Databázové systémy 2

KIV/DB2

1. „Vnitřní“ programovací konstrukce (Embedded SQL) - procedurální prostředky v rámci jazyka SQL.

- **princip**
  - do některého z vyšších programovacích jazyků lze přímo vkládat SQL příkazy označené standardním prefixem
  - **prekompilátorem** poté přeloženy na volání funkční knihovny, která realizuje spojení s DB serverem, překlad požadavků na server a odpovědi serveru
  - soubor z prekompilátoru lze potom **přeložit překladačem** daného programovacího jazyka
- příklad SQL v C
  - **include – EXEC SQL INCLUDE** sqlca.h; - hlavič. soubor s prototypy knihovních funkcí pro komunikaci s DB
    - obsahuje **SQLCA** (Communications Area) – datová struktura zajišťující komunikaci s DB
  - proměnné použité v hostitelském jazyce se v SQL nazývají **vazební proměnné** a nacházejí se v sekci ohraničené:
    - **EXEC SQL BEGIN DECLARE SECTION**
    - **EXEC SQL END DECLARE SECTION**
- **příkazy**
  - **přihlášení k DB –** EXEC SQL CONNECT :myUserId IDENTIFIED BY :myPassword;
  - **odhlášení z DB s commitem/RB –** EXEC SQL COMMIT/ROLLBACK RELEASE
  - **nastavení implicitního zpracování exec**
    - `EXEC SQL WHENEVER SQLERROR CONTINUE;`
    - `EXEC SQL WHENEVER SQLERROR GOTO navesti;`
    - `EXEC SQL WHENEVER SQLERROR DO BREAK;` při chybě opustí smyčku
    - řeší prekompilátor, tzn. že se prochází sekvenčně a chování může být odlišné
- **kurzory**
  - práce jako v PLSQL, pouze s prefixem EXEC SQL
  - často chceme při práci s daty data uzamknout – konkurenční přístup
  - řešení:

```
EXEC SQL DECLARE crsr_osoby CURSOR FOR
 SELECT  prijmeni, jmeno, plat, funkce
  FROM  osoby
  WHERE  plat < 2000
  FOR UPDATE [OF plat, funkce]; - uzamkne pouze dané položky
```

Dynamické SQL

- umožňuje za běhu používat příkazy DDL – vytváření tabulek, pohledů, atd.
- **Příkazy**
  - není dotaz a neobsahuje vazební proměnné
    EXEC SQL EXECUTE IMMEDIATE a pak CREATE TABLE atd..
  - není dotaz a obsahuje vazební proměnné
    prikaz := "**INSERT INTO** osoby (jmeno, prijmeni, plat) **VALUES** (*:par1*, *:par2*, 10000.00)"
    EXEC SQL PREPARE p_vloz FROM :*prikaz*;
    jmeno := "Honza"; prijmeni := "Cervenka"
    EXEC SQL EXECUTE p_vloz **USING** *:jmeno*, *:prijmeni*;
    jmeno := "Honza"; prijmeni := "Zelenka"
    EXEC SQL EXECUTE p_vloz **USING** *:jmeno*, *:prijmeni*;
  - je dotaz a obsahuje vazební proměnné – je jej nutné spojit s kurzorem
    prikaz := "**SELECT** plat **FROM** osoby **WHERE** jmeno = *:par1* **AND** prijmeni = *:par2*";

    EXEC SQL PREPARE p_vyber **FROM** *:prikaz*;
    EXEC SfQL DECLARE c_vyber **CURSOR FOR** *:prikaz*;

    EXEC SQL OPEN c_vyber **USING** *:jmeno*, *:prijmeni*;
    EXEC SQL WHENEVER NOT FOUND DO BREAK;

    for (;;) {
    EXEC SQL FETCH c_vyber **INTO** *:osoba*;
    …
    }
    EXEC SQL CLOSEEXEC SQL COMMIT WORK RELEASE;

PL/SQL

- blok PL/SQL vkládání mezi direktivy **EXEC SQL EXECUTE** a **END EXEC**
- musí se někde nastavit přístup do databáze, který má kompilátor PL/SQL, aby se daly ošetřit chyby
- symbolické jméno DB – umožňuje se připojit k více databázím
  - **EXEC SQL DECLARE jmeno_db DATABASE;**
  - a potom se všude za **EXEC SQL** vkládá **AT jmeno_db**

2. Jazyk PL/SQL. Kurzory – definice, klasifikace, použití kurzorů. Uložené procedury a funkce, balíky (packages), kompilace, spouštění

PL/SQL

- SQL je deklarativní, PL/SQL je jazyk, který rozšiřuje SQL o proceduralitu
- **deklarace**
  - konstant, proměnných, kurzorů,
- **podpora**
  - transakcí
  - výjimek, modularita
- **struktura programového bloku**
  - deklarační část – kurzory, proměnné, konstanty atd. – **DECLARE**
    - jméno proměnné + [const] + datový typ + [:= hodnota]
    - **přístupové proměnné**
      - tabulka%ROWTYPE
      - tab.column%TYPE
  - výkonná část - funkční logika – **BEGIN**
    - obsahuje příkaz; příkaz; atd…
    - přiřazení jako v Pascalu „**:=**“
    - IF podm THEN příkaz; ELSIF …. ELSE …. END IF;
    - CASE promenna, potom na řádkách WHEN vyraz THEN …; ELSE příkaz END CASE
    - LOOP … END LOOP; - opuštění smyčky EXIT (`EXIT WHEN podm`)
    - RAISE nazev_exc – do řízení exception
  - část pro zpracování výjimek - řeší vzniklé chyby – **EXCEPTION**
    - následuje WHEN nazev_excep THEN …
    - WHEN zero_divide (definován exception pro dělení nulou)
  - konec – **END**;
- **kurzor:** pracovní oblast obsahující data, které lze dále využívat prostřednictvím operací nad kurzory
  - **klasifikace**
    - **implicitní –** na jednom řádku
    - **explicitní –** deklarujeme v části DECLARE pomocí klíč. slova DECLARE
  - **deklarace**
    - DECLARE CURSOR cursor_name IS SELECT seznam FROM tabulka;
  - **použití:**

```
OPEN cursor_name;
LOOP
   FETCH cursor_name INTO p_jmeno;
   dbms_output.put_line(p_jmeno);
   EXIT WHEN cursor_name%NOT_FOUND;
END LOOP
```

  - a zakončíme uzavřením kurzoru – CLOSE cursor_name
  - WHERE CURRENT OF cursor_name – odkaz na aktuální položku kurzoru
  - **zjištění stavu kurzoru – cursor_name%…**
    - **FOUND –** obsahuje záznamy?
    - **NOT_FOUND –** neobsahuje záznamy?
    - **ISOPEN**
    - **ROWCOUNT –** dosud zpracováno řádků?

Procedury a funkce

- bloky příkazů v PLSQL lze pojmenovat a uložit, ve spustitelné formě, do DB
- v DB jsou uložené ve zkompilovaném tvaru
- mohou volat další proc/fce či samy sebe
- **funkce –** vrací jednu hodnotu
- **procedura –** nemusí vracet žádnou hodnotu nebo libovolné množství hodnot
- **příkazy**
  - **vytvoření procedury:**
    ```
    CREATE PROCEDURE nazev_procedury [form. parametry] AS [lokalni deklarace]
      [prikazy]
      [EXCEPTION ]
    END;
    ```
  - **zrušení –** `DROP nazev_procedury/nazev_fce`
  - **parametry –** oddělené čárkou, nazev | typu IN / OUT / IN OUT | typ
    - **inicializace –** pouze vstupních parametrů
    - `CREATE PROCEDURE nazev_procedury (id IN NUMBER) AS ...`
- **kompilace a spuštění**
  - zápis ukončíme znakem **. (tečka)** na novém řádku
  - **RUN** přeloží proceduru
  - **EXECUTE [seznam parametru]** proceduru spustí
- **funkce**
  - zápis jako u procedury, jen na začátku **navíc** CREATE FUNCTION `nazev_fce` **RETURN typ**
  - v těle potom return jmeno_p;

Balíky

- slouží ke sdružení logicky spolu souvisejících proc a fcí
- mohou obsahovat globální proměnné – životnost omezena délkou spojení s DB
- definice balíku představuje **definice rozhraní** balíku a **těla** balíku
- **definice rozhraní:**
  ```
  CREATE OR REPLACE PACKAGE nazevpkg AS
    usage INTEGER
    ...hlavicky fci, proc…
    FUNCTION  add(a IN INTEGER, b IN INTEGER) RETURN INTEGER;
    FUNCTION  sub(a IN INTEGER, b IN INTEGER) RETURN INTEGER;
    PROCEDURE inc(a IN OUT INTEGER);
  END;
  ```
- **definice těla:**
  ```
  CREATE OR REPLACE PACKAGE BODY nazevpkg AS
   … definice těla funkcí a procedur
  ```
- **použití balíku:**
  ```
  a:= nazevpkg.add(3,4);
  ```
- **zrušení balíku [těla]**
  `DROP PACKAGE [BODY] nazev_pkg`
  - rozhraní balíku může existovat bez těla

3. Aktivní databáze – klasifikace a spouštění triggerů, oblast použití. Transakce, dvoufázový uzamykací protokol

Trigger

- uživatelsky definovaný blok PL/SQL sdružený s určitou tabulkou
- je implicitně spuštěn, jestliže je nad tabulkou prováděn aktualizační příkaz
- **typy triggerů a podoba**
  - BEFORE / AFTER – **kdy se spustí**
  - INSERT / UPDATE / DELETE– **na jaký dotaz**
  - omezení triggeru (nepovinná klauzule **WHEN**)
  - akce triggeru – **blok PL/SQL**
- může být buďto příkazový a nebo řádkový – **klíčové slovo FOR EACH ROW**
  - řádkový se spustí pro každý aktualizovaný řádek tabulky
- **vytvoření:**

