PA152: Efektivní využívání DB

10. Ladění schéma

Vlastislav Dohnal

Schéma

- Schéma relace
  - Seznam atributů, jejich typů a integritních omezení
  - Např.
    - Relace student(uco, jmeno, prijmeni, datum_narozeni)
- Schéma databáze
  - Schéma všech relací

Rozdíly ve schématech

- Stejná data lze uložit různými způsoby

- Příklad
  - Dodavatelé
    - Adresa
  - Objednávky
    - Výrobek, počet, dodavatel

Rozdíly ve schématech

- Alternativy
  - Schéma 1
    - Objednávka1(dodavatel_id, výrobek_id, počet, dodavatel_adresa)
  - Schéma 2
    - Objednávka2(dodavatel_id, výrobek_id, počet)
    - Dodavatel(id, adresa)
- Rozdíly
  - Schéma 2 šetří místem
  - Schéma 1 nemusí zachovat adresu, pokud není objednávka

Rozdíly ve schématech

- Rozdíly ve výkonu
  - Častý přístup k adrese dodavatele konkrétního výrobku
    - → schéma 1 je vhodnější (není potřeba spojení)
  - Mnoho objednávek
    - → schéma 1 plýtvá místem (opakování adresy)
    - → relace bude mít mnoho bloků

Teorie pro správný návrh schéma

- Normální formy
  - 1NF, 2NF, 3NF, Boyce-Coddova NF, …
- Funkční závislost
  - $A \rightarrow B$
    - *B funkčně závisí na A*
    - Hodnotu atributu *B* zjistíme, pokud známe hodnotu atributu *A*
    - Nechť *t*, *s* jsou lib. řádky relace,
      pak platí: t[A] = s[A] $\Rightarrow$ t[B] = s[B]

Teorie pro správný návrh schéma

- Objednávka1(dodavatel_id, výrobek_id, počet, dodavatel_adresa)
- Příklad funkčních závislostí
  - dodavatel_id $\rightarrow$ dodavatel_adresa
  - dodavatel_id, výrobek_id $\rightarrow$ počet

Teorie pro správný návrh schéma

- K je primární klíč
  - $K \rightarrow R$
  - $L \not\rightarrow R$ pro libovolné $L \subset K$
    - Tj. pro každý atribut A z R platí: $K \rightarrow A$ a $L \not\rightarrow A$

- Příklad
  - Dodavatel(id, adresa)
  - $id \rightarrow$ adresa
  - *id* je primární klíč

Teorie pro správný návrh schéma

- Příklad
  - Objednávka1(dodavatel_id, výrobek_id, počet, dodavatel_adresa)
  - dodavatel_id $\rightarrow$ dodavatel_adresa
  - dodavatel_id, výrobek_id $\rightarrow$ počet
  - *dodavatel_id, výrobek_id* je primární klíč

Normalizace schématu

- 1NF – všechny atributy jsou atomické
- 2NF – všechny atributy závisí na celém klíči
- 3NF – všechny atributy závisí přímo na klíči
  - není tranzitivní závislost

- Normalizace = převod do BCNF (3NF)

Normalizace schématu

- Relace *R* je normalizovaná, pokud
  - Pro každou netriviální funkční závislost $X \rightarrow A$ nad atributy relace *R* platí, že *X* je (super-)klíč.
- Příklad
  - Objednávka1(dodavatel_id, výrobek_id, počet, dodavatel_adresa)
    - dodavatel_id $\rightarrow$ dodavatel_adresa
    - dodavatel_id, výrobek_id $\rightarrow$ počet
  - Není normalizovaná

Normalizace schématu

- Příklad
  - Objednávka2(dodavatel_id, výrobek_id, počet)
    - dodavatel_id, výrobek_id $\rightarrow$ počet
  - Dodavatel(id, adresa)
    - id $\rightarrow$ adresa
  - Schéma je normalizované

Příklad normalizace

- Banka
  - Zákazník má otevřený účet
  - Zákazník má korespondenční adresu
  - Účet je vedený konkrétní pobočkou banky

