Databázové modely

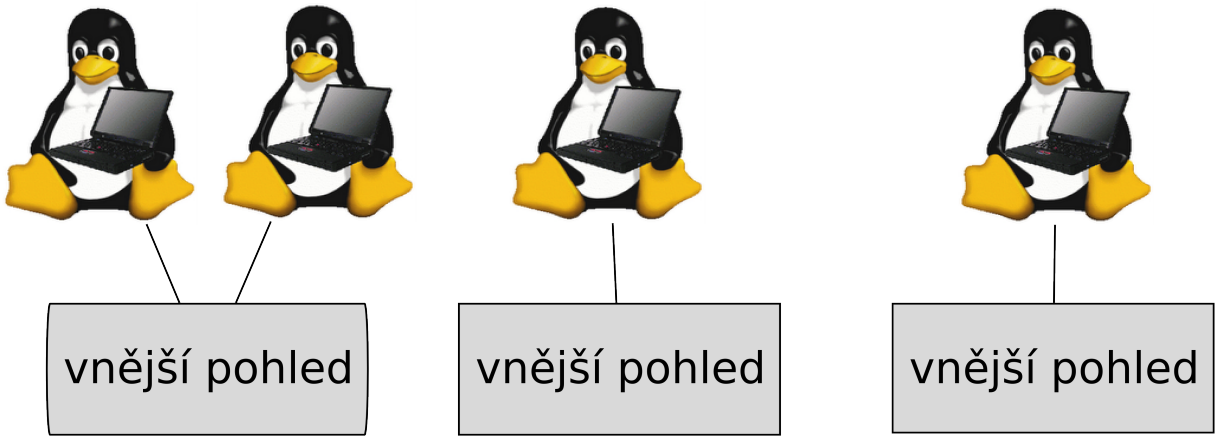
Michal Valenta

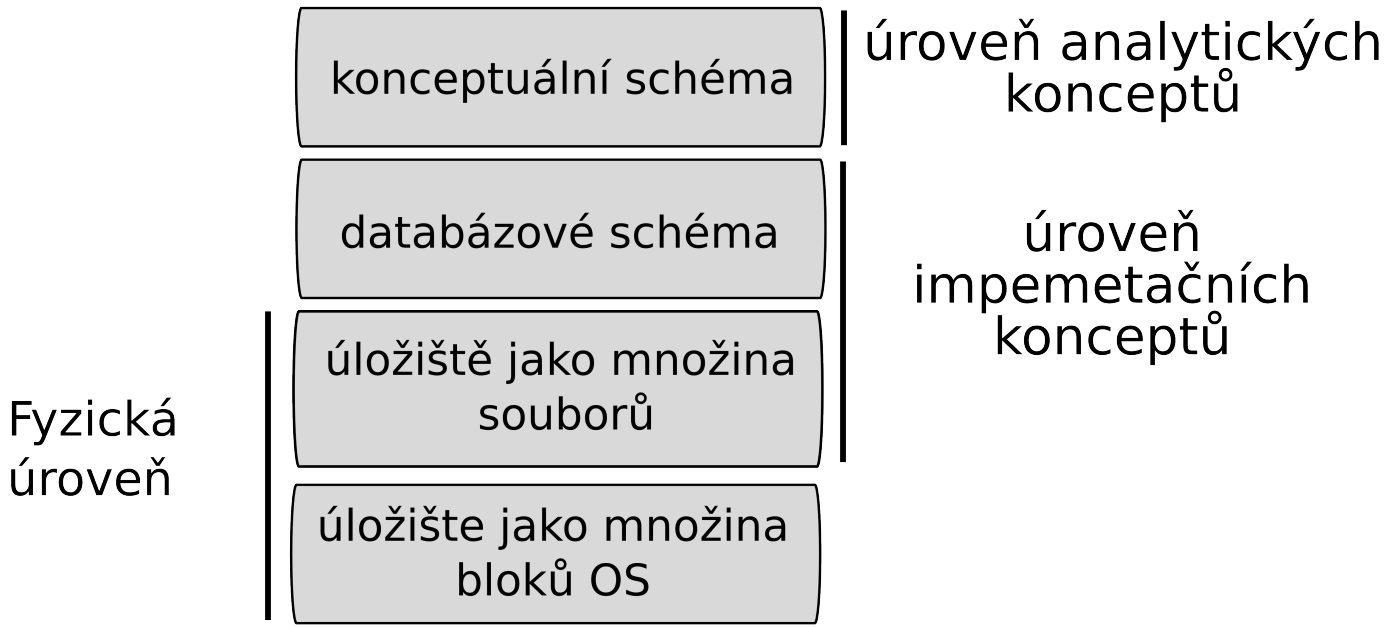
Katedra softwarového inženýrství
Fakulta informačních technologií
České vysoké učení technické v Praze

BI-DBS, LS 2021/2022

https://courses.fit.cvut.cz/BI-DBS/

Různé úrovně pohledu na data





Konceptuální, logická, fyzická úroveň

- 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, grafová, ...).
- Fyzická
  - Jde o fyzické uložení dat (sekvenční soubor, indexy, clustery...). Uživatelé (programátoři aplikací, příležitostní uživatelé) jsou od ní odstíněni logickou vrstvou DBMS.