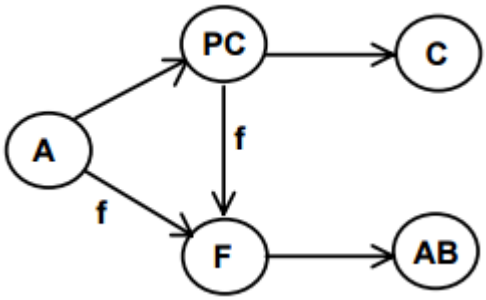
```
CREATE [OR REPLACE] TRIGGER jmeno
  typ_triggeru_kdy spousteci_akce [OF sloupec, sloupec] ON tabulka
  [FOR EACH ROW] [WHEN podmínka]
BEGIN
  …
END;
```

- **hodnoty v triggeru – :OLD a :NEW –** logicky OLD u insert null a NEW u delete null
- **použití**
  - zajištění referenční integrity – vyhodíme výjimku
  - aktualizace údajů (při změně data narození chceme automaticky aktualizovat věk, atp.)
  - logování (chceme vědět, že uživatel provedl aktualizaci nějaké tabulky, atp.)
  - ohlídání business rules
- **postup spouštění triggeru**
  - do ORACLE DB předán příkaz INSERT/UPDATE/DELETE
  - provede se příkazový trigger BEFORE
  - pro každý řádek, kterého se SQL týká se provede
    - řádkový trigger BEFORE
    - změní se řádek a provedou se kontroly integritního omezení
    - řádkový trigger AFTER
  - dokončí se odložené kontroly IO s ohledem na přechodná porušení
  - provede se příkazový trigger AFTER
  - návrat do aplikace
- **aktivace a deaktivace -** po vytvoření defaultně aktivní
  - **jednoho –** ALTER TRIGGER trigrname ENABLE | DISABLE
  - **všech pro tabulku –** ALTER TABLE tablename ENABLE | DISABLE ALL TRIGGERS

Transakce

transakce – jednotka provádění programu, která zpřístupňuje, případně i modifikuje data v databázi

- **ACID vlastnosti**
  - **atomičnost** (Atomicity) – buď se provede celá transakce nebo žádná z databázových operací, které ji tvoří
  - **konzistence** (Consistency) – izolovaná transakce zachovává konzistenci databáze
  - **izolace** (Isolation) – souběžné transakce se vykonávají sekvenčně
  - **trvalost** (Durability) – potom co transakce skončí, mají změny v DB trvalý charakter – i při výpadku systému
- **Stavy transakce**



- **aktivní A –** počáteční stav, transakce v něm setrvává po dobu provádění
- **částečně potvrzená PC (PARTIALLY COMMITED) –** po provedení posledního příkazu
- **chybový stav F –** po zjištění, že normální provádění není dál možné
- **zrušená AB (ABORTED) –** poté, co byly změny v DB provedené transakcí anulovány (RB), DB ve stavu před spuštěním transakce
- **potvrzená C (COMMITED) –** po úspěšném dokončení transakce

- **typy transakcí – COMMIT –** commit příkazů **/ ROLLBACK –** zahození změn

Dvoufázový uzamykací protokol

- soustava pravidel stanovující, kdy může transakce uzamknout, resp. odemknout DB objekt
- 2 fáze
  - **1) fáze růstu (growing) –** transakce uzamyká podle potřeby objekty, ale žádný neodemyká
    - konec fáze – tzv. **lock point**
  - **2) fáze zmenšování (shrinking) –** transakce odemyká objekt, ale již nesmí žádný uzamknout
    - zajišťuje uspořadatelnost vzhledem ke konfliktům, ale nevylučuje možnost zablokování
- modifikace
  - **striktní 2PL –** všechny **výlučné** zámky uvolňuje transakce až ve stavu potvrzení
    - zabraňuje kaskádnímu rušení transakcí
  - **rigorózní 2PL –** všechny zámky uvolňuje transakce až ve stavu potvrzení

4. Optimalizace dotazu, jednotlivé přístupy (např Cost Based optimalizace (CBO)), podstata optimalizátoru, přínos optimalizace

- **dvěma** či více různými dotazy je možné získat **stejná data – rychlost** dotazů může být ovšem **rozdílná**
- **cíl –** nalézt nejefektivnější plány, které vedou ke stejnému výsledky
  - **selekce** by se měli vyhodnocovat dřív
- **zavedení indexů => přínos optimalizace –** minimalizace nákladů na
  - zdrojový čas
  - kapacitu paměti
  - programátorskou práci
- **podílí se**
  - návrhář DB, vývojář, správce DB i uživatel
- **postup zpracování SQL dotazu**
  - uživatel zadá SQL příkaz – SELECT * FROM LIDE
  - příkaz se zpracuje v parseru
  - potom se použije nastavený **optimalizátor**
  - generátor řádkových zdrojů
  - vlastní provádění SQL řekne
- **plán vyhodnocení –** strom dotazu + algoritmus pro každou operaci
- **myšlenky –** jaké plány se uvažují pro dotaz a jak se odhadne cena plánu
- **podstata –** jednotlivé akce si optimalizátor ohodnotí nějakou cenou a pak pro daný plán určí celkovou cenu

Pravidla pro vytváření dotazů

- používat **SELECT** jenom na sloupce který mi zajímají – většinou pracujeme stejně jenom s některými
  - nepoužívat * (hvězdičku), pořád je lepší vypsat všechny sloupce (alespoň DB nemusí jejich názvy zjišťovat)
- při více podmínkách začínat tou nejobecnější
- výběr vhodného pořadí spojení
- používat **hint**, **indexy a limit**
  - **hint:** mohu optimalizátoru vnutit některou operaci, ptž si myslím, že její užití přispěje k lepší optimalizaci
- používat minimálně **IN, NOT IN, LIKE**

Optimalizátor

- jádro celého zpracování, analyzuje sémantiku dotazu
- hledá optimální způsob jeho provádění
- v ORACLE CBO (Cost Based Optimization) a RBO (Rule Based Optimization)
- generátor plánů a odhadovač ceny, který pro každý vygenerovaný plán odhadne cenu
- výstup –> plán vykonávání, který určuje
  - přístupové cesty k jednotlivým tabulkám používaných v dotazu
  - pořadí jejich spojování

Přístupy optimalizace

Rule Based Optimizer

- vyhodnocuje jednotlivé přístupové cesty pomocí předem daného systému pravidel
- v Oracle označen jako *deprecated*

Cost Based Optimizer

- používá slovník se statistikami, hledá plán s nejmenšími náklady, doporučené společností Oracle
- pro jednotlivé plány se počítá cena provedení v řadě hledisek
  - množství: IO operací, řádek, byte
  - cena prováděných třídících operací, cena za HASH operace
- **statistiky** – řada údajů o DB objektech (tables, indexes, …)
  - přístupné i z vnějšku – uložené v tabulkách, pohledech
  - výpočtem nebo odhadem se aktualizují
  - typy
    - údaje o tabulkách – počet řádků, bloků, nevyužitých bloků, prům. délka záznamu
    - údaje o sloupcích – počet unikátních / NULL hodnot, histogram dat
    - údaje o indexech – počet listových bloků, počet úrovní, cluster faktor

5. Postrelační databáze – vysvětlení principu, výhody a nevýhody

- - - - - viz 9. přednáška důvody vzniku - - - - -

Postrelační databázový systém – relační DBS rozšířený o nějakou specializaci na DB úrovni, jelikož aplikační řešení by bylo nedostačující (dnes prakticky všechny DB post-relační – obsahují nějaká rozšíření oproti původnímu relačnímu schématu (např. triggery)

Příklady postrealčních DBS:

- **prostorové DB –** rozšířeny o práci s prostorovými objekty a vztahy mezi nimi
  - složité na návrh a školení uživatelů, drahá implementace
  - jsou schopné zpracovávat prostorová data v podobě jednoduchých geometrických entit (obrovského množství)
- **objektově orientované DB –** rozšířeny o objektový model dat a vazby
  - nedostatek standardů, nedostatek zkušeností, chybí univerzální datový model
- **deduktivní DB –** rozšířeny o funkce pro analýzu dat
  - poskytují matematickou logiku (vztahy) spolu s relačními databázemi (znalosti) a využívají znalostí a vztahů k dedukci závěrů
- **temporální DB –** rozšířeny o temporální logiku
  - temp. db jsou databáze rozšířené o časovou dimenzi
  - používají se v bankovnictví, účetnictví a podobně
  - mají dobré vlastnosti pro archivaci a monitorování změn
- **multimediální DB –** rozšířeny o fce pro práci s multimediálním obsahem (např. Oracle interMedia)
  - vyžaduje zkušené uživatelé, high end hardware, zbytečný overhead
- **aktivní DB –** rozšířeny o aktivní pravidla

Specializované nerelační DB (post-relační)

- **pro specifiky strukturovaná data**
  - čistě objektové či XML DB, úložiště tagovaných dokumentů atp
- **pro specificky uložená/přístupová data**
  - in memory, embedded a real time DB

6. Objektové vlastnosti SQL99, rozšíření datových typů. Vlastnosti objektově orientovaného datového modelu, možnosti použití

- - - - - TODO tady docela chaos - - - - -

Objektové vlastnosti SQL99

- **rozšiřitelnost –** nové datové typy UDT
  - abstraktní datové typy
  - řádkové typy
  - reference
- možnost vytvářet vlastní strukturované datové typy zapouzdřující data a operace (např porovnávací operace)
  - metody buďto v SQL nebo v C, Javě

