

# DBS – Konceptuální modelování

Michal Valenta

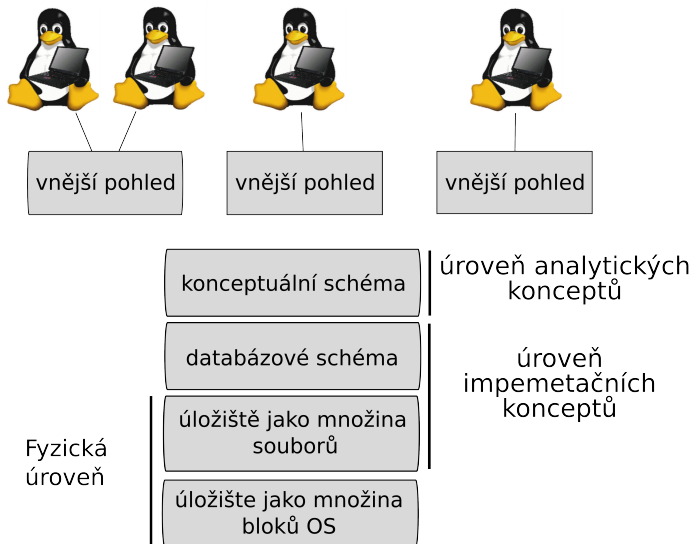
Katedra softwarového inženýrství FIT  
České vysoké učení technické v Praze  
©Michal Valenta, 2014

BI-DBS, ZS 2014/15

<https://edux.fit.cvut.cz/courses/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, ...).

- Fyzická

- ▶ Jde o fyzické uložení dat (sekvenční soubor, indexy, clustery, ...). Uživatelé (programátoři aplikací, příležitostní uživatelé) je od ní odstíněn logickou vrstvou SŘBD.

# Konceptuální modelování databází - proč?

- 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.

# Návrhy IS “postaru”

Funkční a datová analýza relativně oddělená.

- **Funkční schéma** – výsledek funkční analýzy a návrhu.
  - ▶ Kdo bude používat aplikaci? – Kategorie uživatelů systému.
  - ▶ Pracovní postupy v organizaci, které mají být počítačově podporovány.
  - ▶ Události, která spouští počítačově podporovaný pracovní postup.
- **Datové schéma** – výsledek datové analýzy a návrhu.

Problém: udržení konzistence funkční a datové analýzy.

# Návrh IS v objektově orientovaném prostředí

- Funkční a datová analýza není tak striktně oddělená.
- Objektový přístup zdůrazňuje zapouzdření dat, pracuje s voláním metod na objektech. Masivně využívá konstrukce jako agregace, kompozice, dědění.
- UML notace je objektově orientovaná.

# Prvky konceptuálního modelu

orientace na entity (třídy) a vztahy (asociace) mezi nimi

- entity (třídy), instance (objekty)  
atributy – doména, povinnost, identifikátor/unikátnost
- vztahy (asociace)
  - ▶ obecný vztah (asociace)  
kardinalita, parcialita
  - ▶ identifikační vztah (slabá a silná entita)
  - ▶ ISA-hierarchie (podtyp jako specializace)  
nepřehánět a pozor na sémantiku !!!

# Tvorba datového modelu – postup

- Identifikace **entit (entitních typů)** jako tříd objektů stejného typu.

## příklady entit

FILM, ZÁKAZNÍK, ZAMĚSTNANEC, KOPIE

- Identifikace **vztahů (vztahových typů)**, do kterých entity mohou vstupovat :

## příklady vztahu

ZAKAZNÍK (entita)

**MÁ\_PŮJČEN** (vztah)

FILM (entita)

- Identifikace atributů popisujících blíže vlastnosti entit a vztahů.



# Atributy a IO

## Příklady atributů:

- **příjmení** (atribut) zaměstnance (entita),
- **rodné číslo** (atribut) zaměstnance (entita),
- **datum** (atribut vztahu), do kdy má zákazník (entita) půjčenou (vztah) kopii filmu (entita).

## Příklady integritních omezení (IO):

- Doménou atributu **vaha** (entity zákazník) je integer.
- Atribut **vaha** (entity zákazník) musí mít jednu hodnotu (pro jednoho zákazníka).
- Atribut **rodné číslo** je identifikátorem entity zákazník.
- Atribut **datum** (vztahu půjčeno) může mít nejvýše jednu hodnotu.
- Atribut **herec** entity film může mít mnoho hodnot.

