

DBS – Databázové modely

Michal Valenta

Katedra softwarového inženýrství FIT
České vysoké učení technické v Praze
Michal.Valenta@fit.cvut.cz
©Michal Valenta, 2010

BIVŠ DBS I, ZS 2010/11

<https://users.fit.cvut.cz/valenta/>
(odkaz “**Výuka na BIVŠ**”)



- 3 úrovně pohledu na data
- integritní omezení
- konceptuální a logický model ? souvislost s SI
- konceptuální modelování (ER, UML Class Diagram)
- logické (databázové) modely (popis, příklad)
 - síťový
 - hierarchický
 - relační
 - objektový
 - XML

Různé úrovně pohledu na data

- Konceptuální
 - Zabývá se modelováním reality.
 - Snaží se nebýt ovlivněna budoucími prostředky řešení. Používá se grafická notace (obvykle ER model nebo UML Class Diagram), případně další IO.
- Logická (databázová)
 - Vztahuje se ke konkrétnímu databázovému modelu a používá jeho konstrukční dotazovací a manipulační prostředky (relační objektová, síťová, hierarchická, XML, ...).
- Fyzická
 - Jde o fyzické uložení dat (sekvenční soubor, indexy, cluster, ...). Uživatelé (programátoři aplikací, příležitostní uživatelé) je od ní odstíněn logickou vrstvou SRBD.

- IO jsou tvrzení vymezující korektnost DB.
- Definují se na konceptuální úrovni, promítají se do úrovně logické.
- Na databázové (logické) úrovni se definují pomocí JDD (DDL). Někdy DDL není dost silný, proto další prostředky (triggery, uložené procedury), nebo až na úrovni aplikace.
- DDL prostředky datových modelů jsou různě silné (relační objektová, síťová, hierarchická, XML, ...)
- Příklady IO (Uvažujeme půjčovnu filmů, která půjčuje filmy do kin.):
 - Kino je jednoznačně určené názvem.
 - Film si lze v půjčovně rezervovat jen tehdy, jsou-li všechny jeho kopie vypůjčeny
 - Zákazník si může vypůjčit nejvýše 6 filmů (kopií)
 - Vypůjčující si osoba musí být v seznamu zákazníků půjčovny.

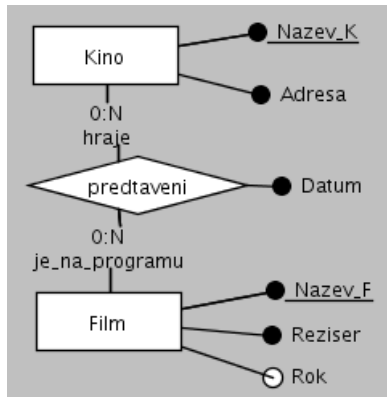
- V polovině 70. let 20. století.
- Nejdůležitější přínosy:
 - společné chápání objektů aplikace uživateli a projektanty,
 - integrace různých uživatelských pohledů,
 - výsledek je vstupem pro realizaci databáze,
 - slouží jako dokumentace.
- Důsledky vypuštění konceptuální úrovně :
 - Příliš nízká úroveň pohledu na data:
 - ⇒ obtížná komunikace se zadavatelem (zákazníkem),
 - ⇒ neumožní realizaci větší databáze.
 - V rozsáhlejší databázi je velmi těžké se zorientovat.

Konceptuální modelování

- E-R diagramy (1976)
 - de facto standart (pro relační databáze)
 - chenova notace
 - binární ER (Oracle)
 - mnoho dalších ...

Konstrukty ER:

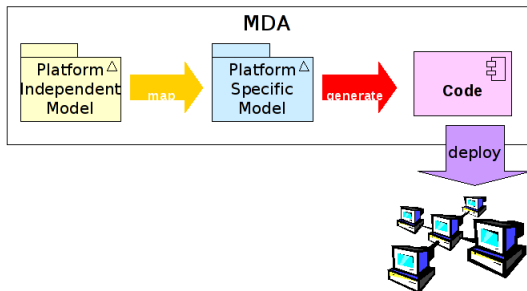
- entitní množiny (entity)
 - atributy entit
- vztahové typy (vztahy)
 - účast ve vztahu
 - atributy vztahů
- integritní omezení
 - identifikátory
 - násobnost účasti (kardinalita a parcialita vztahu)