Rozšíření datových typů

- standardizace typů pro objemná data
  - typ **BLOB** (**B**inary **L**arge **O**bjecs)
  - typ **CLOB** (**C**haracter **L**arge **O**bjects)
    - omezení –> nemůžou být PK, FK, UNIQUE, ani použity v GROUP BY nebo ORDER BY
  - možnosti interpretace jejich vnitřního obsahu – texty, obrázky, zvuky, videa
  - možnost doplňování dalších formátu a manipulace s nimi
- nové uživatelské datové typy – UDT
  - abstraktní datové typy – ADT / strukturovaná data
    - mohou tvořit hierarchii s jednoduchou dědičností
    - podporován polymorfismus metod a dotazů nad daty
  - řádkové typy
    - vytvoření odpovídající tabulky jako u ADT, instance nejsou identifikovány OID
      - mohou být použity i jako typy sloupců při vytváření tabulky
  - reference (odkaz na řádek referencovatelné tabulky)
  - typ **ARRAY** – VARCHAR (15) ARRAY[5] – umožňuje ukládat kolekce do jednoho sloupce (př. dny v týdnu)
  - typ **ROW** – umožňuje ukládat strukturovaná data do jednoho sloupce (př. adresa – ulice, ČP, město)
- **abstraktní datové typy**
  - příklad – CREATE TYPE T_Students AS (Jm CHAR(30)….) UNINSTANTIABLE NOT FINAL
    - METHOD POC_PREDNASEK() RETURNS INTEGER;
  - [UN]INSTANTIABLE je možné povolit či zakázat vytváření instancí daného typu
  - [NOT]FINAL je možné zakázat či povolit vytváření potomků daného typu
  - **vytvoření metody:**

```
CREATE METHOD Poc_Prednasek()
 FOR T_Student
  BEGIN
    …
  END;
```

  - **vytvoření odvozeného typu:**

```
CREATE TYPE T_TphDSTudent
UNDER T_Student …
```

Vlastnosti objektově orientovaného datového modelu

- ADT mohou tvořit hierarchie s jednoduchou dědičností, podporován polymorfismus metod a dotazů nad daty
- vytváření tabulky instancí abst. datového typu
  - `CREATE TABLE PhD OF T_PhDStudent`
  - každá řádka identifikovaná pomoc umělého OID klíče
- na instanci ADT se lze odkázat pomocí typu reference REF
  - potom po výběru prvku se odkazujeme např Majitel->Jmeno
  - pokud odkazovaný objekt neexistuje, potom REF ve stavu DANGLING
- povolené hodnoty je možné omezit jen na reference do dané tabulky pomocí
  - REF(T) SCOPE Tabulka
- základní typ bez předků, odvozený typ, implementace metod, zrušení typu

7. SQL/MM – základní rámec normy, full-textová data, prostorová data, obrázky

Rámec normy

- **full-textová data**
- **prostorová data**
- **obrázky (statické i videa)**

Framework

- jednotlivé části SQL/MM jsou na sobě poměrně nezávislé
- část společná a závazná zbytku světa
- **poskytuje**
  - definici společného konceptu užitého v dalších částech SQL/MM
  - hlavní rysy přístupu k definici těchto částí

Fulltext

- **def:** textová data, lišící se od běžných znakových řetězců
  - obvykle delší záznamy
  - specifické operace
  - způsob indexování
    - index slov v dokumentu
    - index vzájemných vzdáleností slov a frází
- **konstruktor –** řetězce znaků [+ zadání jazyka]
- **konverze** do běžných SQL znak. řetězů – FullText_to_Character
- **vyhledávací metody –** ano/ne CONTAINS, rank
- **vyhledávání**
  - vzorek + wild cards
  - odvozená slova(STEMMED)
  - slova s podobným nebo odvozeným významem(SYNONYMS)
  - stejně znějící slova(SOUNDS, LIKE)
  - dle pozice v textu(NEAR)
  - dle konceptu textu
- **podpora jazyků**
  - počítá se především s podporou jazyků, kde je snadné výpočetně rozeznat jednolité tokeny
- příklad: `CREATE TABLE informace ( číslo_doc INTEGER, dokument FULLTEXT);`
- dotaz:
  - `WHERE dokument.CONTAINS( 'STEMMED FROM OF "standard" IN SAME PAR. AS SOUNDS LIKE "sequel") = 1`

*Spatial (**SQL/MM Spatial**)*

--- viz otázka 12 – Prostorové databáze ---

- **definuje**
  - ukládání, výběr, dotazování a aktualizaci jednoduchých prostorových objektů
  - reprezentaci prostorových objektů pomocí **prostorových datových typů**
  - **funkce** pro práci s prostorovými objekty – 0,1,2 dimenzionální
- **rozdělen** do několika klauzulí
  - prostorové datové typy a jejich metody
  - datový katalog
- **datové typy**
  - **body –** pouliční lampa, lavička
  - **křivky –** koryto řekni, hranice pobřeží
  - **polygony –** půdorysy budov, území států
  - funguje dědičnost

- vychází z geometrického modelu OGC
  - rozdíly – místo Line a `LinearRing` zavádí nový typ `ST_LineString`
  - nové typy pro repr. křivek a povrchů tvořených kruhovými oblouky – `ST_CircularString`
  - u kolekcí není zřejmé z jakých se skládá jednoduchý typů – `ST_MultiPoint` je z `ST_Point`??
- prostorový objekt – svázán s nějakým prostorovým systémem souřadnic – **Spatial Reference Systém** – některé resp. i různé zakřivení země
  - **ST_SpatialRefSys** – typ který určí systém souřadnic
  - všechny hodnoty sloupců musí být ve stejným systému souřadném
- pro reprezentaci prázdného prostorového objektu neexistuje samostatný typ – instance každého prostorového typu může být prázdný prost. Objekt
  - pokud návratový typ není spec., pak návratovou hodnou bod
- **operace**
  - průnik, sjednocení, sousedi
  - relace – objekt A obsahuje B?
  - Vzdálenost
- **4 kategorie metod**
  - konverze do/z externích datových formátů
    - WKT – well known text (point(10,10), multipolygon(1 1 , 2 2, 1 2, 11)
    - WKB – well known binary
    - GML – geography markup lang – xml zápis
  - práce s atributy prostorových objektů
    - datové typy mají společné atributy (např. Dimenze)
    - podtyp si přidávají některé specifické atributy
    - `ST_X` – souřadnice X, `ST_Length` – délka křivky, `ST_Boundary` – hranice
  - porovnávání prost. Objektů
    - `ST_Equals` – prost. rovnost, `ST_Touches` – dotýkání, `ST_Within` – obsah
  - generování nových prost. objektů – výsledkem operací
    - `ST_Intersection`, `ST_union`, `ST_difference`
    - algoritmu na jeden objekt- `ST_ConvexHull`

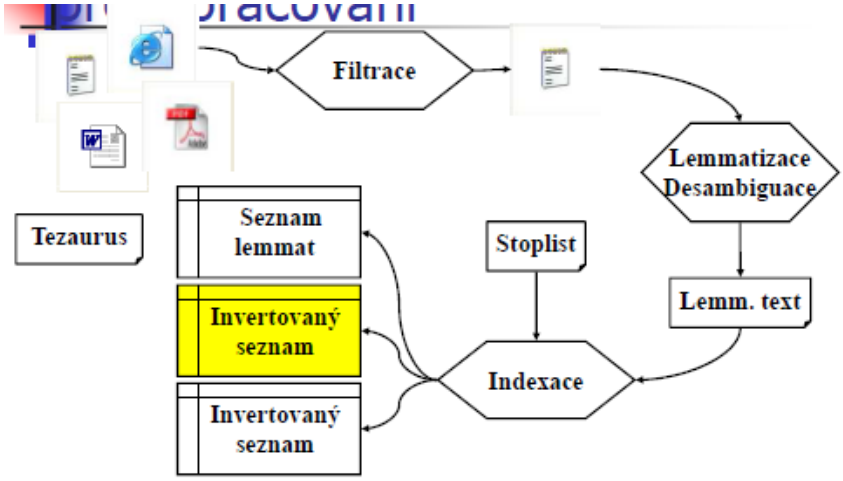
Still Image

- **obrázky** = hodnotná data
- ukládání obrázků
- úprava obrázků
- vyhledávání obrázků – dle vizuálních vlastností
- datové typy
  - `SI_StillImage` – obrazová data
  - `SI_Feature` – vlastnosti obrázku
    - užitečné při vyhledávání, má podtypy
      - `SI_AverageColor`, `SI_ColorHistogram`, `SI_PositionalColor`, `SI_Texture`
    - metoda `SI_Score` – vrací podobnost obrázku <0, 1>
  - `SI_FeatureList` – seznam pro vlastnosti obrázku (i všechny třeba)
- konstruktory
  - BLOB [+FORMAT] – JPEG, TIFF, GIF
- metody
  - pro přeformátování – SI_ChangeFormat
  - vytvoření miniatury,
  - změna velikosti,
  - ořezá**ní rotace**

8. Zpracování full-textových dat, XML databáze – charakteristické vlastnost, výhody a nevýhody.

Zpracování full-textových dat (TODO – tak na půl)

- je třeba mít možnost třídit odpovědi podle předpokládané vhodnosti pro tazatele
  - je nutné mít možnost vyjádřit **míru shody**
- hledají se homonyma, synonyma, slova stejně znějící, slova s nějakou vazbou na hledané slovo – hierarchie (auto = BMW) či ohnutá slova (jdu = jdou)
- **obvyklý postup předzpracování**
  - DB obvykle používají nějaký z boolovských modelů reprezentace modelů
    - **snadná implementace**
    - nejlépe odpovídá běžným dotazům
    - dotazy ve formě boolovských formulí, kde operandy tvoří jednotlivá slova



