurácie je aj vytváranie vhodných indexov, ktoré typicky robí databázový špecialista. Vzhľadom aj na finančné náklady je zaujímavé túto činnosť plne automatizovať. Problém automatického výberu indexov v databázových systémoch je NP-ťažký a jeho riešeniu bolo venované značne množstvo výskumu. Pre komerčné databázové systémy existujú nástroje, ktoré tento problém riešia, pre open-source databázový systém PostgreSQL takéto nástroje chýbajú, s výnimkou veľmi mladého projektu Dexter. Cieľom práce je nadviazať na výskum v tejto oblasti, navrhnúť a implementovať open-source nástroj pre automatický výber a manažovanie indexov v open-source databázovom systéme PostgreSQL. Súčasťou práce bude aj experimentálne vyhodnotenie relevantných parametrov vytvoreného nástroja.

**Automatický  
manažment  
indexov  
pre PostgreSQL**



# Agenda

**01**

Zdroje

**02**

Problematika

**03**

Existujúci softvér

**04**

Implementácia





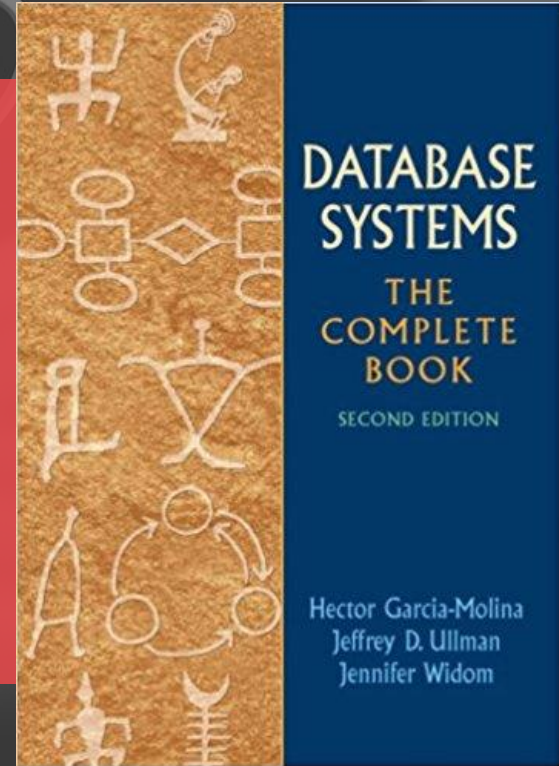
Zdroje

1

A close-up, shallow depth-of-field photograph of a person's hands typing on a laptop keyboard. The hands have light-colored skin and are wearing dark red nail polish. The fingers are positioned over the keys, with the right hand more prominent in the foreground. A semi-transparent white circle is centered over the text. To the right of the circle, there are three red circles of varying sizes, arranged in a cluster. The background is blurred, showing more of the laptop and the person's arms.

Knižné zdroje

# Database Systems The Complete Book



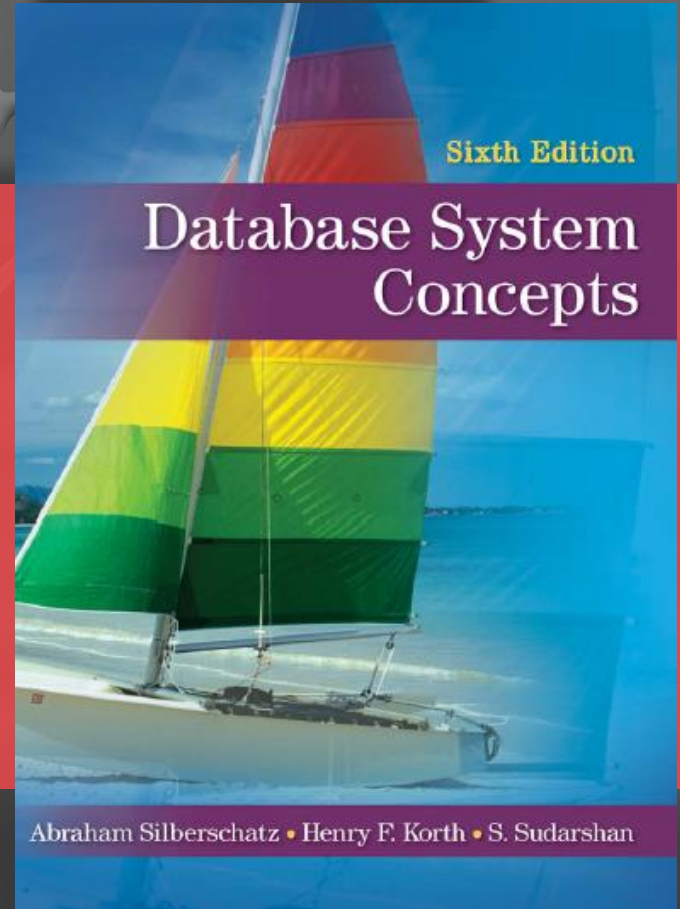
H. Garcia-Molina, J. D. Ullman, J. Widom; ISBN-13: 978-0131873254