- Je relace normalizovaná?
  - Banka(zákazník, účet, adresa, pobočka)

Příklad normalizace

- Relace
  - Banka(zákazník, účet, adresa, pobočka)
  - zákazník $\rightarrow$ účet
  - zákazník $\rightarrow$ adresa
  - účet $\rightarrow$ pobočka
- Primární klíč je *zákazník*
  - Důkaz lze provést pomocí funkčních závislostí
- Relace není normalizovaná
  - Máme tranzitivní závislost

Příklad normalizace

- Rozklad relace na
  - Banka(zákazník, účet, adresa)
    - zákazník $\rightarrow$ účet
    - zákazník $\rightarrow$ adresa
  - Účet(účet, pobočka)
    - účet $\rightarrow$ pobočka
  - Nyní již je normalizované

Postup návrhu schématu

- Identifikace entit
  - Zákazník, dodavatel, objednávka, …
- Každá entita má atributy
  - Zákazník má adresu, telefon, …
- Entita musí splňovat:
  1. Atribut nemá další atribut (je atomický).
  2. Musí existovat funkční závislost pro každý atribut, který není součástí klíče.

Postup návrhu schématu

- Každá entita je nová relace
- Vztah mezi entitami je vyjádřen samostatnou relací
  - PracujeNa(zaměstananec_id, projekt_id)

- Nalezení funkčních závislostí mezi všemi atributy a kontrola normalizace schématu
  - Pokud existuje závislost $AB \rightarrow C$, pak ABC musí být ve stejné relaci.

Vertikální dělení

- Příklad: Telefonní operátor
  - Entita zákazník má id, adresu a zbývající kredit
    - Závislosti:
      - id $\rightarrow$ adresa
      - id $\rightarrow$ kredit
  - Normalizované schéma
      - Zákazník(id, adresa, kredit)
    - Nebo
      - ZákazníkAdresa(id, adresa)
      - ZákazníkKredit(id, kredit)
  - Které je vhodnější?

Vertikální dělení

- Volba správného schéma závisí na způsobu používání (dotazování)
  - Výpisy z účtu jsou posílány jednou měsíčně.
  - Kredit je měněn po každém telefonním hovoru.
- → Druhé schéma je vhodnější
  - Relace ZákaznícKredit bude menší
    - Méně bloků, může se vejít do paměti
    - → rychlejší table/index scan

Vertikální dělení

- Jedna relace je vhodnější než dvě
  - Pokud jsou atributy přistupovány současně
  - $\rightarrow$ není potřeba operace spojení
- Dvě relace jsou vhodnější
  - Atributy jsou přistupovány samostatně (nebo některé řádově častěji)
  - Atributy jsou velké (dlouhé řetězce, …)
    - Pozor: LOB jsou uloženy mimo relace.
  - Některé atributy jsou mnohem častěji aktualizovány než ostatní

Vertikální dělení

- Jiný příklad
  - Zákazník má id a adresu (ulice, město, psč)
- Jaký má smysl schéma:
  - ZákazníkUlice(id, ulice)
  - ZákaznícMěsto(id, město, psč)