  - **postup**
    - **filtrace –** odstraní se formátovací značky a nechá se čistý ASCII text
    - **desambiguace –** určí význam slova podle kontextu
      - **pět** chválu – sloveso pět
      - **pět** vozidel – číslovka
    - **lemmatizace –** určí se základní tvar slova a gramatický tvar v dokumentu
      - nahrazen často pomocí **stemmeru** => hledá kmen slova
    - **indexace**
      - vytvoří se pomocné seznamy lemmat a dokumentů a invertovaný soubor
      - dvojice[id_dok, id_lemmatu] setříděné dle id lemmatu a zbavené duplicit
      - dnes vice info – ještě č. odstavce, věty, slova…
  - **fulltext vyhledávání**
    - třeba vytvořit nad textovým sloupcem index – invertovaný soubor
    - běžné textové sloupce pro tyto účely krátká a nevyhovující
      - indexují se obvykle sloupce některého z **LOB** typů
        - BLOB, CLOB, NCLOB (native) – až 4gb
      - ve sloupcích pouze **deskriptor –** LOB lokátor, odkazuje na samostatně uložená data

XML databáze

- **vlastnosti**
  - oproti mnoha malých XML dokumentů několik XML databází
  - obvykle dynamické
  - explicitní struktura – schéma
  - záznamy
  - přizpůsobený stroj
  - obsah – schéma, data, metody
  - paradigmata – atomicita, souběžnost, izolace, trvanlivost
  - metadata – popis schématu
- **výhody**
  - univerzální formát pro výměnu informací s mnoha výhodami
  - snadná transformace do jiných formátů
  - ideální pro ukládání dokumentů
  - univerzální pro datové modelování
  - snažší (přirozenější) reprezentace objektů
- **nevýhody**
  - tam kde stačí relační, raději relační
  - aplikace se specifickými požadavky na výkon – XML pomalejší
  - v plenkách
    - nedosahují takové robustnosti – transakce, souběžný přístup, škálovatelnost
    - chybí podpora standardů
    - metody pro indexování a optimalizaci se zatím vyvíjí
- **práce s databází**
  - aktualizace
  - čistění dat
  - dotazování
  - skládání / transformování
- **mezi dokumenty a databázemi** neexistuje jasná hranice
- **legitimní** oba přístupy – někde uprostřed semi-strukturovaná data, která jsou neuspořádaná či neúplná
  - struktura se může měnit, dokonce nepredikovatelně
  - HTML, Bibtexovské soubory
- **XML** – značkovací jazyk rodiny SGML – založen na logickém vyznačování
  - stromová struktura
  - jediný kořenový element
  - součástí mohou být atributy – dvojic klíč - hodnota
  - tag obsah /tag
- **DTD – Document Type Definition**
- **termíny**
  - **XML –** specifikace XML jazyka
  - **XSD/DTD –** způsob specifikace XML dokumentu; dokumenty splňující strukturu se označují jako validní
  - **XSLT –** způsob transformace XML dokumentů – do mnoha výstupních formátů XML, HTML, WML, PDF…
  - **XLink/XPointer –** specifikuje, jak jsou jednotlivé XML dokumenty (případně jejich části) spojeny dohromady
  - **XPath/XQL/XML-QL/Quilt/XQuery –** dotazovací jazyky určené pro hledání v XML dokumentu dle určitých vyhledávacích kritérií

XML-QL

- dotazovací jazyk, určen pro elektronickou výměnu dat(EDI – electronic data interchange)
- vychází z předpokladu že XML dokument odpovídá databázi a jeho databázovému DTD
- více **datově** a **databázové** orientovaný, než ostatní dotazovací jazyky
- vychází z jazyka SQL a byl navržen na standard, ale nebyl schválen
- níže uvedená konstrukce vrací struktury, které mají v NAME Amazon.com a vrací jejich telefonní číslo
  - **CONSTRUCT –** vytváří výsledek (jako resultSet v SQL)

XQL

- podobný XPath, něco navíc, něco naopak není třeba
- navržen jako jednoduchý dotazovací jazyk použitelný v různých XMl prostředích
- pokouší se dívat na několik XML dokumentů jako na databázi
- výsledkem dotazu množinu uzlů, které lze zpracovat nebo použít pro další dotazy
- navržen Microsoftem, nakonec konsorcium vybralo jako doporučení k implementaci XPath
- **př:** Book[Author=”Karel Čapek”][1]

XPath

- standard pro procházení XML dokumentů
- přesná definice polohy prvku v XML dokumentu
- podobné cestě k souboru
- hledaným prvkem může být i komentář či jmenný prostor
- **př:** //zakaznik[@id=”2001”]

XQuery

- připravovaný standard W3C
- vychází z Quilt jazyka
- inspirace u XPath, XQL, XML-QL…

9. NoSQL DB – charakteristika, porovnání ACID a BASE

Charakteristika

- databáze, které nejsou založené na datovém modelu – **nejsou relační**
- dovedou zpracovávat **obrovské množství dat** v reálném čase
- neřeší se transakční zpracování **(ACID)** , ale data jsou snadno dostupná vždy i v částečně konzistentním stavu
- podporují **nerelační datový model**
- podporují **distribuovanou architekturu –** lze použít jako centrální DB, ale síla v distr.
- většina DB NoSQL je open source
- řeší většinou **CAP** omezením konzistence dat - **BASE**
- **kategorizace**
  - **klíč-hodnota (key-value) –** do DB se ukládá klíč a hodnota
    - podle klíče jsme schopni získat hodnotu
    - **zástupci: Redis, Riak**
  - **sloupcové (**big table**) –** ke každému klíči je možné uložit více hodnot odpovídající příslušnému sloupci
    - každý klíč může mít vyplněné hodnoty **jiných** sloupců – heterogenní záznamy
    - **zástupci: Hadoo, Cassandra**
  - **dokumentové (**document**) –** do DB se ukládají **dokumenty** ve formátech **JSON** a **BSON**
    - každý ukládaný dokument může mít **jinou strukturu**
    - **zástupci: MongoDB, Elasticsearch**
  - **grafové (**graph**) –** do DB se ukládají uzly, jejich vlastnosti a hrany mezi nimi
    - hlavním přínosem jsou implementované **algoritmy pro prohledávání** grafů a vyhledávání příslušných uzlů
    - neporovnatelně **rychlejší** než běžná DB
    - **zástupci: Neo4j, HyperGraphDB**

ACID vs BASE

- **ACID**
  - **Atomic** *(=atomicita)* **–** atomičnost transakcí, žádný rozpracovaný stav a to i ve vztahu k možné chybě OS nebo HW (celá transakce proběhne nebo nic)
  - **Consistency** *(=konzistence)* **–** v DB jsou pouze platná data podle daných pravidel. izolovaná transakce zachovává konzistenci DB
  - **Isolation** *(=izolovanost)* **–** souběžné transakce se neovlivňují – serializace, pořadí není zajištěno
  - **Durability** *(=trvalost)* **–** uskutečněná transakce nebude ztraceni, a to ani po pádu SW nebo HW
  - nevýhody => netriviální, omezuje změny dat (zamykání) a přístup k datům (rychlost)
- **teorém CAP u sdílených distribuovaných systémů**
  - **Consistency –** každý uzel/klient vidí ve stejnou chvíli stejná data
    - konzistentní nezávisle na běžících operacích či jejich umístění
  - **Availability** *(=dostupnost)* – každý request dostane odpověď, jestli byl nebo nebyl úspěšný
    - nepřetržitý provoz, vždy možnost zapsat a číst data
  - **Partition tolerance** *(=odolnost proti rozdělení)* **–** funkční navzdory chybám sítě nebo výpadkům uzlů
    - možnost výpadku části infrastruktury (odstávka pro údržbu), databáze je schopna korektně fungovat, i když je přerušena komunikace mezi jednotlivými servery.
  - **teorém** – u sdílených systémů je možné uspokojit pouze 2 ze 3 požadavků
    - **CA** – Postresql, MySQL (relační), Vertica (sloupcová)
    - **CP** – BigTable (sloupcová), Mongo (dokumentová), Redis (Key-Value)
    - **AP** – Cassandra (sloupcová), CouchDB (dokumentová), SimpleDB (dokumentová)
- **požadavky na moderní DB**
  - **cloud, distribuovaná DB**
    - decentralizace úložiště dat, redundance pro odolnost proti výpadkům a rychlost, velké objemy dat a velké množství operací
  - **problematické datové typy**
    - údaje klíč-hodnota, objekty, nestrukturované dokumenty...
  - **iterativní vývoj**
    - časté změny schématu DB, nebo dokonce žádné schéma

- **vysoké požadavky na škálovatelnost**
  - mobilní zařízení jako klienti
  - nerovnoměrné zatížení prostorové i časové
  - spec. požadavky na dostupnost
- **relační DB přestává odpovídat relačnímu konceptu**
  - nejedná se o matematické relace, ale spíše kolekce/množiny/ grafy nestr. dat
- **dodržování ACID omezuje práci s DB**
  - úmyslné zanedbání/odpuštění A , C, I nebo D – pro vznik rychlosti a dostupnosti dat

- **Base**
  - řeší CAP omezením konzistence dat
    - Basically Available – aplikace pracuje víceméně pořád
    - Soft state – nemusí být stále konzistentní
    - Eventual consistency – nakonec se do konzistentního stavu dostanou
  - nekonzistence řešeny při čtení (verzování, nevalidní cache), zápisu (distribuce změn) nebo asynchronně (replikace dat)