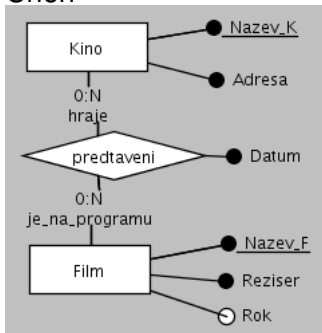
- součást UML
 - ⇒ implementace v mnoha návrhových nástrojích,
 - ⇒ včetně automatických generátorů databázového modelu (DDL),
 - ⇒ Enterprise Architect, UML Star, Rational Rose,
- UML (a Class Diagram) je objektově orientovaná notace
 - ⇒ výhodné pro objektově orientovanou implementaci
 - ⇒ většina (současných) databází je však relační (resp. OR)
 - ⇒ pro návrh databáze se často nevyžívají všechny výrazové prostředky (jednodušší Class Diagram má přímočarý překlad do zvoleného JDD na databázové úrovni, převod složitějších konstrukcí nemusí být jednoznačný nebo úplný a může působit problémy, viz například ISA hierarchie v dalších přednáškách).

UML future?

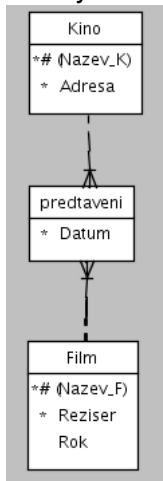
- The future of development of UML will be increasingly affected by Model Driven Architecture (MDA)



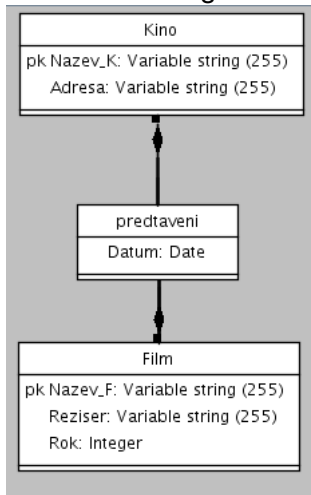
Chen



Binary



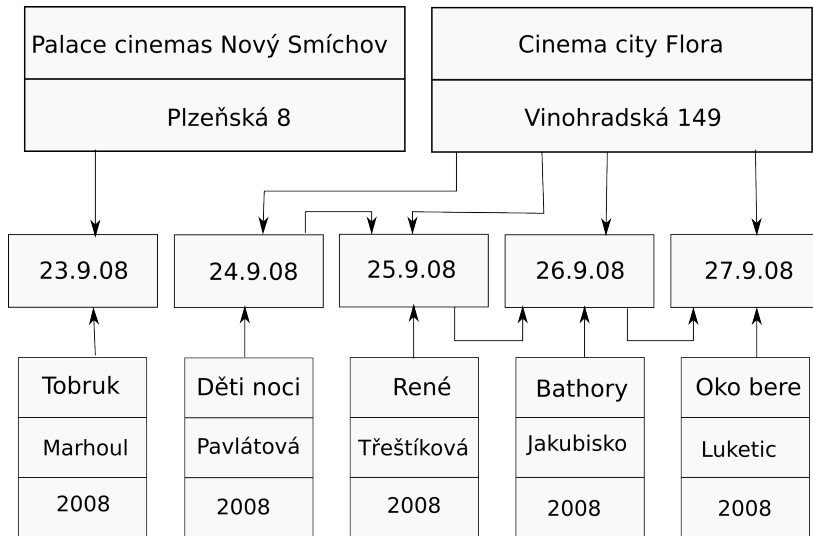
UML Class Diagram



- **síťový** - 60. léta 20. století
- **hierarchický** - konec 60.let; lze chápat jako specializaci síťového modelu
- **relační** – 70.léta
- **objektový** - 80.léta; lze chápat jako rozšíření síťového modelu
- **objektově-relační** - 90-léta; komerčně úspěšný kříženec relačního a objektového modelu; podpora ve standardech SQL (SQL99, SQL2003)
- **XML** - konec 90 let, mnoho prvků hierarchického modelu; aplikační doména?; zpracování XML dat také proniká do standardu SQL

- Volba databázového modelu určuje prostředky pro vytváření struktury databáze (DDL) a prostředky pro tvorbu aplikací (DML, dotazovací jazyk, TCL, DCL)
- Příklady:
 - relační model - SQL
 - Objektový model - OQL
 - XML model - Xpath, XQuery