Vertikální dělení – výkonnost

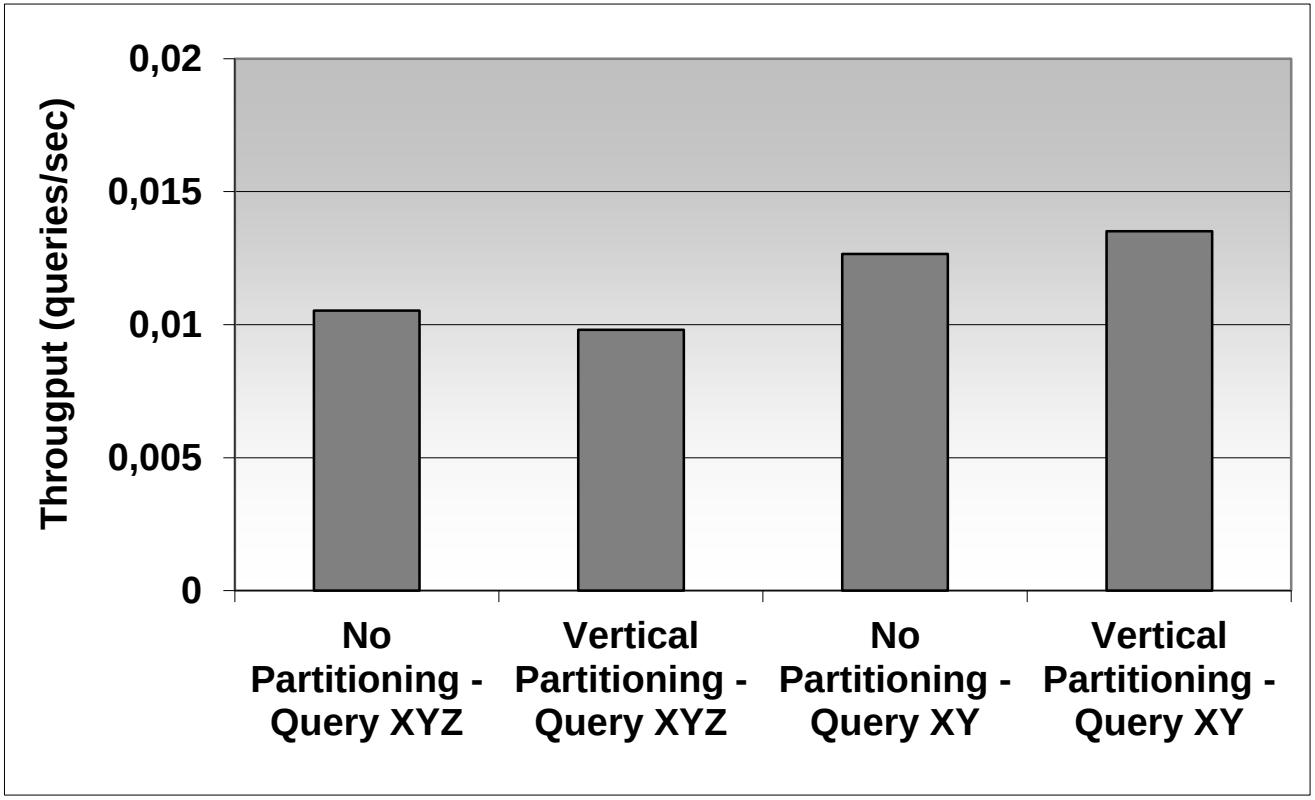
- R(X,Y,Z) - X číslo, Y a Z dlouhé řetězce
  - Rychlost závisí na stylu dotazování

Table-scan

Bez dělení:
R(X,Y,Z)

Vert. dělení:
R1(X,Y)
R2(X,Z)