- **ACID vs BASE**
  - silná konzistence X slabá konzistence,
  - izolovanost X dostupnost na prvním místě
  - orientace na commit X přibližné odpovědi jsou ok
  - vnořené transakce X jednodušší, rychlejší
  - dostupnost? X dodávka jak to jen půjde,
  - konzervativní (pesim.) X agresivní (optimistické)
  - složitá evoluce (schématu..) X jednodušší evoluce

10. Kategorie NoSQL DB na základě datového modelu, příklady architektur

Kategorie NoSQL DB na základě datového modelu

1) **Key-Value**
   - jeden klíč, jedna hodnota, žádný duplikát
     - klíč může být složený (z hlavní a upřesňující části)
   - přístup podle klíče přes hash tabulky => brutálně rychlé
   - hodnotou BLOB – databáze se nesnaží chápat
     - zpracování hodnoty na aplikaci
   - pokud nás zajímá jen část, pak neefektivní
   - **zástupci:** Oracle NoSQL, Dynamo, Redis, Riak
2) **Document**
   - jako klíč hodnota, akorát že hodnota je strukturovaná
     - DB vidí dovnitř, hodnota pochopena, analyzována
   - hodnotou např. XML/JSON – nebo jako objekt
     - vnořování, reference, kolekce
   - dotazy i složitější než přes klíče
     - XPath nebo jako v objektových DB
   - **zástupci:** MongoDB, Elasticsearch
3) **Sloupcové**
   - jako u RDB, akorát že sloupce se mohou pro jednotlivé řádky vzájemně lišit
   - supercolumn –adresář – řádek může obsahovat kolekci supersloupců, kde každý obsahuje kolekci sloupců
   - řídká, více dimenzionální, uspořádaná mapovací funkce
   - **zástupci:** Cassandra (Facebook), BigTable (Google – nepoužívá ani Google)
4) **Grafové**
   - uzly, vlastnosti a hrany
   - různé implementace úložiště – nastavitelné, generické, uživatelovo
   - pro reprezentaci sítí a jejich topologií (sociální, dopravní..)
   - **RDF** specifickou kategorií grafových NoSQL
     - orientovaný ohodnocený graf
     - hrana začíná v **subjektu** , končí v **předmětu** a ohodnocena **predikátem**
     - subjekt a predikát jsou repr. **URI**
     - předmět je hodnota nebo URI odkazující na nějaký předmět

Architektura

- používá se distribuovaná architektura
- např: u MongoDB se zavádí pojmy jako je **PRIMARY** a **SECONDARY** uzel
  - do PRIMARY se zapisuje a do ostatních se replikují data
  - uzly musí být ve stejné replikační množině
  - zajišťuje dostupnost dat při výpadku nějakého uzlu, rychlejší
- **existující projekty a řešení**
  - jednotlivé implementace NoSQL DB systémů se od sebe hodně liší
  - některá architektonická řešení se lze však vysledovat
  - u jednotlivých implementací se potom vyskytují určité rozdíly

Partitioning

1) **Virtuální nódy**

- velký počet serverů lišící se konf.
- z důvodu zohlednění a využití rozdílného výkonu serverů se používají v konf. DBS tzv **virtuální nódy**
- na jednom fyzickém serveru může běžet několik virtuálních nodů, v závislosti na jeho konfiguraci
- používá např **MongoDB, BigTable** či **Cassandra**

2) **Sharding (dělení)**

- metody, která rozděluje velké soubory dat na několik serverů
- **Shard** v pojetí MongoDB je skupina serverů, které udržují identické kopie určené části dat
- používá **MongoDB** nebo **Redis**

3) **Hashing**

- nejjednodušší způsob rozdělení dat spočívá v rozdělení rozsahu primárního klíče
- pomocí hash fce se dá zjistit, na kterém serveru leží data
- například používá systém **Memcached**
- **nevýhoda -** v případě výpadku jednoho ze serverů nebo při přidání serveru, musí být rozdělení dat provedeno znovu a data podle nového rozdělení opět rozdistribuována
  - u **Memcached –** tolik nevadí, pouze malý objem dat v paměti
  - u DB, kde jsou objemná data uložena na HDD by to znamenalo kopírování velkých objemů dat mezi servery

4) **Consistent Hashing**

- používají systémy jako **Cassandra** nebo **MongoDB**
- rozsah klíče na pomyslném kruhu
- každá hodnota má svého správce, kterým je první nod na pomyslné kružnici
- výhodou – změna se dotkne jen sousedních nodů a pouze mezi nimi je nutná redistribuce dat

5) **Replikace**

- používá se k zajištění proti výpadků serveru (při vysokých počtech v infrastr. hor. škál. velmi časté) a k zajištění vysoké dostupnost
- data udržována ve více kopiích
- současně umožňuje **rozkládání** zátěže

11. Big Data – zpracování, Hadoop

Big Data

- data, jejichž velikost je mimo schopnosti zachycovat, spravovat a zpracovávat data běžně používanými SW nástroji v rozumném čase
- obrovská data např v **meteorologii, farmaceutice či telekomunikacích**
- nástup webu, mobilů a dalších technologií zapříčinil zásadní změnu charakteru dat a způsobu jejich využití
  - data **nejsou centralizovaná,** vysoce **strukturovaná** a snadno **zvládnutelná**
  - **jsou** distribuovaná, nemusí být vůbec strukturovaná a mají vzrůstající objem
- často se hovoří **o trojrozměrnosti velikosti** a **růstu** dat – **3V**
  - **objem** – množství dat vznikajících v rámci provozu firem roste exponenciálně každý rok
  - **typ** – různorodost typů dat vzrůstá – například nestr. text soubory, semi-strukt. data (XML), data o geografické poloze, data z logů, …
  - **rychlost** – rychlost s jakou data vznikají a potřeba jejich analýzy v reálném čase vzrůstá díky:
    - **pokračující digitalizaci** většiny transakcí
    - mobilním zařízením
    - vzrůstajícímu počtu internetových uživatelů
- big data mají **odlišné vlastnosti** – odlišují se od tradičních firemních dat

Hadoop

- open source framework pro zpracování, ukládání a analýzu velkého množství nestrukturovaných, distribuovaných dat
- původně vytvořen v Yahoo!
- inspirací **fce MapReduce –** uživatelsky def. funkce, vyvinutá společností **google**
- zvládá **petabyty** a **exabyty** distribuované přes více uzlů současně
- **MapReduce**
  - **výpočetní vrstva** v rámci Hadoopu
  - úlohy této vrstvy přistupují k datům, které jsou distribuována na webu nebo v datových centrech, **rozdělují** je do více replikovaných **dílů** a jejich zpracování pošlou na více jednotlivých uzlů
  - dotazy a další zpracování pak probíhá v každém uzlu paralelně
  - výsledky jsou agregovány a ukládány do úložné vrstvy – **HDFS** (**H**adoop Distributed FS)
  - odtud se pak data načtou do jednoho z několika analytických prostředí pro analýzu
  - Hadoop se skládá z dalších vzájemně se doplňujících projektů – např **Cassandra** či **HBase**
- **hlavní výhoda –** umožňuje analyzovat úplné soubory údajů, včetně nestrukturovaných a částečně strukturovaných dat
  - **z hlediska nákladů –** i času – efektivním způsobem
- **nevýhody**
  - částečná **nezralost** a **hektický vývoj**
  - zavádění a řízení Hadoop clusterů a provádění pokročilé analýzy na velké objemy dat vyžaduje značné odborné znalost
    - vesměs pro firmy nepřijatelný model
    - v rámci Ekosystému Hadoopu existuje řada firem, která staví komerční řešení na bázi Hadoopu, tak aby se začal více používat

12. Prostorové databáze, modelování prostorových dat

Příkladem systém Oracle Spatial

SQL/MM Spatial

- **definuje**
  - ukládání, výběr, dotazování a aktualizaci jednoduchých prostorových objektů
  - reprezentaci prostorových objektů pomocí **prostorových datových typů**
  - **funkce** pro práci s prostorovými objekty – 0,1,2 dimenzionální
- **rozdělen** do několika klauzulí
  - prostorové datové typy a jejich metody
  - datový katalog
- **datové typy**
  - **body:** pouliční lampa, lavička
  - **křivky:** koryto řekni, hranice pobřeží
  - **polygony:** půdorysy budov, území států
  - funguje **dědičnost** – definovaná určitá hierarchie prostorových datových typů
  - **0 dimenzionální objekty**
    - `ST_Point` – X a Y souřadnice vzhledem k nějakému SS (souřadný systém)
    - `ST_MultiPoint` – kolekce bodů
  - **1 dimenzionální objekty** – křivky
    - `ST_LinearString`
    - `ST_CircularString`
    - `ST_CompoundCurce` – kombinace 2 předchozích – složeniny
    - `ST_MultiLineString`
  - **2 dimenzionální objekty** – povrchy
    - `ST_CurvePolygon`, `ST_MultiPolygon`
    - **hranice –**reprezentovaná uzavřenou křivkou (více, když jsou díry)
- vychází z **geometrického modelu OGC** (Open Geospatial Consortium)
  - **rozdíly** – **místo** Line a `LinearRing` zavádí **nový** typ `ST_LineString`
  - **nové** typy pro repr. křivek a povrchů tvořených kruhovými oblouky – `ST_CircularString`
  - u kolekcí **není zřejmé**, z jakých se skládá jednoduchý typů
    - `ST_MultiPOint` je z `ST_Point`??
- prostorový objekt – svázán s nějakým prostorovým systémem souřadnic – **Spatial Reference Systém** – některé resp. i různé zakřivení země
  - `ST_SpatialRefSys` – typ který určí systém souřadnic
  - všechny hodnoty sloupců musí být ve stejným systému souřadném
- pro reprezentaci prázdného prostorového objektu neexistuje samostatný typ – instance každého prostorového typu může být prázdný prost. objekt
  - pokud návratový typ není spec., pak návratovou hodnou bod
- **operace**
  - průnik, sjednocení, sousedi
  - relace – objekt A obsahuje B?
  - vzdálenost
- **4 kategorie metod**
  - **konverze** do/z externích datových formátů
    - **WKT** – well known text (point(10 10), multipolygon((1 1, 2 2, 1 2, 1 1), (2 3, 1 3, 2 4, 2 2))
      - textový značkovací jazyk pro reprezentaci vektorových objektů na mapě
    - **WKB** – well known binary (pro ukládání do DB)
    - **GML** – geography markup lang – xml zápis (od OGC)
  - **práce** s atributy prostorových objektů
    - datové typy mají společné atributy (např. dimenze)
    - podtyp si přidávají některé specifické atributy
    - `ST_X` – souřadnice X, `ST_Length` – délka křivky, `ST_Boundary` - hranice
  - **porovnávání** prost. objektů
    - `ST_Equals` – prost. rovnost, `ST_Touches` – dotýkání, `ST_Within` – obsah