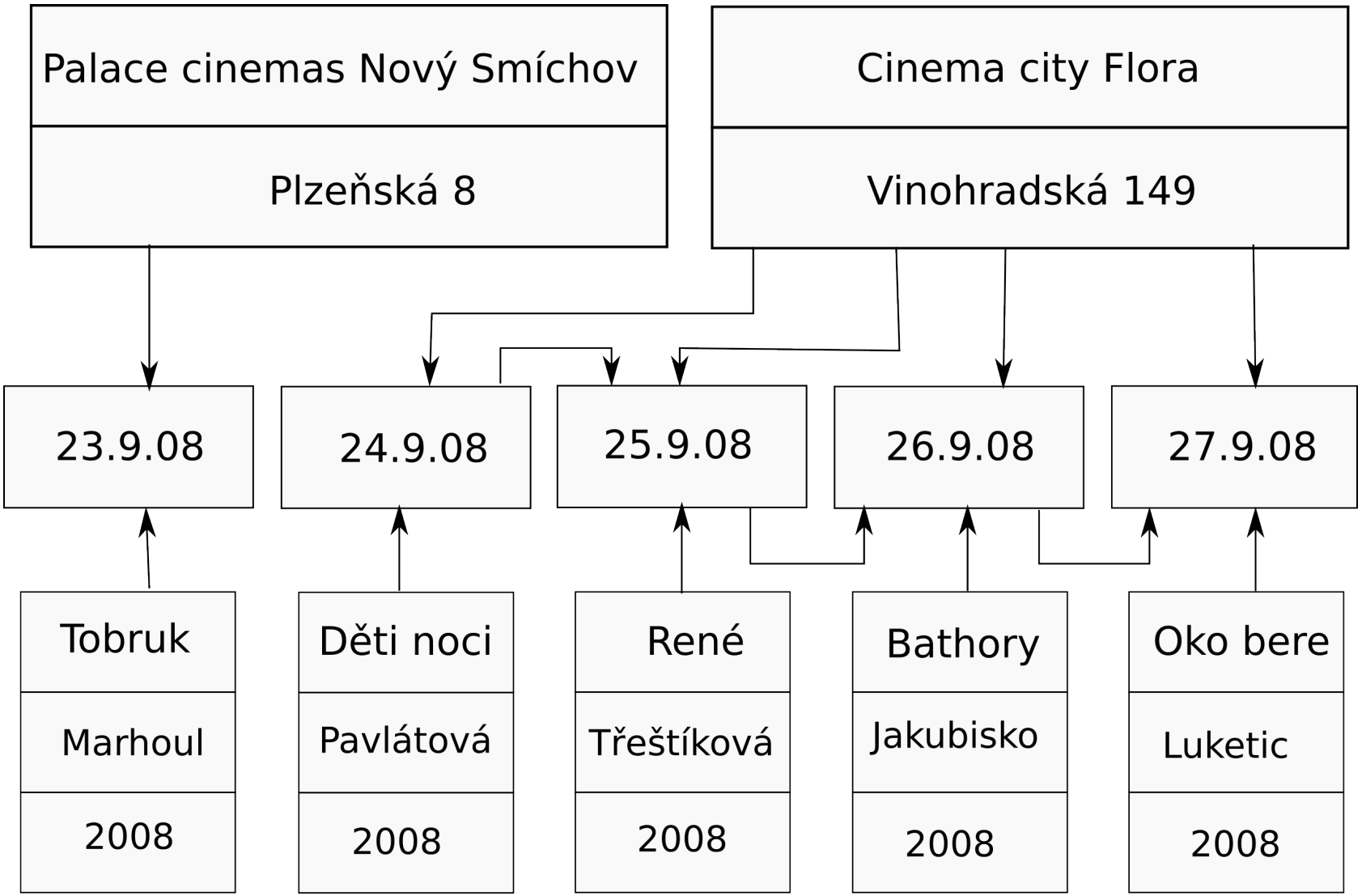
Logické (databázové) modely

- **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-ralační** - 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
- **grafový** – cca 2010
- **dokumentové, klíč-hodnota, sloupcové (wide column), ...**