SQLServer 2k
Windows 2k



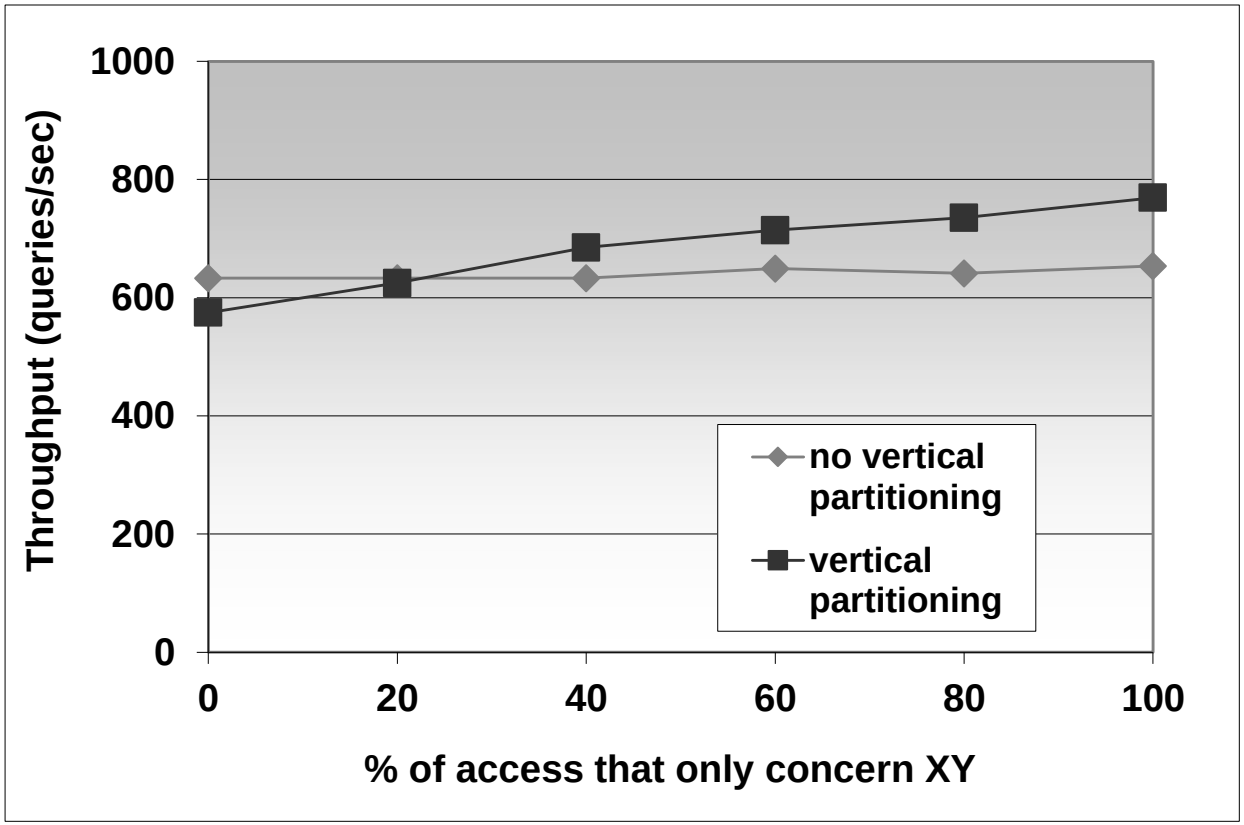
Vertikální dělení – výkonnost

- R(X,Y,Z) - X číslo, Y a Z dlouhé řetězce
  - Selekce X=?, projekce XY nebo XYZ

Index-Scan
(index on X)

Vert. dělení je vhodné, pokud vybírám XY častěji než v 25% případů.

Spojení znamená 2 přístupy do indexu.



Vertikální spojování (antipartitioning)

- Začínáme s normalizovaným schématem
- Přidáváme atributy k jedné z relací
- Příklad
  - Akciový trh
    - Historie cen akcií za posledních 3 000 dní
    - Makléř se rozhoduje hlavně podle posledních 10 dnů
  - Schéma
    - AkcieDetail(akcie_id, datum_vydání, firma)
    - AkcieCena(akcie_id, datum, cena)

Vertikální spojování (antipartitioning)

- Schéma
  - AkcieDetail(akcie_id, datum_vydání, firma)
  - AkcieCena(akcie_id, datum, cena)
- Dotazy na 10ti denní historii jsou náročné
  - I když je index na *akcie_id, datum*
  - Navíc pro další informace je třeba spojení s AkcieDetail

Vertikální spojování (antipartitioning)

- Proveďme replikaci dat
- Schéma
  - AkcieDetail(akcie_id, datum_vydání, firma, cena_dnes, cena_včera, …, cena_před_10_dny)
  - AkcieCena(akcie_id, datum, cena)
- Dotazování na 10ti denní historii je
  - 1x prohledání indexu, není třeba spojení

Vertikální spojování (antipartitioning)

- Nevýhoda
  - Replikace dat
    - Nepříliš velká
    - Lze odstranit neukládáním v AkcieCena
      - → dotazy na průměrné ceny se komplikují, …

Ladění denormalizace

- Denormalizace
  - Porušení normalizace schéma
  - Pouze z důvodu rychlosti!
- Vhodné pro
  - Současný výběr atributů z různých relací
- Nevhodné pro
  - Časté aktualizace dat
    - → vyhledání „zdrojových“ dat kvůli replikaci