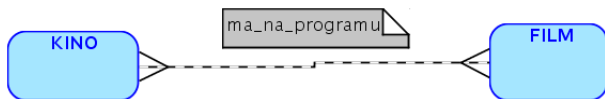
# Entita a vztah

## Lineární zápis:

Entity: Film, Kino

Relace: ma\_na\_programu (Film, Kino)

## Grafický zápis:

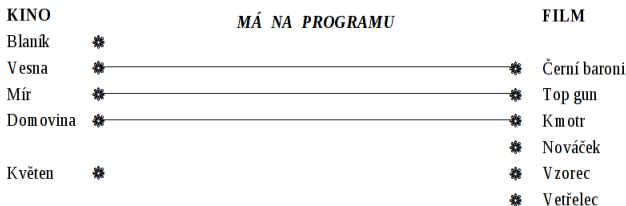


*Poznámka: převážně budu používat binární ER notaci (nástroj Oracle Data Modeller).*

# Atributy

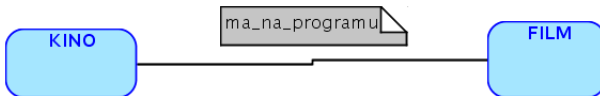


# Kardinalita 1:1



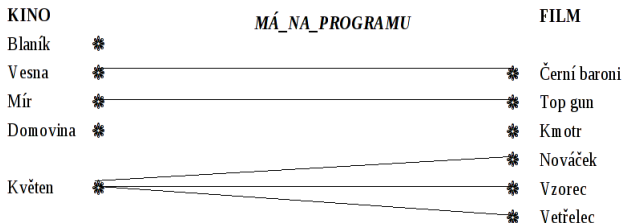
Kino hraje **nejvýše jeden** film.

Film je na programu **nejvýše jednoho** kina.



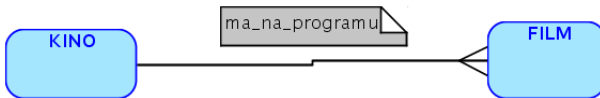
*Poznámka: povinnost/nepovinnost členství ve vztahu (parcialitu) budeme diskutovat později.*

# Kardinalita 1:N

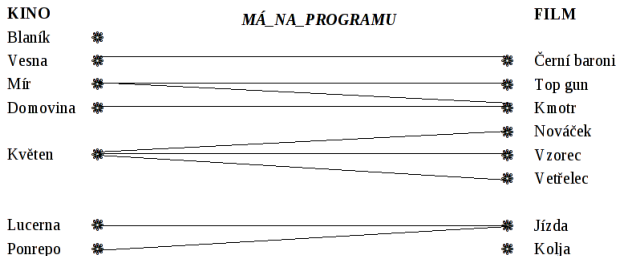


Kino **může** hrát **více** filmů.

Film je na programu **nejvýše jednoho** kina.

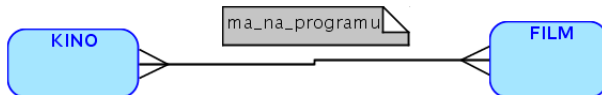


# Kardinalita M:N



Kino **může** hrát **více** filmů.

Film **může** být na programu **více** kin.

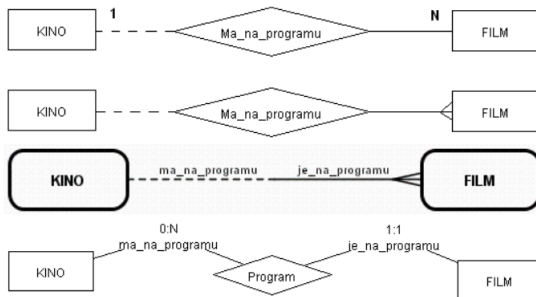


# Povinnost účasti ve vztahu (parcialita)

- povinná účast (obvykle značíme plnou čarou nebo “1”)
  - ▶ Všechny instance **musí** být zapojeny do příslušného vztahu.  
Kino **musí** mít na programu alespoň jeden film.  
Film **musí být** na programu alespoň jednoho kina.
- nepovinná účast (obvykle značíme přerušovanou čarou nebo “0”)
  - ▶ Jednotlivé instance **mohou, ale nemusí** být zapojeny do vztahu.  
Evidované kino **nemusí** hrát ani jeden film.  
(Kino **může** být evidováno i bez programu.)  
Evidujeme i filmy, které se nikde nehrají.  
(Film **nemusí** být na programu žádného kina.)

# Nepovinná účast – různé notace

Kino může hrát více filmů (ale také žádný).  
Film je na programu právě jednoho kina.



