Téma 8 – Databáze – úvod

Obsah

1. Základní pojmy databází
2. Abstrakce, schémata, pohledy
3. Databázové modely
4. Modelování reálného světa
5. Entity a vztahy
6. Entity-Relationship (E-R) model
7. E-R diagramy
8. Převod E-R modelu na tabulky dat
9. Normalizace

Základní pojmy

- Často se směšují pojmy
  - databáze
    - Vlastní data zpravidla ve formě tabulek
  - systém řízení báze dat SŘBD, databázový systém (program), DBMS
    - Komplex softwarových komponent pro (pokročilou) práci s daty
- Základní terminologie databázových systémů
  - Logická struktura dat
    - abstrakce od fyzické organizace uložených dat
    - založeno na vhodném modelu dat
    - pohledy na data (*data views*) – uživatelsky zajímavé dílčí údaje
  - Prostředky pro rychlý přístup k datů
    - klíče – indexy organizované s ohledem na rychlost vyhledávání
  - Prostředky pro pohodlné užívání dat
    - sestavy nebo formuláře; používají se pro výstup dat (tisk, prezentaci nebo pouhé zobrazení). Sestavy mohou být např. doplněny o filtry, které vyberou jen požadované záznamy.
  - Ochranné prostředky
    - uživatelská oprávnění – definice přístupu uživatelů k objektům databáze

Účel databázových systémů

- Historické databáze
  - Speciální programy využívající souborů přímo nad OS
  - Nová úloha = nový program pro správu dat
- Nevýhody přímého použití souborů
  - Redundance a nekonzistence dat
    - Nejednotné formáty souborů, duplikace informací v různých souborech
    - Izolace dat
  - Problémy s přístupem k datům
    - Pro každou „novou úlohu“ je nutno napsat nový program
  - Integritní problémy
    - Integritní omezení (např. mzda > 0) je zanořeno „někde v programu“ místo explicitního vyjádření v požadavcích na datový obsah
    - Velmi obtížné aktualizace podmínek (přidání nebo změna)
  - Atomicita operací s daty
    - Havárie mohou nechat databázi v nekonzistentním stavu, když aktualizace proběhne jen částečně
      - *Příklad:* Převod peněz z jednoho účtu na druhý musí proběhnou buď úplně nebo vůbec ne
  - Souběžný přístup více uživatelů
    - Souběh je nutný z aplikačního hlediska; neřízené přístupy mohou vést k nekonzistencím

Systém řízení báze dat (SŘBD=DBMS)

- DBMS pracuje s libovolnými daty podle zadaného schématu
  - Univerzální program řešící přístup k datům, integritu a atomicitu dat
  - Paralelní přístup k datům i distribuované databáze
  - Prostředí, které je uživatelsky *přívětivé* a při tom *efektivní*
- Aplikace databází
  - Bankovnictví:
    - veškeré transakce
  - Aerolinky:
    - rezervace, letové řády, opravy letadel, ...
  - University:
    - registrace, studijní plány, rozvrhy, diplomy, ... (*KOS*)
  - Prodejci:
    - zákazníci, katalogy, jednotlivé obchody
  - Výroba:
    - produkce, sklady, objednávky, zásobování
  - Personalistika:
    - záznamy o zaměstnancích, mzdy, daňové povinnosti, ...
- Databáze jsou všude kolem nás

Úrovně abstrakce

- Fyzická úroveň
  - popisuje, jak je uložen záznam (např. délka řetězce, velikost celého čísla, reálné číslo)
  - Zrychlení přístupu k datům pomocí indexů
- Logická úroveň
  - popisuje jaká data jsou v databázi uložena a vztahy mezi daty

```
type zákazník = record
                    zákazník_id: string;
                    zákazník_jméno: string;
                    zákazník_adresa: string;
                    zákazník_psč: integer;
end;
```

- Úroveň pohledů
  - Aplikace zakrývají detaily dat, umožňující modifikovat databázi.
  - Pohledy mohou mít týž účel, tedy např. zakrýt výši mzdya zaručit tak důvěrnost či utajení některých údajů
- Konceptuální pohled
  - Modelování reality
  - Snaží se navrhnout logickou úroveň databáze
  - Často používá grafickou úroveň pro přehledný návrh

Schémata a instance

- Schéma – logická struktura databáze
  - Analogie s ***typem*** (*třídou*) proměnné v programu
  - Příklad:
    - Databáze obsahuje informace o množině vyučovaných předmětů, množině časových úseků učeben a vztahů mezi nimi (tj. kdy se kde co učí)
  - Fyzické schéma: struktura databáze na fyzické úrovni
  - Logické schéma: struktura databáze na logické úrovni
- Instance – skutečný obsah databáze v daný okamžik
  - Analogie s hodnotou proměnné (stavem objektu) v programu
- Nezávislost na fyzických datech – možnost modifikovat fyzické schéma beze změny logického schématu
  - Aplikace se opírají pouze o logické schéma a nezávisí na fyzickém zobrazení
  - Obecně: Rozhraní mezi různými úrovněmi (vrstvami) a komponentami jsou přesně definována, takže změny v některých částech neovlivňují zbytek