    - **generování** nových prost. objektů – výsledkem operací
        - `ST_Intersection`, `ST_union`, `ST_difference`
        - algoritmu na jeden objekt – `ST_ConvexHull`
- **vlastní typ** – složením z již existujících
- **nedostatky geometrického modelu**
    - pokud chceme například volat něco nad `ST_Point` a `ST_Multipoint`, můžeme použít pouze nějaké obecné funkce, protože se každý typ nachází úplně v jiné větvi
- **views**
    - `ST_Geometry_Columns` – záznamy o všech sloupcích, kt. jsou deklarované jako prostorový datový typ
        - ke každému sloupci může být přidružen SS
        - **záznam** obsahuje – katalog, schéma, tabulku, název sloupce a také SS
    - `ST_Units_Of_Measure` – info o matematických jednotkách
        - **jednotka má –** název, typ, konverzní faktor (vůči základní jednotce daného typu)

13. Distribuované databáze – koncepce distribuované databázového systému, replikace a fragmentace dat, distribuovaná správa transakcí

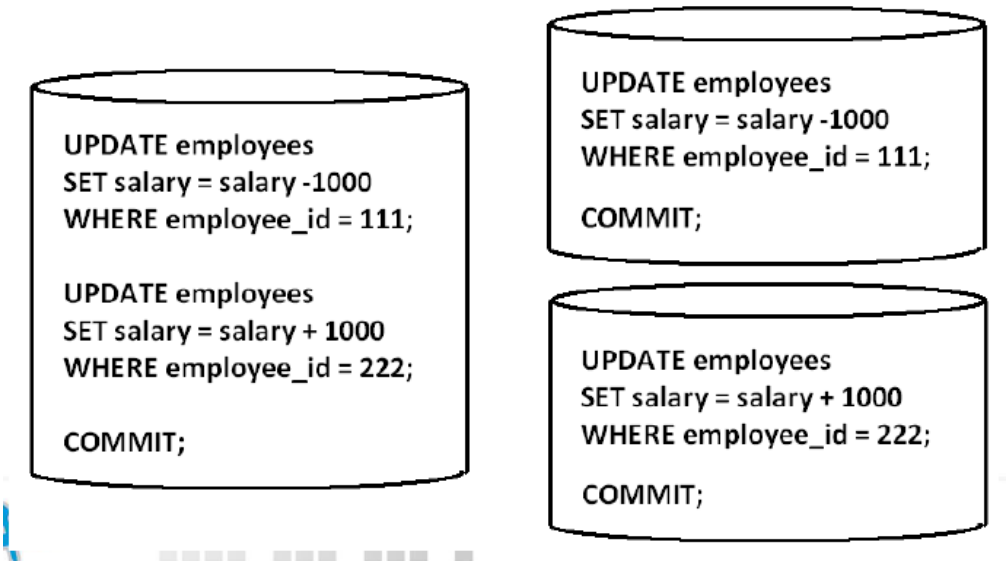
DBMS – Database Management System

= kolekce programů, které umožňují ukládat, modifikovat a získávat data z databáze

- **standard –** Klient – Server , na serveru databáze (databázový server), klient přistupuje k DB protokolem SQL typicky ze své aplikace např.
- **typy databázových systémů**
  - **centrální DBS**
    - jeden databázový server pro celý segment (může být přes WAN)
  - **decentralizovaný DBS**
    - každý lokální segment svůj dat. server
  - **federativní –** dat servery mohou mezi sebou komunikovat po síti
  - **distribuovaný DBS –** kolekce DBMS propojených sítí, dohromady tvořících jeden systém
    - lokální data řízena lokálním DBMS
    - DBMS zajišťuje globální zpracování (práce s daty celé distribuované DB)
    - integrace dat **bez** nežádoucí **centralizace**
    - **návrh**
      - **zdola- nahoru –** integrace existujícího systému
      - **shora – dolů –** green filed, návrh dat modelu, návrh distribuce dat atd.
- **výhody DDBMS**
  - **data blízko**, kde se používají
  - oproti centrálnímu řešení **vyšší výkon**
    - skrytý paralelismus při zpracování SQL
    - práce s lokálními daty je rychlá
  - **vyšší spolehlivost**
  - umožňuje integrovat existující systémy
- **nevýhody DDBMS**
  - složitost
  - bezpečnost – třeba zabezpečit více uzlů, přenos
  - náročná kontrola integrity dat
  - neexistence standardů
  - složitý návrh DB
  - zkušenost
- **klasifikace**
  - **homogenní DDBMS –** v každém uzlu stejný DBMS
  - **heterogenní DDBMS –** v různých uzlech různé DBMS
- **autonomie**
  - **plně autonomní** – izolované, neví o sobě
  - **částečně autonomní –** federativní (lokální data řízena lokálně) , vybraná data možné zpřístupnit pro distribuované zpracování
  - **těsná integrace-** pro uživatele v podstatě jeden DBMS
- **problémy DDBMS**
  - **distribuované SQL**
    - do lokálních tabulek zpracovány lokálním DBMS – jako bez distr.
    - do vzdálených tabulek - musí být rozloženy na dílčí operace v jednotlivých DB (podle umístění dat)
    - nemělo by být nutné přenášet všechna data do jednoho uzlu a tam zpracovávat dotaz
    - možný paralelismus
  - **distr transakce**
  - **podpora distr dat**
    - fragmentace dat
    - replikace dat
  - **zotavení z nových druhů chyb –** výpadek sítě, uzlu, uvíznutí distr. transakce, distr. deadlock

Distribuované transakce

- musí být možné provést transakčně změnu ve více DB najednou



- **Two phase commit (2PC)**
  - protokol pro řešení distribuovaných transakcí
  - řídící uzel - **koordinátor**
  - ostatní uzly poslouchají
  - vlastní commit ve dvou fázích
    - fáze **PREPARE**
      - koordinátor zašle zprávu PREPARE všem uzlům, které se účastní distr. transakce
      - sám koordinátor v této fázi nic nedělá
      - každý uzel se pokusí provést fázi PREPARE
        - uloží transakční log na disk – **neprovádí vlastní commit**
        - transakce stále neukončená, zamčené řádky stále zamčené
        - pouze se zajistí, aby aktuální stav rozpracován transakce byl schopný přežít výpadek
        - **každý uzel –** pošle PREPARED (při úspěchu) nebo ABORT (došlo k problému)
    - fáze **COMMIT**
      - koordinátor po obdržení vyhodnotí výsledky
      - pokud
        - všechny OK – pošle všem zprávu **COMMIT**
        - alespoň jeden ABORTED – všem posílá zprávu **ABORT** , provede RB svých lokálních změn a všem uzlům pošle RB
  - imunní vůči výpadku sítě v libovolné fázi
  - koordinátor si celou dobu udržuje data o stavu transakce a všech pod transakcích
    - po obnovení spojení např se vzdáleným uzlem schopen dokončit spojení

Distribuování dat

- snaha je vždy přiblížit data co nejblíže k uživateli – **vyšší rychlost, nižší zatížení sítě**
- hlavně se uplatňuje při shora-dolů – je třeba znát, jaká data se budou v jakých uzlech používat
- základní **jednotkou alokace** je **tabulka**
- další techniky
  - **fragmentace**
    - vhodné, pokud se v každém uzlu systému pracuje s částí tabulky
    - rozdělení relace na několik subrelací, které se umístí do jednotlivých DB
    - data se umístí tam, kde se nejčastěji používají – ostatní v ostatních DB
      - omezení síťových přenosů
      - vyšší bezpečnost – data pouze tam, kde jsou třeba
    - **musí být –** úplná, disjunktní, rekonstruovatelná
    - **typy**
      - **horizontální –** podmnožina řádků
      - **vertikální –** podmnožina sloupců
      - **dokonce smíšená –** kombinací
  - **replikace**
    - udržování kopií tabulek nebo fragmentů ve více DB
    - **omezení** síťových **přenosů**
    - **zvýšení dostupnosti** dat při výpadku části DS
    - náročná údržba kopií
    - vhodné pro **často používaná** data, která se **málo aktualizují**
    - **typy**
      - **synchronní –** aktualizace na všech uzlech, součástí transakce
        - degraduje výkon X spolehlivý
      - **asynchronní –** na vyžádání nebo v pravidelných intervalech, nemusí být vždy aktuální
    - **typy podle způsobu provádění repliky**
      - pouze změněné řádky – nutný log změněných řádků
      - úplná
      - DML příkazy...
    - **typy podle způsobu zacházení s replikami**
      - **pouze pro čtení (master-slave)**
        - změny možné provádět jen u mastera
        - repliky jsou RO
      - **pro transakční zpracování**
        - změny možné provádět i v replikách – musí se synchronizovat obousměrně
        - přináší řadu problémů – konflikty při změněn