1 = 1..1



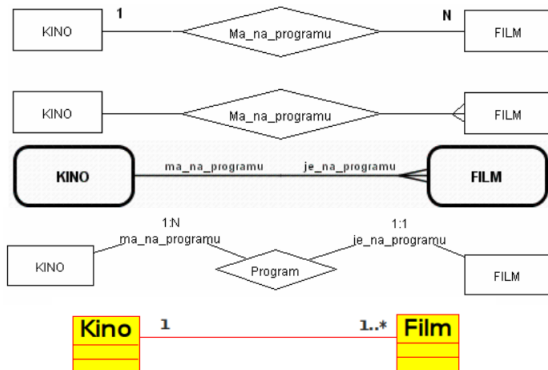
\* = 0..\*



# Povinná účast – různé notace

Kino hraje alespoň jeden film (ale může více).

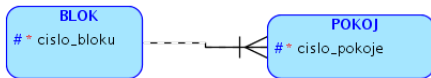
Film je na programu právě jednoho kina.



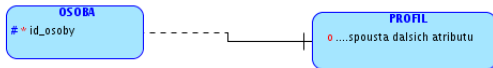
*Jak takovou databázi inicializujeme?*

# Identifikační závislost, slabá entita

Entita je identifikována (částečně nebo plně) vztahem k jiné entitě.



Slabá entita má vlastní atribut, který se podílí na identifikaci.



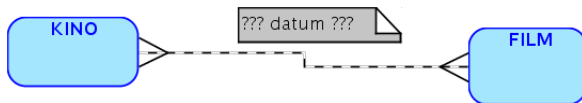
Slabá entita nemá vlastní atribut, který se podílí na identifikaci.

K jednomu vlastníkovi může patřit nejvýše jedna identifikačně závislá instance.

Každá osoba může mít nejvýše jeden profil.

# Atributy ve vztahu - motivace.

- Chenova notace atributy u vztahu povoluje (viz přednáška o databázových modelech).
- Binární notace (i UML Class Diagram) vyžadují explicitní dekompozici vztahu.

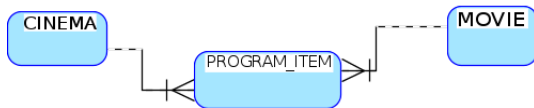


# Dekompozice vztahu M:N

Každý vztah M:N lze dekomponovat na 2 vztahy 1:N.  
Buď vložíme silnou entitu:

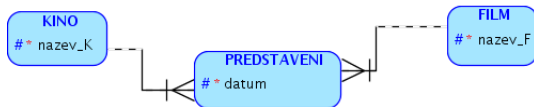


Nebo použijeme identifikační závislost:

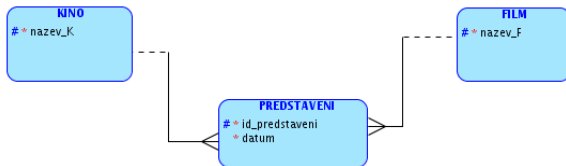


# Atributy ve vztahu - řešení

Použití identifikační závislosti:



Použití silné vztahové entity:



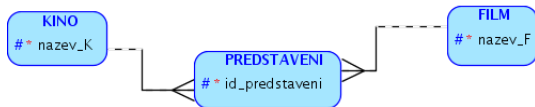
# Dekompozice vztahu M:N - řešení

Použití identifikační závislosti:



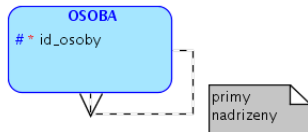
**Pozor:** v tomto případě může konkrétní kino hrát konkrétní film nejvýše jednou!

Použití silné vztahové entity:

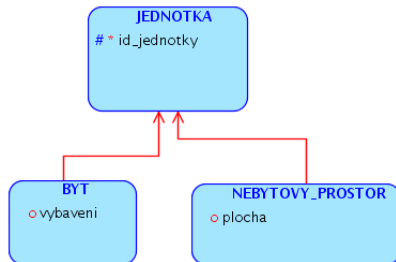


# Rekurzivní vztah

Používá se k vyjádření vztahu mezi instancemi stejné entity.  
Přímý nadřízený, předchůdce-potomek, část-celek, ...



# ISA hierarchie



Původně zkratka z anglického "Is a"

Pozor, ve striktním pojetí ER modelu se vyžaduje, aby každá instance nadtypu měla právě jednu instanci podtypu.

⇒ nevhodné pro role (např. student, učitel, ...), lepší identifikační závislost.