Logické (databázové) modely

- Volba databázového modelu určuje prostředky pro vytváření struktury databáze (DDL) a prostředky pro tvorbu aplikací (DML, dotazovací jazyk, TCL, DCL)
- Příklady:
  - relační (objektově-relační) model - SQL
  - objektový model - OQL
  - XML model - XPath, XQuery
  - grafový model - Cypher, Gremlin

Síťový model – příklad – schéma výskytů



Sítový model – operace

- vytvoř **nový** záznam daného typu, **zruš** záznam, **změn záznam**
- **zařaď** členský záznam do c-množiny daného **vlastníka**
- **vyřaď** 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, příklad dotazování

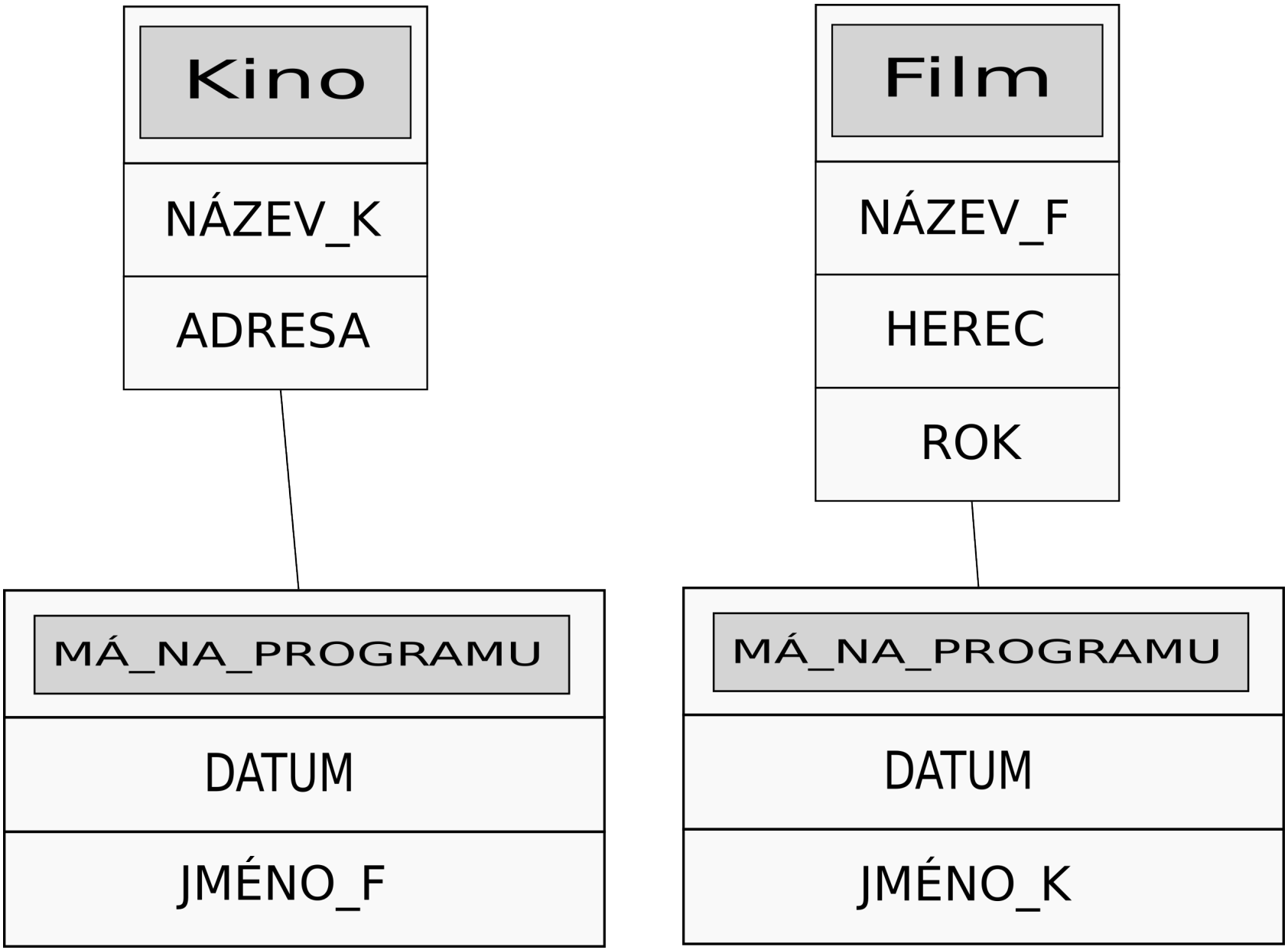
Dotaz: Vypiš program kina Blaník.

```
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;
```