Síťový model – příklad – schéma výskytů



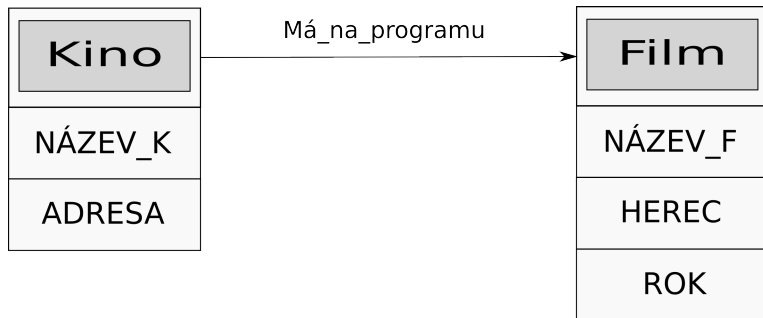
- datový typ **Record** (záznam), který se podobá pascalskému datovému typu *File of record*
- datový typ **Set** (C-množina); dvojice různých datových typů **Record**, který se podobá datovému typu *Seznam*)

Poznámka:

- Snadná konverze mezi ER a síťovým modelem:
 - Každému entitnímu typu odpovídá jeden typ **Record**
 - Každému vztahovému typu 1:N odpovídá jeden typ **Set**

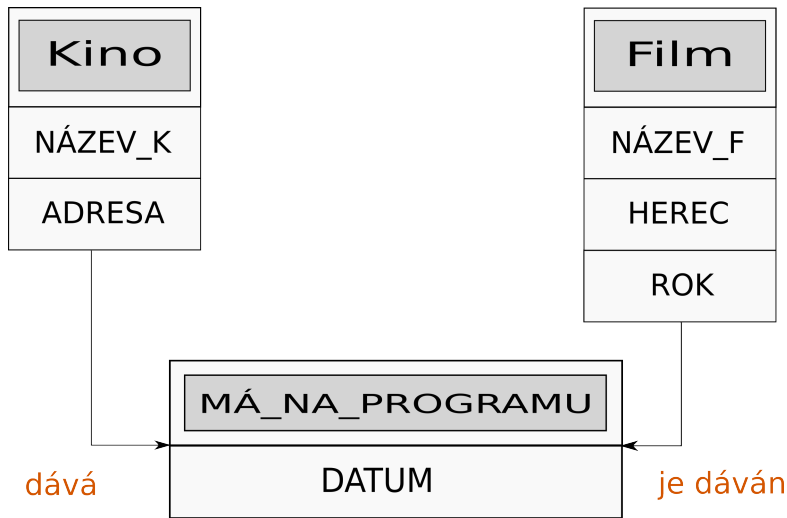
- vytvoř **nový** záznam daného typu, **zruš** záznam, **změň záznam**
- **zařad'** členský záznam do c-množiny daného **vlastníka**
- **vyřad'** daný **člen** z c-množiny
- najdi první **člen** v c-množině daného vlastníka
- najdi **následovníka** v c-množině daného vlastníka
- najdi **vlastníka** daného člena c-množiny

Síťový model – Bachmanův diagram – návrh 1



typ Záznam typ SET (C-množina) typ Záznam

Síťový model – Bachmanův diagram – návrh 2



Blíže realitě než předchozí návrh.

Dotaz: Vypiš program kina Blaník.

Begin

```
Najdi KINO záznam (NAZEV='Blaník');
```

```
Get KINO;
```

```
Najdi prvního člena v MÁ_NA_PROGRAMU;
```

```
While Not EOF MÁ_NA_PROGRAMU Do
```

```
    Get MÁ_NA_PROGRAMU into A;
```

```
    Print (A.Datum);
```

```
    Najdi vlastníka k A ve FILM;
```

```
    Get FILM into B;
```

```
    Print (B.Nazev);
```

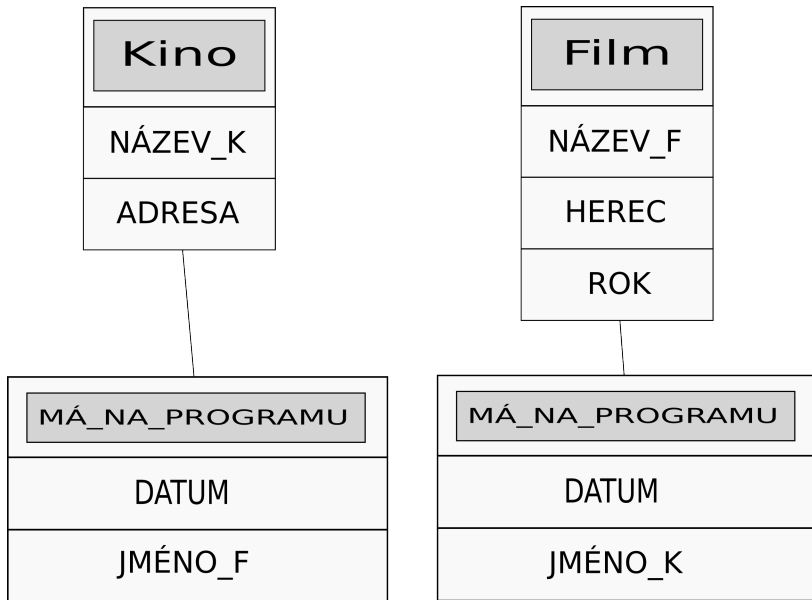
```
    Najdi následovníka v MA_NA_PROGRAMU;
```

```
End;
```

```
End;
```