# Database System Concepts

Kapitola:  
„PostgreSQL“

A. Silberschatz, H. F. Korth, S. Sudarshan; ISBN-13: 978-0073523323



# Advances in Data Management

Kapitola:

„Automatic Index Selection in RDBMS by  
Exploring Query Execution Plan Space“

Z. W Ras, A. Dardzinska; ISBN 978-3-642-02190-9

Studies in Computational Intelligence 223

Zbigniew W. Ras  
Agnieszka Dardzinska (Eds.)

## Advances in Data Management

 Springer



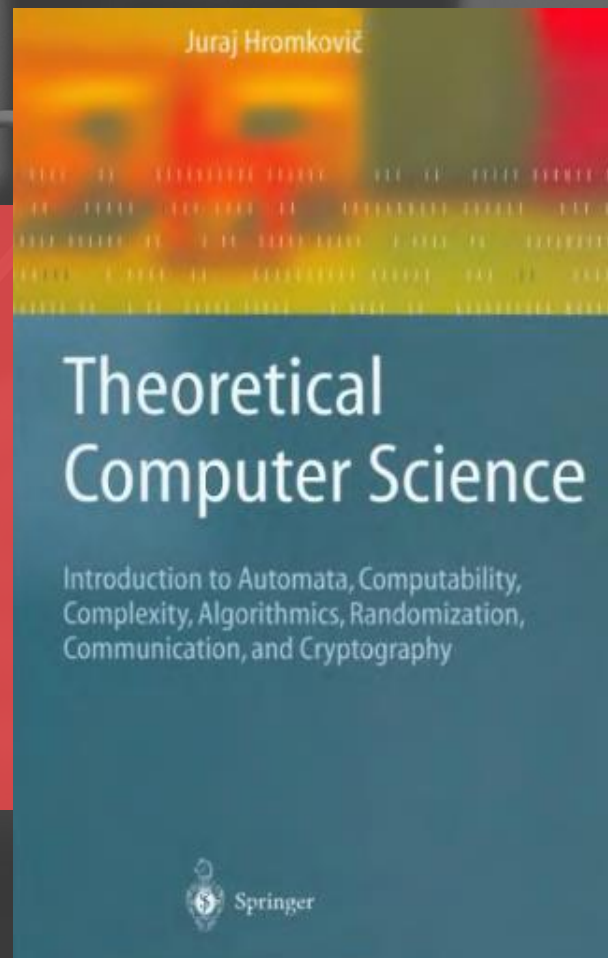
# Theoretical Computer Science

Kapitoly:

„Complexity Theory“ a

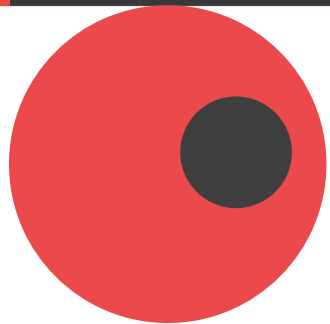
„Algorithmics for Hard Problems“

J. Hromkovic; ISBN 978-3-540-14015-3



A close-up, shallow depth-of-field photograph of a person's hands typing on a laptop keyboard. The hands have light-colored skin and are wearing dark red nail polish. The fingers are positioned over the keys, with the right hand more prominent in the foreground. A semi-transparent white circle is centered over the text, and three red circles of varying sizes are positioned to the upper right of the main circle. The background is blurred, showing more of the keyboard and the person's arms.