14. Temporální databáze, porovnání klasických a temporálních DB, modely času, vztah událostí a času, temporální SQL

Motivace

- potřebujeme v DB čas?
- př: studijní informační systém, skladová evidence, účetní a bankovní systémy
- **klasický DBS**
  - zachycen stav systému v aktuálním časovém okamžiku
  - problém: co dělat se starými daty
- **temporální DBS**
  - DB určitým způsobem podporující čas
  - jednodušší dotazy
  - jednodušší udržování aplikací
- **temporální projekce**
  - jako projekce v klasických DB
  - navíc se bere v úvahu čas – **srůstání** časů
  - nedostatečná podpora v klasických systémech
  - př: např máme zaměstnance na různých pozicích, ale plat mu při dvou pozicích zůstal stejný -> umožňuje udělat platovou historii, doba, kdy měl stejný plat, jenom na jiných pozicích **sroste** dohromady
- **temporální spojení**
  - stejné spojení jako u join v klasických DB
  - navíc bere v potaz čas události

Modely času

- **temporální logika –** čas je libovolná množina okamžiků s daným uspořádáním
- **modely času podle uspořádání**
  - lineární
  - větvený (čas možných budoucností)
  - cyklický
- **modely času podle hustoty**
  - diskrétní
  - hustý
  - spojitý
- **omezenost času**
- **absolutní X relativní čas**

- **datové typy**
  - časový okamžik (instant)– DATE / TIME / TIMESTAMP
  - časový úsek (time period)
    - doba mezi dvěma časovými okamžiky
    - 15:30 – 15:50
  - časový interval (interval)
    - doba o specifikované délce, ale bez krajních bodů
    - 30min
  - množina časových okamžiků (instant set)
  - množina časových úseků (temporal elements)

Vztah událostí a času

- **čas platnosti (valid time - VT)**
  - čas, kdy byla událost pravdivá v reálném světě
  - může být v minulosti, přítomnosti i budoucnosti
- **transakční čas (transcation time - TT)**
  - čas, kdy byl fakt reprezentován v databázi
  - nabývá **pouze** aktuální hodnoty
  - monotónně roste
- čas platnosti a transakční čas jsou ortogonální
- **typy datových modelů**
  - **snapshot –** nepodporuje ani VT ani TT
    - klasický relační model
    - každá n-tice je fakt platný v reálném světě
    - při změně reálného světa se mění prvek modelu
  - **valid-time –** podporuje pouze VT
    - cokoli v relaci může být upraveno
    - čas události (začátek i konec)
    - umožňuje klást dotazy platné v minulosti i budoucnosti
    - **Segeův datový model** – razítko, kdy fakt začal platit
    - **Sardův datový model –** razítko má podobu intervalu
    - **HRDM –** razítkují se hodnoty atributů
  - **transaction-time –** podporuje pouze TT
    - posloupnost snapshotů indexovaná transakční časem
    - umožňuje získat informaci ze stavu DB v nějakém časovém okamžiku z minulosti
    - možnost větvení
  - **bitemporální –** podporuje TT i VT
    - append only
  - **temporální –** podporuje TT nebo VT
  - **obvykle** založeno na relačním datovém modelu objektově orientovaného datového modelu

Temporální SQL (TODO, trochu guláš)

- **používá temporální datový model**
  - objekty s přesně danou strukturou
  - omezení pro dané objekty
  - operace na daných objektech
  - **ideální**
    - snadná implementace
    - vysoký výkon
    - souvislá prezentace chování měnícího se v čase
    - minimální rozšíření exist. DM
  - **hlavním cílem –** zachytit sémantiku dat měnících se v čase
- **nadmnožina SQL 92**
- měl **sjednotit** přístup k temporálním datovým modelům
- **časová osa** TSQL2 je obou koncích **omezena -**18 miliard let
- u časových údajů možné různé **granularity –** 1 rok, den, hodina...
- časové typy **– DATE, TIME, TIMESTAMP, INTERVAL, PERIOD**
- použit BCDM **bitemporal conceptual data model**
- řádek relace je **orazítkován** množinou bitemporálních **chrononů** (dvojice chrnono TT a VT)
- **INSERT –** VALID PERIOD – lze specifikovat VT pro daný prvek
- **SELECT SNAPSHOT –** bere aktuální stav, podle toho v jakým časovým intervalu se právě nacházíme
  - bez něj to vypíše celou historii

15. Možnosti tvorby datových skladů a metody dolování znalostí

Data Warehouse (DW)

- **problém s OLTP** (Online Transaction Processing)
  - nedosažitelnost dat vytvořených či skrytých v transakčních systémech
  - dlouhé prodlevy, když se nedostatečně silné systémy snaží provést komplikované dotazy
- **skladování dat –** kolekce technologií podporujících rozhodování, s cílem umožnit řídícímu pracovníku učinit lepší a rychlejší rozhodování
- **datový sklad –** označuje DB architekturu používanou pro údržbu historických dat, která jsou získána z jedné nebo více operativních DB
  - tato data jsou typicky vyčištěna a restrukturována pro podporu dotazů, agregací a analýz
  - **klíčové –** integrace vlastních a externích dat
- **cíle**
  - měl by poskytovat nikoliv operativní data, ale data přeměněná ve strategické informace
- **DW v informačním prostředí organizací**
  - **výroba dat** – selektivní dotazy nad OLTP
  - **skladování dat** - DW, datová tržiště, dotazy intenzivní na data
  - **prodej dat – OLAP –** Online Analytical processing
  - **klíčové –** multidimenzionalita

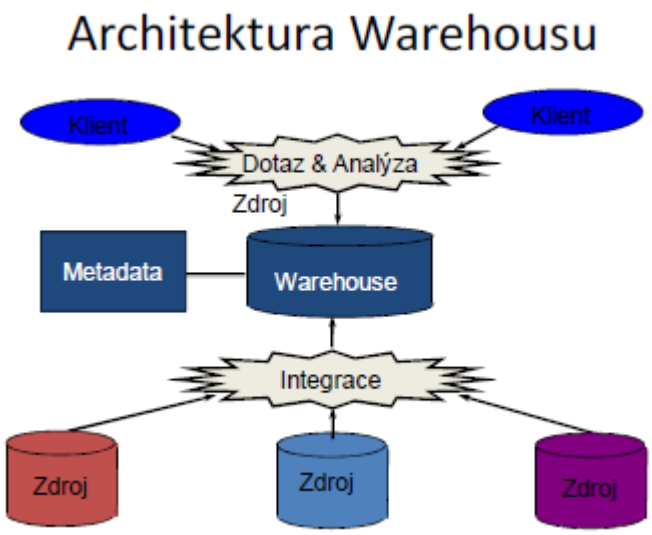
Modelování DW

- **OLTP databáze** - normalizované tabulky, optimalizace pro insert, update
- **OLAP databáze –** odvozené tabulky, redundantní data, **optimalizace pro dotazy**, procesní logika ve schématech
  - většinou se z ní čte, dlouhé komplexní dotazy, GB-TB dat,. sumarizovaná a konsolidovaná data, jako uživatel nějaký vedoucí pracovník, analytik
  - **soubor operací** drill-down, roll-up – různé pohledy na data

- **přístupy k modelování**
  - konceptuální struktury **založené na tabulkách** (dimenzionální a tabulky faktů)
    - organizované do tzv hvězdicových schémat (cizí klíč odkaz na tabulku)
  - konceptuální struktury **založené na hyperkostkách** (kostkách, multidimenzionálních polích)
    - reprezentují data jako multidimenzionální strukturu
- **hyper kostka (hyperqube)**
  - najde se těžiště systému a jako jednotlivé dimenze se vezmou cizí klíče
  - schéma MD databáze je množina vícerozměrných polí
  - MD DB je dána vícerozměrnými množinami dat uloženými v těchto polích
  - **výhody**
    - pole nabízí přímo jisté informace – např: počet pozic v každé dimenzi
    - jednodušší vyhledávání
    - v poli se přirozeně seskupují data (řezy kostkou)
    - seskupování dat => agregace
  - **objem –** množina dat
  - nejvýhodnější pokud existují multizávislosti, efektivnější rozložení do kostky
  - **agregáty** – jenom se agreguje podle nějaký vlastnosti (odebere se dimenze) a vrátí se výsledek

- **pomocí tabulek a vztahů**

- jako standardní schéma
- hierarchie dimenzí – zakresluji se hrany tam, kde je cizi klic, jako uzly nazvy tabulek



Data Mining

- Neustále generujeme a ukládáme stále více dat,
- DB technologie jsou rychlejší, levnější, data rozsáhlejší,
- ale vyvodit užitečné závěry je stále složitější → **smysl**: dát uloženým datům význam

Definice: Netriviální proces identifikace nových, platných, potenciálně použitelných a snadno pochopitelných vzorů v datech.