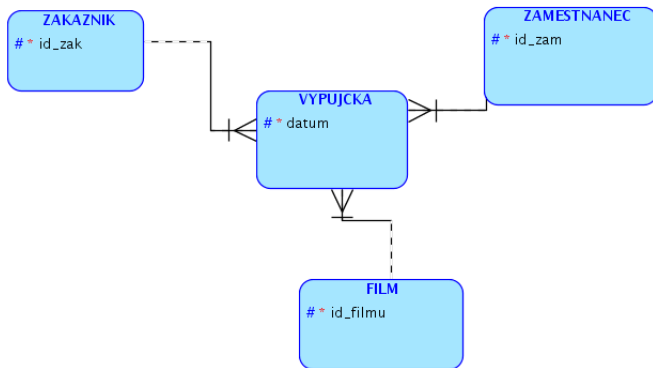


# Sémantický relativismus

- modelujeme situaci, kdy pacienti leží na pokojích
- zákazník zdůrazňuje, že je důležité znát na kolikalůžkovém pokoji pacient leží
- ??? PACIENT(ID\_PAC, JMENO, .., POCET\_LUZEK)
- lépe: PACIENT(ID\_PAC, JMENO...)  
POKOJ(ID\_POKOJE, POCET\_LUZEK, ...)
- Informace o počtu lůžek na pacientově pokoji je dohledatelná ze vztahu mezi pacientem a pokojem.

## Příklad - návrh videopůjčovny 1/3

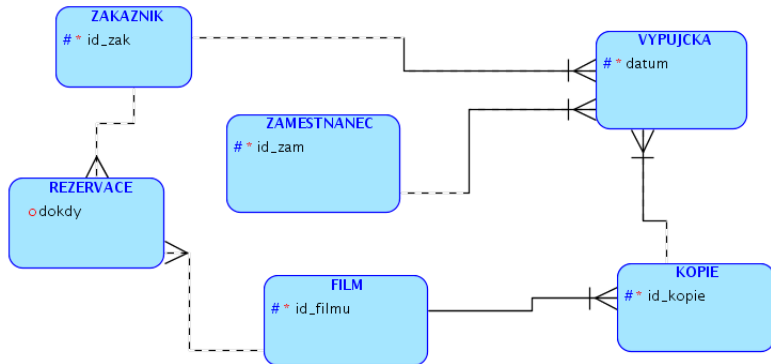
- Půjčovna se rozhodla použít počítač pro evidenci vyjpuček filmů. eviduje se skutečnost, kterou lze jednou větou vyjádřit jako :  
Zákazníci si od zaměstnanců půjčovny půjčují filmy.



*Poznámka: umělé versus přirozené identifikátory.*

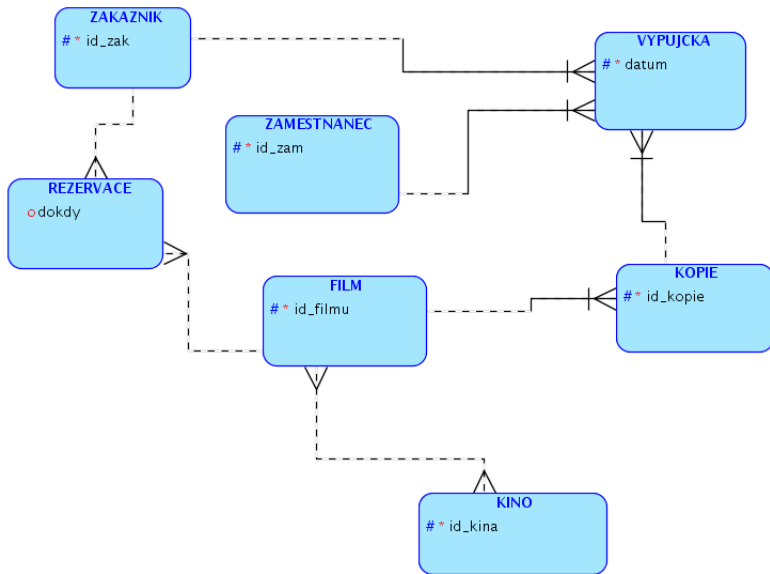
## Příklad - návrh videopůjčovny 2/3

... rozlišujeme KOPIE a FILMY, film si lze rezervovat:



## Příklad - návrh videopůjčovny 3/3

... a krom toho, chceme vědět, které filmy se ještě hrají v kinech...



## Příklad 2 – Multikina – přehled

Navrhovaná databáze se skládá ze tří logických a relativně autonomních částí (agend):

- evidence multikin, jejich sálů a vybavení a evidence zaměstnanců kin včetně jejich historie,
- evidence filmů a umělců, kteří se na nich podílejí,
- evidence představení a evidence prodaných vstupenek.