Odborné články



## **A Genetic Algorithm for the Index Selection Problem**

J. Kratica, Ivana Ljubič, D. Tošič;  
Print ISBN: 0-7695-2657-8



## **Dexter**

Kane A.;  
<https://github.com/ankane/dexter>



## **Implementation of an Agent Architecture for Automated Index Tuning**

R.L. de Carvalho Costa, S. Lifschitz,  
M.F. de Noronha;  
Print ISBN: 0-7695-2657-8



## **Autonomic Index Management**

S. Lifschitz; M. A. Vaz Salles;  
Print ISBN: 0-7965-2276-9

# Problematika

2



# NP problém



- NP-úplný problém je taký problém, ktorý patrí do triedy NP (je vypočítateľný v nedeterministickom polynomiálnom čase)

- NP-úplné problémy v istom zmysle reprezentujú tie najťažšie problémy spomedzi triedy NP

- ľubovoľný iný problém z triedy NP je naň polynomiálne redukovateľný (tzn. je NP-ťažký).



**NP ťažký problém**


$$NP = \bigcup_{k \in \mathbb{N}} NTIME(n^k)$$

$NTIME(n^k)$  je množina možností problému, ktoré môžu byť vyriešené nedeterministickým Turingovým strojom so zložitou  $O(n^k)$ .

NP problém

# The Index Selection Problem



- kľúčový problém v návrhu databáz

- problém známy ako NP ťažký

- Pre efektívne fungovanie databázového systému, je potrebné ho nakonfigurovať

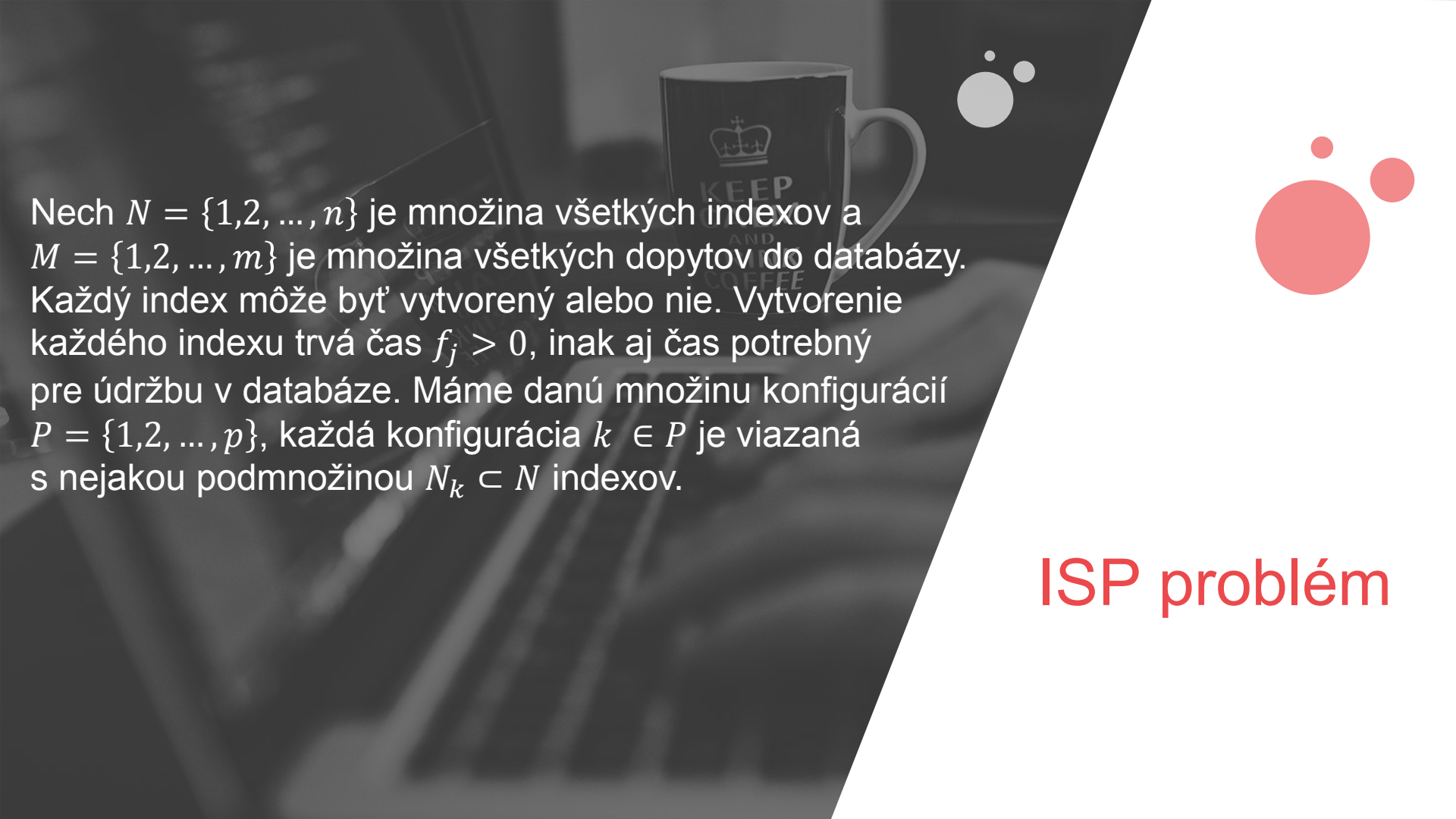
- jeho riešeniu bolo venované značne množstvo výskumu