- specializace modelu síťového
- síťový = orientovaný graf, hierarchický = strom
- omezené použití (nevhodné pro náš příklad!)
- vhodné pro modelování typu část/celek
- aplikace – evidence součástek v projektu Apolo

Hierarchický model – příklad



- Jediný konstrukt – **relace**
 - schéma relace: jméno relace, jména atributů, specifikace domén atributů
 - prvky domén jsou atomické hodnoty (1.normální forma)
 - formální zápis $R(A_1:D_1, \dots, A_n:D_n)$
 - příklad: `KINO (NAZEV_K:CHAR(15), ADRESA:CHAR(25))`
- Integritní omezení: primární klíč, cizí klíč

KINO(NAZEV_K, ADRESA)

FILM(JMENO_F, HEREC, ROK)

MA_NA_PROGRAMU(NAZEV_K, JMENO_F, DATUM)

Integritní omezení:

- primární klíče:
 - NAZEV_K
 - JMENO_F
 - {NAZEV_K, JMENO_F}
- cizí klíče
 - MA_NA_PROGRAMU.NAZEV_K
 - MA_NA_PROGRAMU.JMENO_F

IO relace MA_NA_PROGRAMU jsou příliš silná. Proč?

Nelze aby jedno kino hrálo jeden film víckrát (v jiný den a/nebo čas).

KINO

NÁZEV_K	ADRESA
Blaník	Václ. n. 4
Vesna	Olšiny 3
Mír	Strašnická 3
Domovina	V dvorcích 7

MA_NA_PROGRAMU

NÁZEV_K	JMENO_F	DATUM
Blaník	Top Gun	29.3. 1994
Blaník	Kmotr	8.3. 1994
Mír	Nováček	10.3. 1994
Mír	Top gun	9.3. 1994
Mír	Kmotr	8.3. 1994

FILM

JMENO_F	HEREC	ROK
Černí baroni	Vetchý	1994
Černí baroni	Landovský	1994
Top gun	Cruise	1986
Top gun	McGillis	1986
Kmotr	Brando	1972
Nováček	Brando	1990
Vzorec	Brando	1980

- **vytvoř novou relaci** (tabulku)
- **přidej novou n-tici** (řádek) do dané relace (tabulky)
- **vymaž n-tice** (řádky) zadaných vlastností
- ve vybraných **n-ticích** (řádcích) zadané relace (tabulky) **změň hodnoty** zadaných prvků (polí)
- vytvoř **novou relaci** (tabulku) ze zadané relace:
 - výběrem n-tic (řádků) zadaných vlastností – **selekce**
 - výběrem zadaných atributů (sloupců) – **projekce**
- vytvoř **novou relaci** (tabulku) ze zadaných relací (tabulek) pomocí **množinových operací sjednocení, průnik, rozdíl, kartézský součin**
- vytvoř **novou relaci** (tabulku) ze zadaných relací pomocí operace **spojení**

Dotaz: Vypiš program kina Blaník.

- relační algebra

```
(KINO (NAZEV_K = 'Blaník') *  
MA_NA_PROGRAMU * FILM) [jmeno_f, datum]
```

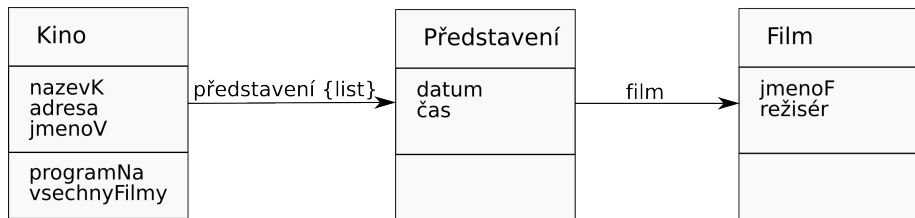
- SQL

```
Select Jmeno_F, Datum  
From KINO K JOIN MA_NA_PROGRAMU MNP  
ON (K.NAZEV_K= 'Blaník'  
and K.NAZEV_K= MNP.NAZEV_K)  
JOIN FILM USING (Jmeno_F);
```

- **Objekty** = data + metody. Mezi objekty existuje skládání, dědění, závislost, klasifikace podle tříd, ... Strukturované informace není třeba rozdělovat jako v RDM.
- **Protokol** objektu je dán množinou **přístupných zpráv** (ne atributů jako v RMD).
- Jedna množina (objektů) může s využitím **polymorfismu** obsahovat objekty s různou strukturou dat i metod.
- Je rozdíl mezi **množinou** objektů a **třídou**.
- **Identita objektu** je dána nejen vnitřními, ale i vnějšími vazbami. Klíče jsou interní záležitostí.