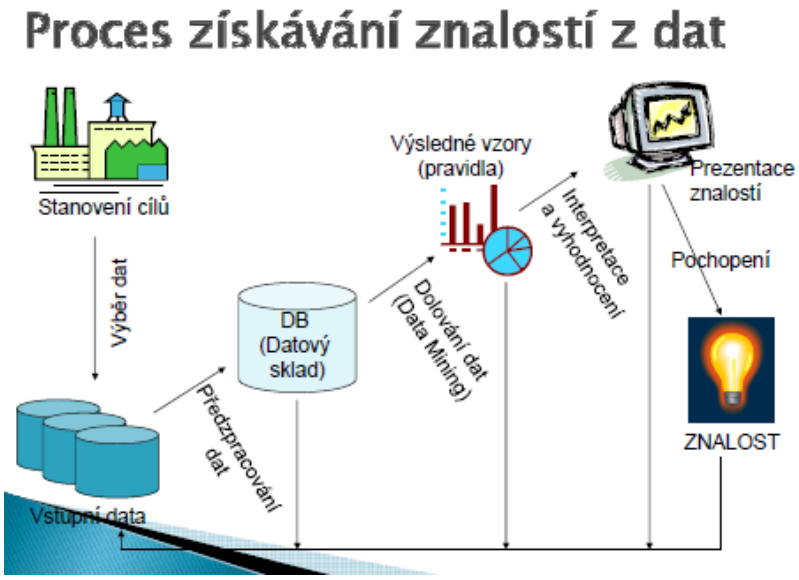
- Poznatky z několika oborů matematiky a informatiky – umělá inteligence, vizualizace, databázové systémy, statistika

OLAP vs. Data Mining

- **Data Mining**
  - Hledání nových vzorů, znalostí, které nejsou explicitně uvedeny
  - Dosahováno pomocí sofistikovaných algoritmů
- **OLAP**
  - **Soubor operací** (drill-down, roll-up, …) – poskytují různé pohledy na data
  - Dosahováno pomocí sumačních a předdefinovaných operací

Vlastnost	OLAP	Data Mining
Motivace použití	Co se děje v podniku?	Predikce budoucnosti, skryté znalosti
Granularita dat	Sumační data	Data na úrovni záznamu
Počet obchodních dimenzí	Omezený	Velký (až nekonečný)
Počet vstupních atributů	Spíše velmi nízký	Mnoho atributů
Velikost dat pro jednu dimenzi	Ne velká pro každou dimenzi	Obvykle velmi rozsáhlá pro každou dimenzi
Přístup k analýze	Řízený uživatelem – interaktivní analýza	Řízený daty - automatický
Techniky analýzy	Multidimenzionální, drill-down, slice-and-dice	Příprava dat, použití nástrojů pro získávání znalostí
Stav technologie	Známý a rozsáhle využívaný	Stále se vyvíjející, některé metody již využívány v praxi

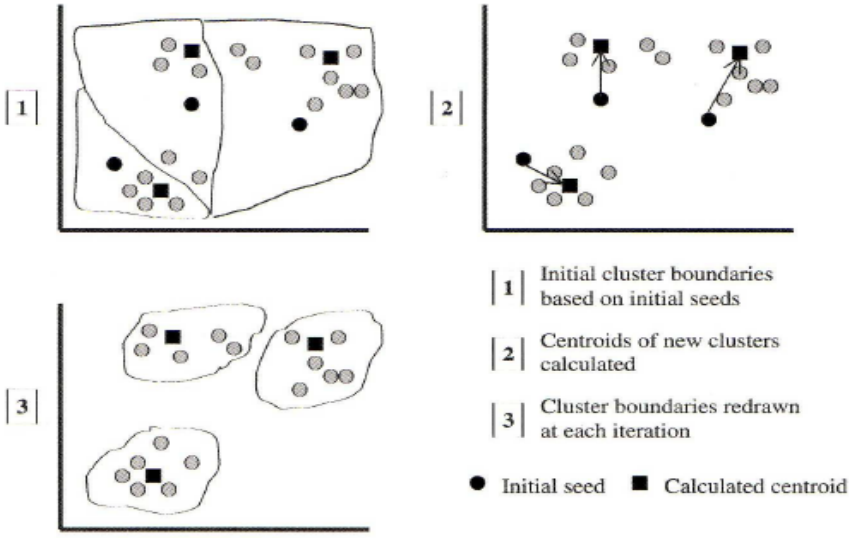
Proces získávání znalostí



- Stanovení cílů
  - Co chceme nalézt, v čem, bude to k něčemu, je problém řešitelný?
  - Jak zobrazíme výsledky, jsou data pro metodu vhodná?
- Výběr zdrojů dat
  - Typy dat z hlediska zaměření
    - Demografická
    - Behaviorální
    - Psychografická
  - Typy databází z hlediska obsahu
    - Zákaznické
    - Transakční
    - Databáze historie nabídek
    - Datový sklad
  - Typy dat z hlediska formátu
    - Relační a transakční databáze
    - OO databáze
    - Multimediální databáze
    - WWW, textové dokumenty, prostorová, časová data, …
  - Externí data
- **Předzpracování dat** – vybrat z objemné databáze relativní data
  - Čistění dat
    - položky obsahující neúplné nebo chybné hodnoty, nekonzistentní data
    - řešení: ruční opravení, nebo opravné rutiny
  - Integrace dat
    - více zdrojů do jedné databáze -> redundance, určení ekvivalentních entit?
    - Detekce a řešení konfliktů hodnot atributů (různé kódování, jednotky, …)
  - Transformace dat
    - Do formátu vhodného pro dolování dat
      - Slučující techniky – sumační operace (více hodnot -> jedna hodnota)
      - Generalizace – nižší úroveň nahrazena vyšší (ulice -> město)
      - Normalizace – přepočítání hodnot do daného intervalu
    - Přidávání nových (odvozených) atributů
    - Diskretizace hodnot numerických atributů
      - Rozdělení numerických hodnot na intervaly, ekvidistantní, do hloubky, pokročilé metody
  - Redukce dat
    - Agregace v kostce – redukce sumačními operacemi
    - Redukce rozměrů – detekce a odstranění nadbytečných a nepoužívaných atributů
    - Komprese dat – zmenšení objemu dat
    - Numerosity – data nahrazena alternativní menší reprezentací
- **Dolování dat** – aplikace zvoleného algoritmu na předzpracovaná data, dle typu znalosti a dat
  - Typy znalostí:
    - **Asociační pravidla A => B**
      - *„Jestliže transakce obsahuje položky z množiny A, pak také pravděpodobně obsahuje položky z B“*
      - **Dva ukazatele:**
        - Podpora (support) – ppst, že se vyskytují v db položky z A i B
        - Spolehlivost (confidence) – podmíněná ppst, že se v transakci vyskytuje B za předpokladu, že se tam vyskytuje A
      - Silné asociační pravidlo – má podporu a spolehlivost vyšší než uživatelem zadaní hodnota – málo silných asociačních pravidel -> víceúrovňová asociační pravidla (položky sdruženy do skupin)
      - Frekventovaná množina – množina, která má podporu vyšší než minimální hodnota

- Postup: výpočet frekventovaných množin (alg. Apriori), generování asociačních pravidel z frekventovaných množin
- Asociační pravidla v relačních databázích
  - kategorické atributy
    - konečný počet hodnot, lze použít Apriori
  - kvantitativní atributy
    - nemají konečný počet hodnot, nutnost diskretizace

- **Shlukování**
  - Nejstarší nástroje
  - Roztřídění objektů do skupin (shluků), které nejsou předem stanoveny a nemají tedy význam (ten je potřeba zjistit)
  - Rozdíly uvnitř shluků minimální, rozdíly jednotlivým shluků maximální
  - Problém: Jakou metriku použít?
  - Metody:
    - **Rozdělovací**
      - Rozdělení na předem daný počet shluků
      - Např. alg. K-means – optimalizuje těžiště shluků a dané prvky pak přiřadí nejbližšímu těžišti



    - **Hierarchické**
      - Postupné rozdělování velkých shluků nebo postupné slučování malých shluků -> hierarchická struktura
      - Ukončení procesu při splnění určité podmínky (např. minimální počet shluků)
    - Další metody – neuronové sítě, mřížky, apod.
  - **Příklady aplikací**: marketing, plánování města, studie zemětřesení, pojištění, geografie
- **Klasifikace**
  - Rozdělování do předem známých skupin
  - Úspěšnost se měří procentem úspěšně klasifikovaných objektů
  - **Klasifikátory** (modely)
    - Nejčastěji rozhodovací stromy
      1. Konstrukce stromu na základě vzorku dat
      2. Klasifikace objektů na základě stromu
         - Vnitřní uzel – test hodnoty jistého atributu
         - Koncový uzel – třída, kam je klasifikován
      - Pravidla – IF … THEN …, lze převést na rozhodovací strom
      - Neuronové sítě
- Predikce

- **Využití data miningu**
  - Členění (segmentace) zákazníků – porozumět zákazníkovi a jeho chování
  - Analýza nákupního košíku – nalezení závislostí mezi různým zbožím
  - Management rizik – odhalení rizikových zákazníků (např. u pojišťoven)
  - Detekce podvodů – např. hledání extrémních útrat na kreditce
  - Odhalování zločinnosti – odhalení potenciálních neplatičů půjček
  - Predikce požadavků – předpověď zájmu zákazníků o různé zboží

- Významní dodavatelé: SAS, IBM, SPSS, Silicon Graphics, Angnoss Software, StatSoft
- **Dotazovací jazyky** pro data mining – součásti dotazu
  - **Relevantní data** – jméno DB/DB skladu, DB tabulky/kostky, podmínky pro selekci dat, kritéria pro seskupování, …
  - **Typ znalosti** – asociační pravidla, shlukování, klasifikace, …
  - **Doménová znalost** – typické využití: konceptuální hierarchie (stromová, seskupovací, odvozená z operace, …)
  - Metriky zajímavosti
    - Jednoduchost – počet prvků pravidla, velikost rozhodovacího stromu
    - Použitelnost – např. podpora a spolehlivost
    - Jedinečnost – odstranění podobných znalostí
  - Vizualizace/prezentace získaných znalostí
    - Různé formy – grafy, tabulky, konceptuální hierarchie