**ISP problém**

- Minimalizovanie času pri práci s databázou

- pre komerčné databázové systémy existujú nástroje, ktoré tento problém riešia, pre open-source databázový systém PostgreSQL takéto nástroje chýbajú


- Hlavným cieľom je minimalizovať celkový čas vykonávania, definovaný ako súčet časov údržby a odpovedí databázového systému pre všetky dopyty



Nech  $N = \{1, 2, \dots, n\}$  je množina všetkých indexov a  $M = \{1, 2, \dots, m\}$  je množina všetkých dopytov do databázy. Každý index môže byť vytvorený alebo nie. Vytvorenie každého indexu trvá čas  $f_j > 0$ , inak aj čas potrebný pre údržbu v databáze. Máme danú množinu konfigurácií  $P = \{1, 2, \dots, p\}$ , každá konfigurácia  $k \in P$  je viazaná s nejakou podmnožinou  $N_k \subset N$  indexov.

ISP problém





Konfigurácia je aktívna, ak sú všetky jej indexy vytvorené, počas spustenia dopytu  $i \in M$ , tým získavame  $g_{ik} \geq 0$  čas.

V praxi, väčšina párov  $(i, k), i \in M, k \in P$  má  $g_{ik}$  rovný nule. Toto môže byť jednoducho vysvetlené faktom, že konkrétna konfigurácia má vplyv na obmedzené množstvo dopytov z množiny  $M$ . Naším cieľom bude vytvoriť také indexy, ktorých čas potrebný na spustenie všetkých dopytov bude minimalizovaný. T.j. celkový čas  $g$  bude maximalizovaný.

ISP problém

# Hypotetické indexy



- Hypotetické indexy ukazujú, ako by sa zmenili podmienky pri spustení dopytu, ak by konkrétny index existoval

- Nevyžadujú si žiadny čas pre vytvorenie a žiadne miesto v pamäti, nespomaľujú procesor, existujú len pre konkrétnu reláciu

- Existujú len v schéme databázy, nie sú skutočne vytvorené

- Je skutočne užitočné vedieť, či konkrétne indexy dokážu zvýšiť výkonnosť pri problematických dopytoch



**Hypotetické  
indexy**

# HypoPG

**hypopg():** vráti množinu hypotetických indexov - podobne ako aj pg\_index()

**hypopg\_add\_index(schema, table, attribute, access\_method):**  
vytvorí jedno-stĺpcový hypotetický index

**hypopg\_create\_index(query):** vytvorí hypotetický index  
štandardným CREATE INDEX príkazom v SQL jazyku

**hypopg\_drop\_index(oid):** odstráni konkrétny hypotetický index

**hypopg\_list\_indexes():** vráti stručnú verziu zoznamu  
dostupných hypotetických indexov

**hypopg\_relation\_size(oid):** vráti odhadovanú veľkosť  
hypotetického indexu

**hypopg\_reset():** odstráni všetky hypotetické indexy



Hypotetické  
indexy  
v PostgreSQL

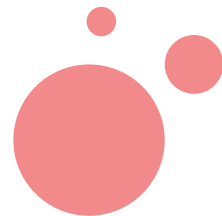
PostgreSQL navrhuje plán dopytu pre každý dopyt, ktorý zaznamená

Výber správneho plánu, ktorý zodpovedá štruktúre dopytu a vlastnostiam dát, je absolútne rozhodujúci pre dobrý výkon.