- základní konstrukt – **objekt**
 - generován jako instance dané třídy (která nese informace o jménech atributů, specifikaci domén atributů, názvech metod, ...)
 - má stav (hodnoty atributů)
- množinové konstrukce - **kolekce**:
 - set, bag, list, array, dictionary, ...
- množinové operace
 - so:, select:, collect:, detect:, inject:, reject:, intersect:, union:, ...

Objektový model – příklad



Metody objektu Kino:

```
programNa: datum
```

```
  ^predstaveni select: [:p | p datum = datum]
```

```
vsechnyFilmy
```

```
  ^(predstaveni collect: [:p | p film]) asSet
```

- Podobá se hierarchickému – XML dokument je obvykle chápán jako strom; DOM API pro přístup.
- Aplikační doména?
- Datový model : elementy, atributy, PCDATA, zachovává pořadí (document order). Někdy je bohatší.
- Silné a standardizované dotazovací jazyky (XPath,XQuery)
- Mnoho implementací a mnoho věcí stále ve vývoji (indexování, zamykání, ...)

XML model – příklad schématu – DTD

```
<!ELEMENT program (kino*)>
<!ELEMENT kino (nazev_k,  adresa,  hraje*)>
<!ELEMENT hraje (film,  datum)>
<!ELEMENT film (nazev_f,  herec,  rok)>
<!ELEMENT nazev_k (#PCDATA)>
<!ELEMENT adresa (#PCDATA)>
<!ELEMENT datum (#PCDATA)>
<!ELEMENT nazev_f (#PCDATA)>
<!ELEMENT herec (#PCDATA)>
<!ELEMENT rok (#PCDATA)>
```

Poznámka

Toto schéma bude jistě obsahovat množství opakujících se hodnot. Zřejmě by bylo nevhodné i pro DML operace (aktualizační anomálie). Naopak by bylo vhodné pro přímé vygenerování reportu (html, pdf, ...) s programem jednotlivých kin.

XML model – příklad – data

```
<program>
  <kino>
    <nazev_k> MAT </nazev_k>
    <adresa> Karlovo nám. 18, Praha 2 </adresa>
    <hraje>
      <film>
        <nazev_f> Forest Gump </nazev_f>
        <herec> Tom Hanks </herec> <rok> 1998 </rok>
      </film>
      <datum> 3.1. 2007 </datum>
      <film>
        <nazev_f> Vratné láhve </nazev_f>
        <herec> Zdeněk Svěrák </herec> <rok> 2006 </rok>
      </film>
      <datum> 17. 5. 2007 </datum>
    </hraje>
  </kino>
  <kino> ... </kino>
  ...
</program>
```

- **Názvy kin v databázi :**
 - `/program/kino/nazev_k`
- **Všichni herci:**
 - `//herec`
- **Kina, která mají na programu aspoň 2 filmy:**
 - `//kino[count(./hraje/film)>2]/nazev_k`
- **Filmy, které hrají v kině Blaník:**
 - `//kino[nazev_k="Blanik"]//nazev_f`

Dotaz: Názvy filmů se seznamem kin, kde se hrají

```
let $kina :=  
    "file:///home/valenta/vyuka/dbs/2007/kina.xml"  
return element obraceny_vypis {  
    for $film in distinct (doc ($kina)//nazev_f)  
    return element film {$film/text(),  
        element se_hraje_v {doc ($kina)//kino  
            [hraje/film/nazev_f = $film/text()]/nazev_k}  
        }  
    }  
}
```

Jedná se vlastně o inverzní výpis databáze.

V tomto předmětu se budeme dále věnovat **relačnímu databázovému modelu**.

- Je však dobré si uvědomit, že:
 - Relační model není jediný, ze kterého si můžeme vybírat.
 - Pro určitý typ aplikace nebo aplikační doménu můžeme výběrem vhodného DB modelu mnoho ušetřit.
 - Volbu DB modelu je třeba dobře uvážit a zdůvodnit.