Hierarchický model

- 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 Apollo

Hierarchický model – příklad



Relační model – charakteristika

- Jediný konstrukt – relace
  - schéma relace: jméno relace, jména atributů, specifikace domén atributů
  - prvky domén jsou atomické hodnoty (1.normální forma)
  - formální zápis R(A1:D1,...,An:Dn)
  - příklad: `KINO(NAZEV_K:CHAR(15), ADRESA:CHAR(25))`
- Integritní omezení: primární klíč, cizí klíč

Relační model – příklad – schéma

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

Zjevně nevhodná volba PK relace MA_NA_PROGRAMU. Proč?

Kino nemůže hrát jeden film víckrát (v jiný den a/nebo čas). Pro další výklad OK.

Relační model – příklad – data

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

Relační model – operace

- **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) **změn 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

Relační model – dotazování – příklad

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);
```

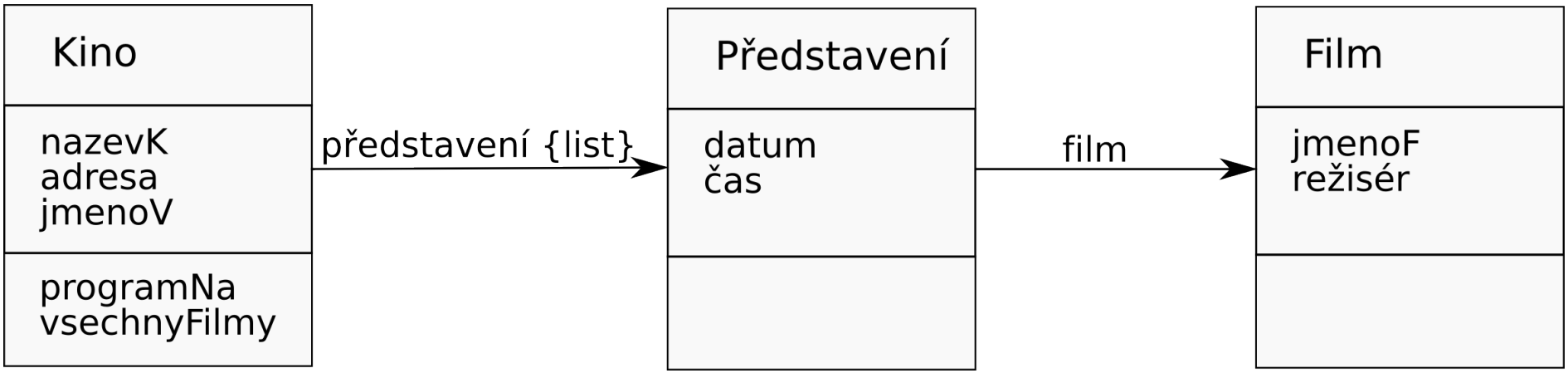
Objektový model – charakteristika

- **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.

Objektový model – konstrukty

- 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
```

objektově-relační model – charakteristika

Motivace

- relační model neumožňuje strukturované a vícehodnotové atributy
- nástup objektově orientovaných jazyků (od 80. let 20. století)
- obj. orient. jazyky jsou „intuitivnější“ (lépe popisují realitu)
- relační databáze se však staly „průmyslovým standardem“
- „sémantická mezera“ (semantic gap): relační data, objektové aplikace

Provedení

- „upgrade“ relačních databází i jazyka SQL
- krok 1: zavedení **uživatelem definovaných datových typů**
- krok 2: zavedení **objektových tabulek**
- standard SQL:99, implementace oracle8 (1997), ...

objektově-relační model – praxe

- implementace mnoha zajímavých rysů
- prakticky se využívají spíše ty základní
- otevřená cesta pro implementaci (strukturovaných) datových typů (enum, array, ..., dále XML, JSON, spatial (GPS,...), ...)
- všechny současné relační databáze nějaká taková rozšíření mají
- tedy, jsou spíše objektově-relační (minimálně porušením 1NF)

Jak se řeší „sémantická mezera“?

- rozšíření datových typů mnohdy postačuje
- technologie ORM (objektově relační mapování)
- jiné databázové modely (i v průmyslu)

Další kritika: relační/OR DB jsou příliš obecné, jde v nich „všechno“, ale někdy moc složitě, jindy zase moc pomalu.
Stav: v relačních/OR DB je dnes většina dat a pro mnoho případů užití je to stále nejlepší varianta.