Systém obsahuje komplexný plánovač, ktorý sa snaží vybrať si vhodné plány.

Pomocou príkazu EXPLAIN môžete zistiť, aký plán dopytu vytvorí plánovač pre akýkoľvek dotaz.

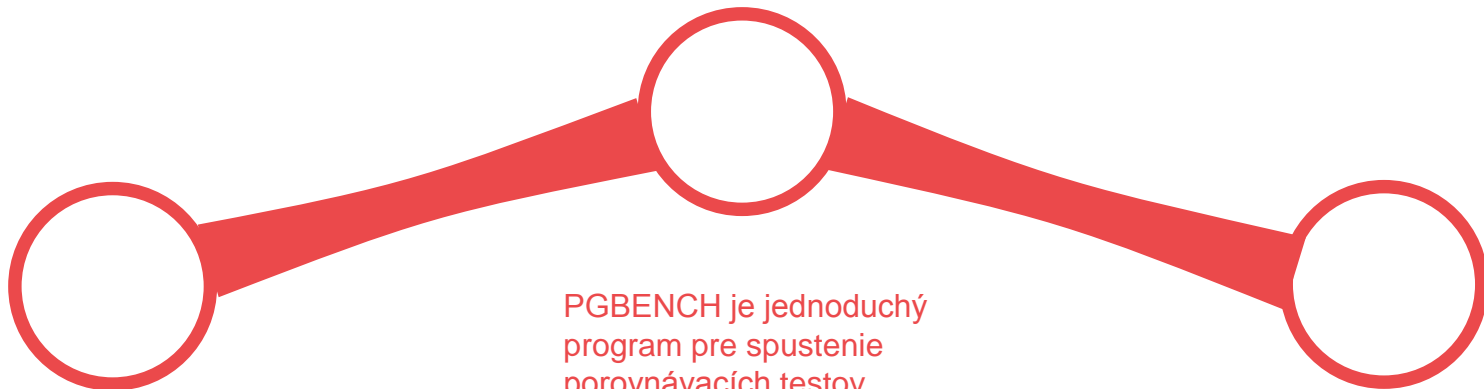
Príkaz EXPLAIN má dve formy výstupu. Predvolený výstup je vo forme textu, ktorému človek ľahšie porozumie. Ak chceme výstup ďalej spracovávať, vhodné je využiť druhú formu výstupu ako sú formáty XML, JSON alebo YAML



# Príkaz EXPLAIN v PostgreSQL



# Benchmarking v PostgreSQL - PGBENCH



Spúšťa rovnakú sekvenciu príkazov SQL znovu a znovu, prípadne v niekoľkých súbežných databázových reláciách a potom vypočíta priemernú transakčnú rýchlosť (transakcie za sekundu).

PGBENCH je jednoduchý program pre spustenie porovnávacích testov na PostgreSQL. Príkaz PGBENCH spustí porovnávacie testy v PostgreSQL.

Štandardne PGBENCH testuje scenár, ktorý je voľne založený na TPC-B, zahŕňajúci päť príkazov typu *SELECT*, *UPDATE* a *INSERT* na jednu transakciu. Je však ľahké otestovať ďalšie prípady tým, že napíšete vlastné súbory skriptu transakcií.



Existujúci softvér

3

# Dexter, the Automatic Indexer for Postgres



Dexter pracuje v dvoch fázach:


- Zhromažďovanie dopytov
- Generovanie indexov



# Dexter, the Automatic Indexer for Postgres



## Fáza zhromažďovania dopytov:

- 
- Priame sledovanie záznamov v nástroji Dexter
  - Analýza zo záznamov - spustené sql príkazy a ich trvanie
  - Zoskupovanie podobných sql príkazov  
– využitý modul PG\_STAT\_STATEMENTS



# Dexter, the Automatic Indexer for Postgres



## Fáza generovania indexov:

Vytváranie hypotetických indexov

– snaha urýchliť pomalé sql dopyty, ktoré sme v prvej fáze zhromaždili

- Filtrovanie dopytov na systémových tabuľkách a ďalších databázach
- Analýza tabuliek pre aktuálne štatistiky plánovača
- Ohodnotiť počiatočné náklady na dopyty
- Použiť hypotetické indexy, ak ešte neexistujú
- Znovu hodnotiť dopyty, ak boli použité hypotetické indexy