Ladění denormalizace

- Příklad (TPC-H)
  - **region**(r_regionkey, ***r_name***, r_comment)
  - **nation**(n_nationkey, n_name, ***n_regionkey***, n_comment)
  - **supplier**(s_suppkey, s_name, s_address, ***s_nationkey***, s_phone, s_acctbal, s_comment)
  - **item**(i_orderkey, i_partkey, ***i_suppkey***, i_linenumber, i_quantity, i_extendedprice, i_discount, i_tax, i_returnflag, i_linestatus, i_shipdate, i_commitdate, i_receiptdate, i_shipmode, i_comment)
  - T(item) = 600 000
    T(supplier) = 500, T(nation) = 25, T(region) = 5
- Dotaz: vyhledání položek (items) evropských výrobců

Ladění denormalizace

- Denormalizovaná relace *item*
  - *itemdenormalized* (i_orderkey, i_partkey , *i_suppkey*, i_linenumber, i_quantity, i_extendedprice, i_discount, i_tax, i_returnflag, i_linestatus, i_shipdate, i_commitdate, i_receiptdate, i_shipmode, i_comment, ***i_regionname***);
  - 600 000 řádků

Ladění denormalizace

- Dotazy:

```
SELECT i_orderkey, i_partkey, i_suppkey, i_linenumber,
        i_quantity, i_extendedprice, i_discount, i_tax,
        i_returnflag, i_linestatus, i_shipdate, i_commitdate,
        i_receiptdate, i_shipinstruct, i_shipmode, i_comment, r_name
FROM item, supplier, nation, region
WHERE i_suppkey = s_suppkey AND s_nationkey = n_nationkey AND
        n_regionkey = r_regionkey AND r_name = 'Europe';
```

```
SELECT i_orderkey, i_partkey, i_suppkey, i_linenumber,
        i_quantity, i_extendedprice,  i_discount, i_tax,
        i_returnflag, i_linestatus, i_shipdate, i_commitdate,
        i_receiptdate, i_shipinstruct, i_shipmode, i_comment, i_regionname
FROM itemdenormalized
WHERE i_regionname = 'Europe';
```