XML model – charakteristika

- Podobá se hierarchickému – XML dokument je obvykle chápán jako strom; DOM API pro přístup.
- Kde se hodí?
  - Strukturovaná data jako instance (dokument),
  - potřeba validace (dokumentu),
  - větší flexibilita jednotlivých instancí (dokumentů).
- Datový model: elementy, atributy, PCDATA, zachovává pořadí (document order). Někdy je bohatší.
- Silné a standardizované dotazovací jazyky (XPath, XQuery)

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>
```

XML model – příklad – XPath

- Názvy kin v databázi:
  - `/program/kino/nazev_k`
- Všichni herci:
  - `//herec`
- Kina, která mají na programu aspoň 3 filmy:
  - `//kino[count(./hraje/film)>2]/nazev_k`
- Filmy, které hrají v kině Blaník:
  - `//kino[nazev_k="Blanik"]/hraje/film/nazev_f`

XML model – příklad – XQuery

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.

Grafové databáze

Historie se opakuje...

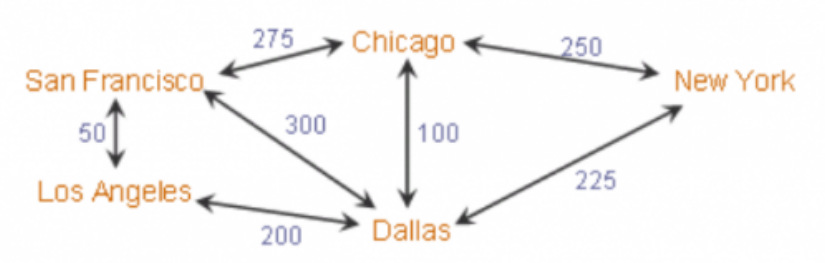
- síťové databáze: (orientovaný) graf, 60. léta 20. století
- hierarchické databáze: strom, konec 60. let
- relační: relace, 70. léta
- XML databáze: strom (+ drobné výjimky), 90. léta
- grafové databáze: graf (+ rozšíření), cca 2010

Grafové databáze

- datový model: property graph (přesněji property, oriented multigraph)
- dotazovací jazyky: Cypher (Neo4j), Gremlin
- databázové stroje: Neo4j, JanusGraph, OrientDB, ...

Poznámka: Zajímavé jsou také tzv. „multi-model“ databáze, které podporují více datových modelů (ArangoDB, OrientDB, ...).

Grafové databáze – příklad



Dotaz: 3 nejlevnější cesty ze San Francisca do New Yorku.

Jazyk Cypher

```
MATCH path=(s{code:'sf'})-[:DIRECT*1..5]->(d{code:'ny'})
RETURN extract(x in nodes(path) |x.code) as total_path,
   reduce(acc=0, x in relationships(path)| acc+x.price) as total_price
ORDER BY total_price
LIMIT 3;
```

Jazyk Gremlin

```
g.V().has('code','sf').
     repeat(outE().inV().simplePath()).until(has('code','ny')).
     project('path','total').
          by(path().by('code').by('price')).
          by(path().unfold().values('price').sum()).
     order().by(select('total')).
     limit(3);
```

Nové trendy v databázích

- NoSQL
  - ACID versus BASE (Basically Available, Soft-state, Eventually consistent)
  - nové aplikační domény: web, texty, semistrukturovaná data
  - více datových modelů - column-based, key-value, document oriented, graph, ...
- BigData
  - nová platforma pro podniková semi-strukturovaná data
  - pouze cca 10 % dat v podniku je přísně stukturovaných, krom toho: maily, smlouvy, objednávky, zápisy, ...
  - roste důležitost data maning a hlavně prezentace datových analýz (KPI, cockpit, ...)
- Distribuce, replikace, cloudy
  - distribuce a replikace nejen pro podporu bezpečnosti, ale hlavně škálování
  - horizontální škálování je snadné u jednoduchých DB modelů
  - velká funkční cloudová řešení (amazon, google, ...)
  - database as a service přístup (DaaS), opensource je v databázích brán velmi vážně

Shrnutí

V tomto předmětu jsme se věnovali **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
    (respektive volbou nevhodného si věci hodně zkomplikovat).
  - Volbu DB modelu je třeba dobře uvážit a zdůvodnit.
  - Často stejně zvítězí relační, protože je univerzální, ověřený časem a lze sehnat dostatek vývojářů.
    Nad něj se pro snadný vývoj aplikace (v objektovém PJ) nasadí ORM vrstva.