Implementácia

4

# Databáza Sakila

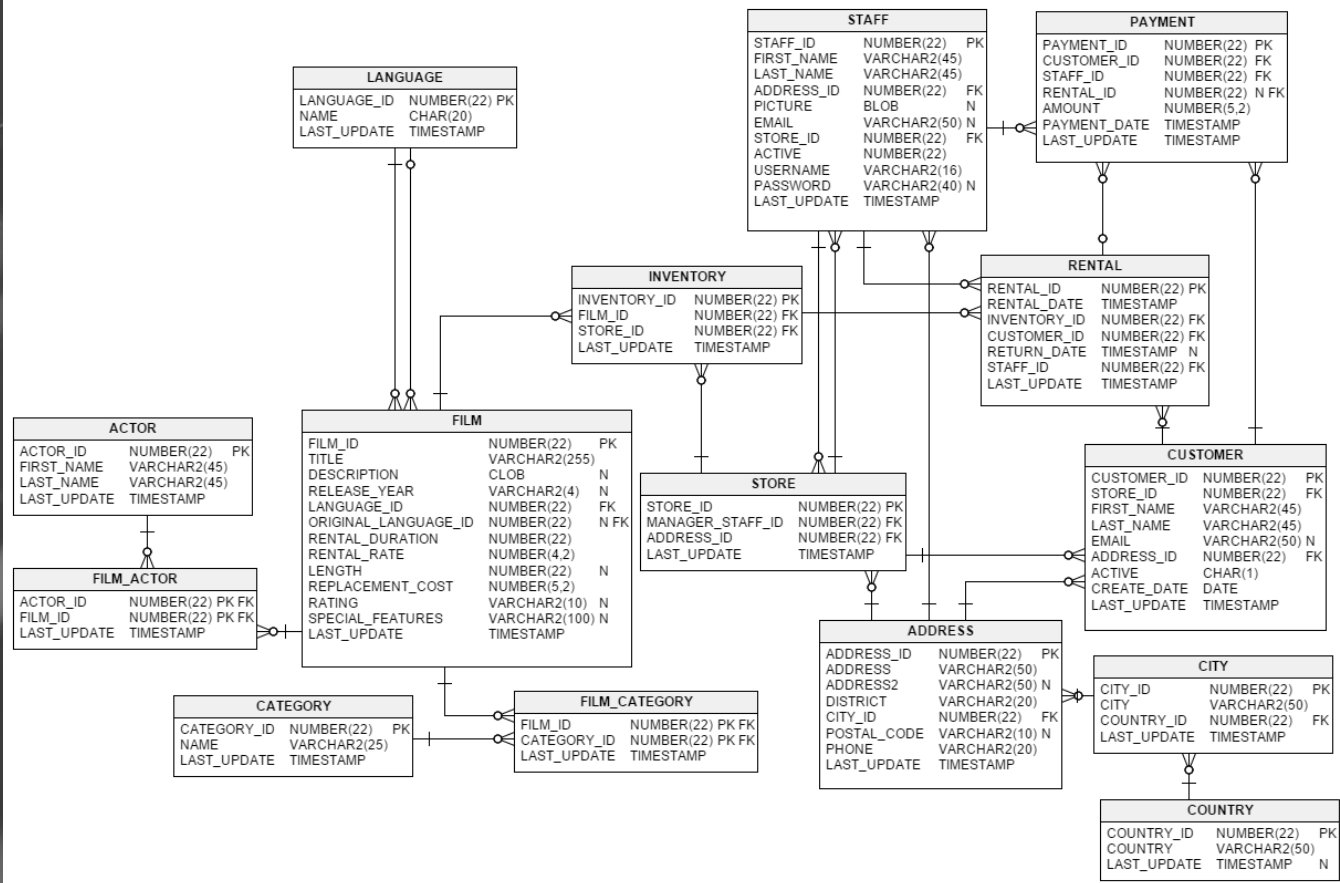


Implementovaná jedna z najlepších príkladov databáz. Ktorá bola originálne vytvorená pre MySQL databázový systém.