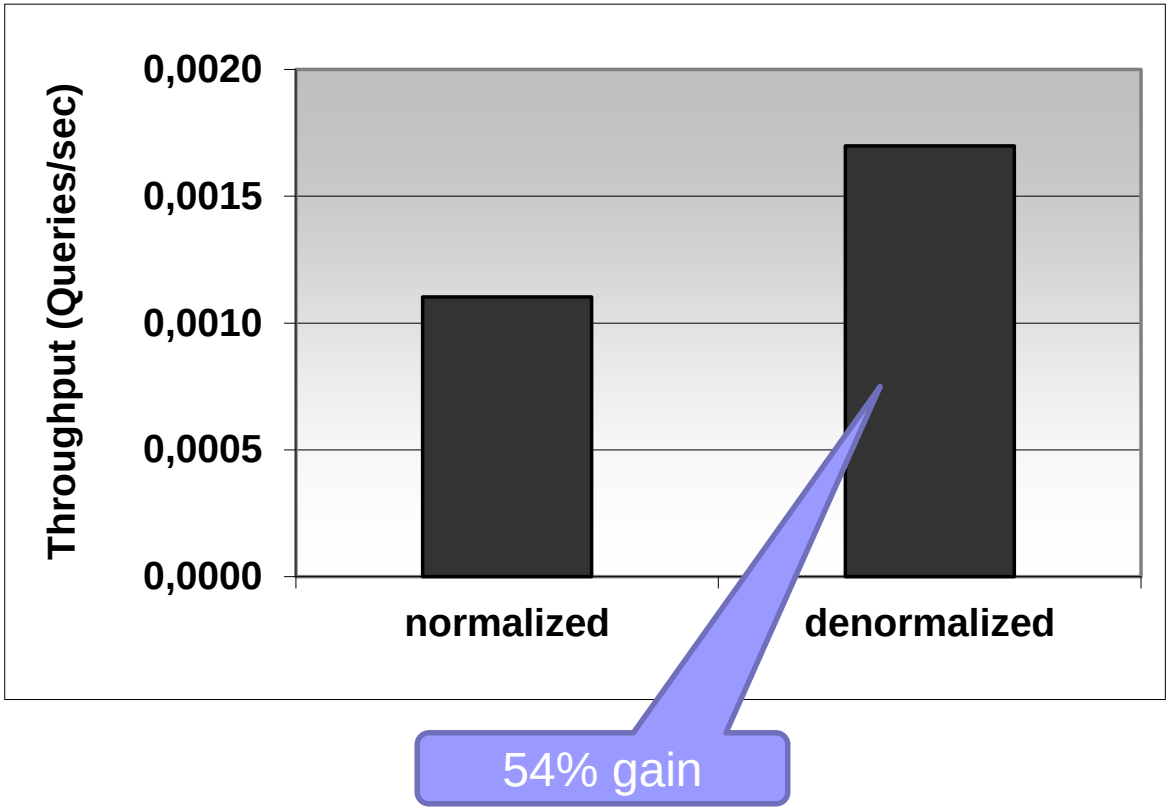
Ladění denormalizace – výkonnost

- Dotaz
  - Najdi všechny položky evropských výrobců

Normalized:
spojení 4 relací

Denormalized:
jediná relace
54% vylepšení

Oracle 8i EE
Windows 2k
3x 18GB disk
(10 000 ot.)



Prokládání relací

- Alternativa k denormalizaci
- Není vždy podporováno DB systémem
- Oracle
  - Prokládání dvou relací
    - Objednávka2(dodavatel_id, výrobek_id, počet)
    - Dodavatel(id, adresa)
  - Uložení
    - U záznamu dodavatele jsou uloženy jeho objednávky

Prokládání relací

- Příklad
  - Objednávka2(dodavatel_id, výrobek_id, počet)
  - Dodavatel(id, adresa)

10, Inter-pro.cz Hodonín 10, 235, 5 10, 545, 10 11, Unikov Bzenec 11, 123, 30 11, 234, 2 11, 648, 10 11, 956, 1	12, Školex Modřice 12, 12, 50 12, 34, 120

...

Horizontální dělení

- Rozděluje obsah tabulky podle řádků
  - Vertikální podle sloupců
- Důvod
  - Snížení objemu dat, se kterým se pracuje
  - Usnadnění mazání
- Použití
  - Archivace dat
  - Prostorové dělení
  - …

Horizontální dělení

- Automaticky
  - Moderní (komerční) DB systémy
    - MS SQL Server 2005 a novější
    - Oracle 9i a novější, …
    - PostgreSQL 10
- Ručně
  - S podporou DB systému
    - Optimalizátor dotazů
  - Bez podpory DB systému

Horizontální dělení

- Změny dotazů:
  - Automaticky
    - Beze změny
  - Ručně
    - S podporou DB systému
      - Beze změny
      - Dědičnost tabulek / definice pohledu (UNION ALL)
    - Bez podpory DB systému
      - Ruční změna dotazu
      - Je třeba upravit seznam používaných tabulek ve FROM části.

Horizontální dělení – SQL Server

- MS SQL Server 2005 a novější
  - Vytvoření dělicí funkce
    - CREATE PARTITION FUNCTION
    - Dělení na intervaly
  - Vytvoření dělicího schéma
    - CREATE PARTITION SCHEME
    - Kam se budou data ukládat (na jaké oddíly úložiště)
  - Vytvoření dělené tabulky
    - CREATE TABLE … ON dělicí schéma
    - Ukládaná data jsou automaticky dělena do oddílů
  - Vytváření indexů
    - CREATE INDEX
    - Indexy jsou vytvářeny na oddílech tabulky, tj. automaticky děleny