## Příklad 2 – Multikina – Evidence multikin a zaměstnanců

Každé multikino má nějaký název a adresu. Název nemusí být unikátní. Multikino je tvořeno typicky několika sály s různou kapacitou a různým vybavením.

Evidovaný zaměstnanec pracuje nejvýše pro jedno multikino. Každý zaměstnanec aktuálně pracuje nejvýše na jedné pozici, zároveň však chceme sledovat i jeho pracovní historii. U zaměstnanců sledujeme obvyklé personální údaje. Každý zaměstnanec může mít nejvýše jednoho přímého nadřízeného. Při stanovení platů zaměstnanců se může jejich vedoucí nezávazně řídit číselníkem platových tříd obsahujícím horní a spodní platovou hranici pro danou třídu.

## Příklad 2 – Multikina – Evidence filmů a umělců

U filmů evidujeme jejich název a rok natočení. Názvy filmů nemusí být unikátní. Dále chceme vědět, kdo byl režisérem příslušného filmu. Zajímají nás i herci v jednotlivých rolích. Počet sledovaných rolí pro jeden film může být libovolný. Dále je třeba mít možnost přiřadit film až do několika žánrových kategorií, stejně tak je třeba evidovat, že film byl natočen v koprodukcii i více zemí.

V naší databázi hodláme evidovat režiséry a herce. Zajímá nás jméno umělce, rok narození a země, ze které pochází. Umělec může být jak režisérem, tak hercem. Zároveň připustíme evidenci i takových umělců, kteří nepatří ani do jedné kategorie.

## Příklad 2 – Multikina – Evidence představení a vstupenek

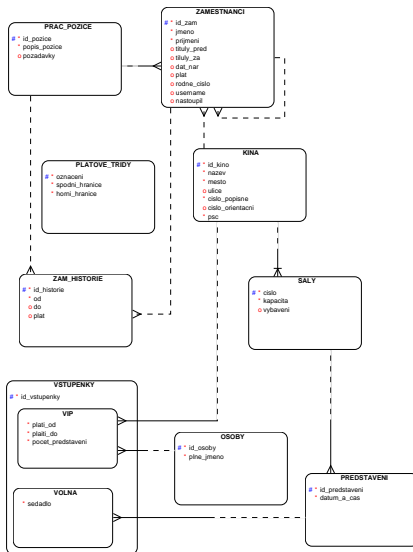
Představení je dáno datem a časem, kdy se hraje, filmem, který je na programu, a sálem, kde se koná.

Pro představení evidujeme prodej volných vstupenek. Vstupenka je vázaná k sedadlu a je vystavena na konkrétní představení. Kromě prodeje tzv. volných vstupenek evidujeme i VIP vstupenky. VIP vstupenka má časově omezenou platnost (od a do) a také je vázaná na konkrétní kino. VIP vstupenka se vystavuje na omezený počet představení.

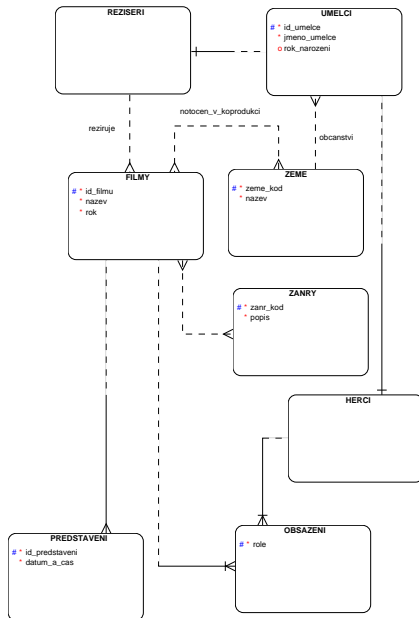
VIP vstupenka je vázaná na konkrétní osobu, u které sledujeme pouze její plné jméno.



# Příklad 2 – Multikina – ER schéma část 1



## Příklad 2 – Multikina – ER schéma část 2



# Poznámky na závěr

- Notace mohou být různé.
- Při konceptuálním modelu se staráme hlavně o **popis reality** , ne o řešení v konkrétním db stroji.
- Používáme **vztahy mezi entitami, kardinalitu a parcialitu** nikoliv cizí klíče.
- Jakmile se v obrázku objeví **cizí klíče**, už se nejedná o konceptuální model, ale o **grafickou podobu modelu relačního**