- 16 tabuliek
- 7 pohľadov
- 6 spúšťačov
- Procedúry a funkcie
- Viac než 46000 záznamov
- Pripravený návrh na rozšírenú verziu databázy, z dôvodu testov nad väčším množstvom údajov

# Schéma databázy Sakila



```
C:\Program Files\PostgreSQL\9.6\bin>pgbench -U postgres dvd_store
Password:
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 1
query mode: simple
number of clients: 1
number of threads: 1
number of transactions per client: 10
number of transactions actually processed: 10/10
latency average = 12.407 ms
tps = 80.599869 (including connections establishing)
tps = 156.242082 (excluding connections establishing)
```



```
C:\Program Files\PostgreSQL\9.6\bin>pgbench -t 100 -c 32 -U postgres dvd_store
Password:
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 1
query mode: simple
number of clients: 32
number of threads: 1
number of transactions per client: 100
number of transactions actually processed: 3200/3200
latency average = 48.899 ms
tps = 654.406180 (including connections establishing)
tps = 659.979702 (excluding connections establishing)
```

```
C:\Program Files\PostgreSQL\9.6\bin>pgbench -U postgres -f "C:\custom_script.sql"
-c 10 dvd_store
Password:
starting vacuum...end.
transaction type: C:\custom_script.sql
scaling factor: 1
query mode: simple
number of clients: 10
number of threads: 1
number of transactions per client: 10
number of transactions actually processed: 100/100
latency average = 918.425 ms
tps = 10.888206 (including connections establishing)
tps = 10.936032 (excluding connections establishing)
```

# pgbench

Malá část výstupu  
testu pre  
„debug“ funkcionalitu

```
client 9 receiving
client 9 sending INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUE
$ (8, 1, 121, -1031, CURRENT_TIMESTAMP);
client 9 receiving
client 9 sending END;
client 9 receiving
client 1 receiving
client 1 sending INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUE
$ (3, 1, 1231, -2235, CURRENT_TIMESTAMP);
client 8 receiving
client 8 sending UPDATE pgbench_branches SET bbalance = bbalance + -2328 WHERE b
id = 1;
client 1 receiving
client 1 sending END;
client 1 receiving
client 8 receiving
client 8 sending INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUE
$ (8, 1, 98214, -2328, CURRENT_TIMESTAMP);
client 8 receiving
client 8 sending END;
client 8 receiving
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 1
query mode: simple
number of clients: 10
number of threads: 1
number of transactions per client: 10
number of transactions actually processed: 100/100
latency average = 461.125 ms
tps = 21.686090 (including connections establishing)
tps = 21.922897 (excluding connections establishing)
```

# príkaz explain

Príklad použitia  
príkazu EXPLAIN

```
EXPLAIN SELECT * FROM rental limit 50 ;
```

Output pane

Data Output

Explain

Messages

History

QUERY PLAN  
text

1	Limit (cost=0.00..0.97 rows=50 width=36)
2	-> Seq Scan on rental (cost=0.00..310.44 rows=16044 width=36)

# príkaz explain


Príklad použitia  
príkazu EXPLAIN

```
explain
select c.last_name, ci.city
from customer c, address a, city ci
where c.address_id = a.address_id
and ci.city_id = a.city_id
and c.active = 1
limit 50
```

Output pane

	Data Output	Explain	Messages	History
	<b>QUERY PLAN</b>			
	text			
1	Limit (cost=1.06..26.09 rows=50 width=16)			
2	-> Nested Loop (cost=1.06..293.41 rows=584 width=16)			
3	-> Merge Join (cost=0.79..81.89 rows=584 width=9)			
4	Merge Cond: (c.address id = a.address id)			
5	-> Index Scan using idx fk address id on customer c (cost=0.28..38.76 rows=584 width=9)			
6	Filter: (active = 1)			
7	-> Index Scan using address pkey on address a (cost=0.28..34.32 rows=603 width=6)			
8	-> Index Scan using city pkey on city ci (cost=0.28..0.35 rows=1 width=13)			
9	Index Cond: (city id = a.city id)			



A close-up, shallow depth-of-field photograph of a person's hands typing on a laptop keyboard. The hands have light-colored skin and are wearing dark red nail polish. The fingers are positioned over the keys, with the right hand more prominent in the foreground. The background is blurred, showing more of the keyboard and the person's arms. Overlaid on the image is a large, semi-transparent white circle containing the text 'Ďakujem za pozornosť'. To the right of this circle are three red circles of varying sizes, arranged in a cluster.

Ďakujem za  
pozornosť