Horizontální dělení – Oracle

- Oracle 9i a novější
  - Dělení podle rozsahu, výčtu (seznamu), hašování
    - Podporuje i dvojité rozdělení
      - Oddíly se dělí na pododdíly
  - Přímo v CREATE TABLE
    http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_7002.htm#i2129707
- PostgreSQL 10 a novější
  - dělení podle rozsahu, výčtu a hašování
    - CREATE TABLE … (…) PARTITION BY RANGE (…);

Horizontální dělení - MySQL

- Part of SQL syntax, applies to indexes

```
CREATE TABLE ti (id INT, amount DECIMAL(7,2), tr_date DATE) ENGINE=MyISAM
        PARTITION BY HASH( MONTH(tr_date) )
        PARTITIONS 6
CREATE TABLE ti …
        PARTITION BY RANGE (MONTH(tr_date)) (
                PARTITION spring VALUES LESS THAN (4),
                PARTITION summer VALUES LESS THAN (7),
                PARTITION fall VALUES LESS THAN (10),
                PARTITION winter VALUES LESS THAN MAXVALUE );
```

- Types:
  - hash, range, list; also double partitioning
- Limitation on UNIQUE constraints
  - All columns used in the table's partitioning expression must be part of every unique key the table may have.

Including primary key

Horizontální dělení – PostgreSQL

- PostgreSQL 8.2 a vyšší
  - Dělení podle rozsahu, výčtu (seznamu)
- Princip (http://www.postgresql.org/docs/current/static/ddl-partitioning.html)
  - Využití dědičnosti tabulek
    - Vytvoření základní tabulky
      - Nebude ukládat data, bez indexů, …
    - Jednotlivé oddíly budou zděděné tabulky
      - Pro každou tabulku definovat CHECK omezení povolených dat
    - Vytvoření případných indexů
  - Nevýhoda: nelze používat cizí klíče.

Horizontální dělení – PostgreSQL

- Princip
  - Vkládání záznamů
    - Vkládání do primární tabulky
    - Primární tabulka má pravidla pro vkládání
      - Vkládání pouze do „nejnovějšího“ oddílu → jeden RULE
      - Obecně je třeba mít RULE pro každý oddíl
      - Lze realizovat i triggerem (instead-of type)…
  - V případě použití pohledů
    - Lze definovat *INSTEAD OF* trigger

Horizontální dělení – PostgreSQL

- Příklad v *db.fi.muni.cz*, schéma *xdohnal*
  - Nerozdělená tabulka *account*
    - Primární klíč *id*
    - R(account) = 200 000
    - V(account,home_city) = 5

home_city	count
home_city1	40020
home_city2	40186
home_city3	39836
home_city4	39959
home_city5	39999

  - Rozdělená tabulka *account_parted*
    - Podle home_city (5 oddílů)
      - Oddíly account_parted1 .. account_parted5

Horizontální dělení – PostgreSQL

- Statistiky

Tabulka	Řádků	Velikost	Indexy
account	200 000	41 984 kB	4 408 kB
account_parted	0	0 kB	8 kB
account_parted1	40 020	8 432 kB	896 kB
account_parted2	40 186	8 464 kB	896 kB
account_parted3	39 836	8 392 kB	888 kB
account_parted4	39 959	8 416 kB	896 kB
account_parted5	39 999	8 424 kB	896 kB
Celkem:	200 000	42 128 kB	4 472 kB

Horizontální dělení – PostgreSQL

- Optimalizátor dotazů
  - Povolení kontroly omezení na oddílech

```
set constraint_exclusion=on;  -- or set it to partition
```

- Dotazy (porovnejte plány provádění)

```
select * from account where id=8;
select * from account_parted where id=8;

select count(*) from account where home_city='home_city1';
select count(*) from account_parted where home_city='home_city1';

select * from account where home_city='home_city1' and id=8;
select * from account_parted where home_city='home_city1' and id=8;
```

Dělení relací – shrnutí

- Více menších relací
  - Rychlejší dotazování na taková data
- Nezávislost relací
  - Lze mazat celé relace
- Není replikací dat
  - Každý záznam/hodnota je pouze 1x.
- Fixní uložení
  - Data se mezi relacemi nepřesouvají