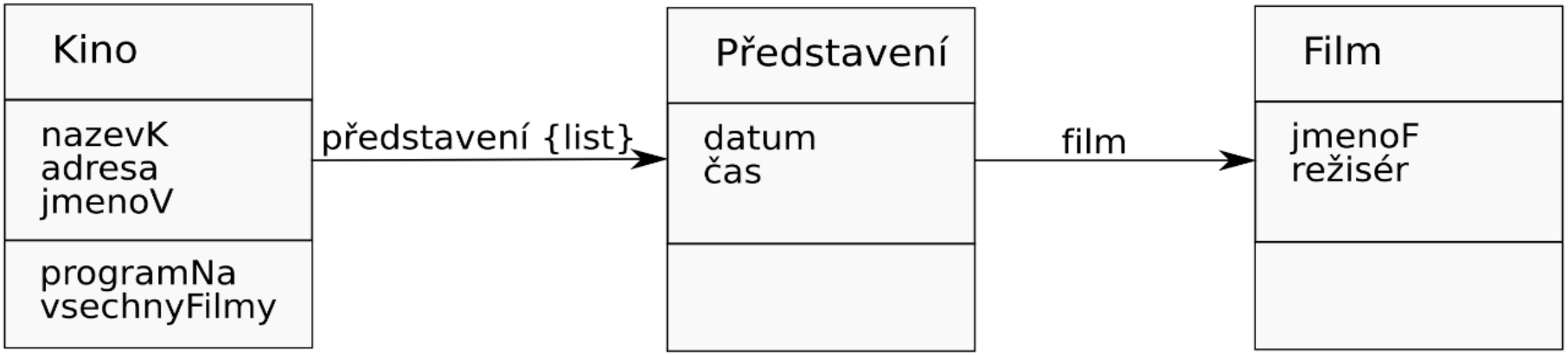
Schéma dat = Modely dat

- Množina prostředků pro popis
  - Vlastních dat
  - Vztahů mezi daty
  - Významu (sémantiky) dat
  - Omezení kladených na data
- Historické modely dat
  - Hierarchický model
  - Síťový model
- Relační model dat ➔
  - v současnosti nejužívanější model
  - Entity-Relationship model (E-R model) ➔
    - abstraktní a konceptuální znázornění reality
    - formální nástroj pro návrh databáze a jejího logického schématu
- Objektové modely dat
  - Objektové databáze a Objektově relační databáze
- Polostrukturované datové modely
  - slouží zpravidla pro výměnu dat mezi různými systémy
  - typický příklad je XML (*eXtensible Mark-up Language*)

Objektový model

- Objektový model = data + metody
  - Mezi objekty existuje skládání, dědění, závislosti
  - Polymorfismus umožňuje nadefinovat objekty s různou strukturou dat i metod
  - Protokol objektu je dán množinou přístupných zpráv
  - Identita objektu je dána nejen vnitřními, ale i vnějšími vazbami
  - Klíče jsou interní záležitostí objektu
- Objekt = základní konstrukt
  - Objekt je instance dané třídy (obsahuje informace o jménech dat = atributů, jejich doménách = typech, názvech metod pro přístup k nim,...)
  - Objekt má stav (hodnoty atributů)
- Kolekce = množinové konstrukce
  - Set = množina, list = seznam, array = pole
  - Množinové operace – sjednocení, průnik, rozdíl, ...

Objektový model - příklad



- Metody objektu Kino

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

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

XML databáze / model

- Databáze je XML soubor/soubory
- Datový model definuje strukturu XML souboru
  - Hierarchická definice struktury XML
  - Definice elementů, atributů, zachovává pořadí
- Dotazovací jazyk Xquery
  - Procedurální „programovací“ jazyk pro analýzu dat
  - Výsledkem dotazu je zase XML soubor
  - Velmi jednoduše analyzuje obsah XML souboru a umožňuje získat z něj potřebné informace
- DBMS podporující XML databáze
  - IBM DB2, Microsoft SQL Server, Oracle Database, PostgreSQL
- Čisté XML databáze
  - BaseX, eXistDB, MarkLogic

XML databáze - příklad

```
<PLAY>
    <TITLE>The Tragedy of Hamlet, Prince of Denmark</TITLE>
    <SCNDESCR>SCENE  Denmark.</SCNDESCR>
    <PLAYSUBT>HAMLET</PLAYSUBT>
    <ACT>
        <TITLE>ACT I</TITLE>
        <SCENE>
            <TITLE>SCENE I.  Elsinore. A platform before the castle.</TITLE>
            <STAGEDIR>FRANCISCO at his post. Enter to him
                                BERNARDO</STAGEDIR>
            <SPEECH>
                <SPEAKER>BERNARDO</SPEAKER>
                <LINE>Who's there?</LINE>
            </SPEECH>
            <SPEECH>
                <SPEAKER>FRANCISCO</SPEAKER>
                <LINE>Nay, answer me: stand, and unfold yourself.</LINE>
            </SPEECH>
            <SPEECH>
                <SPEAKER>BERNARDO</SPEAKER>
                <LINE>Long live the king!</LINE>
            </SPEECH>
            ……
```

XQuery - příklad

```
<html><head/><body>
 {
  for $act in doc("hamlet.xml")//ACT
  let $speakers := distinct-values($act//SPEAKER)
  return
    <div>
      <h1>{ string($act/TITLE) }</h1>
      <ul>
      {
        for $speaker in $speakers
        return <li>{ $speaker }</li>
      }
      </ul>
    </div>
 }
 </body></html>
```

Relační databáze

- Od této chvíle budeme hovořit pouze o relačních databázích
- Nejrozšířenější přístup k uložení dat
- Dobře propracovaná matematická teorie

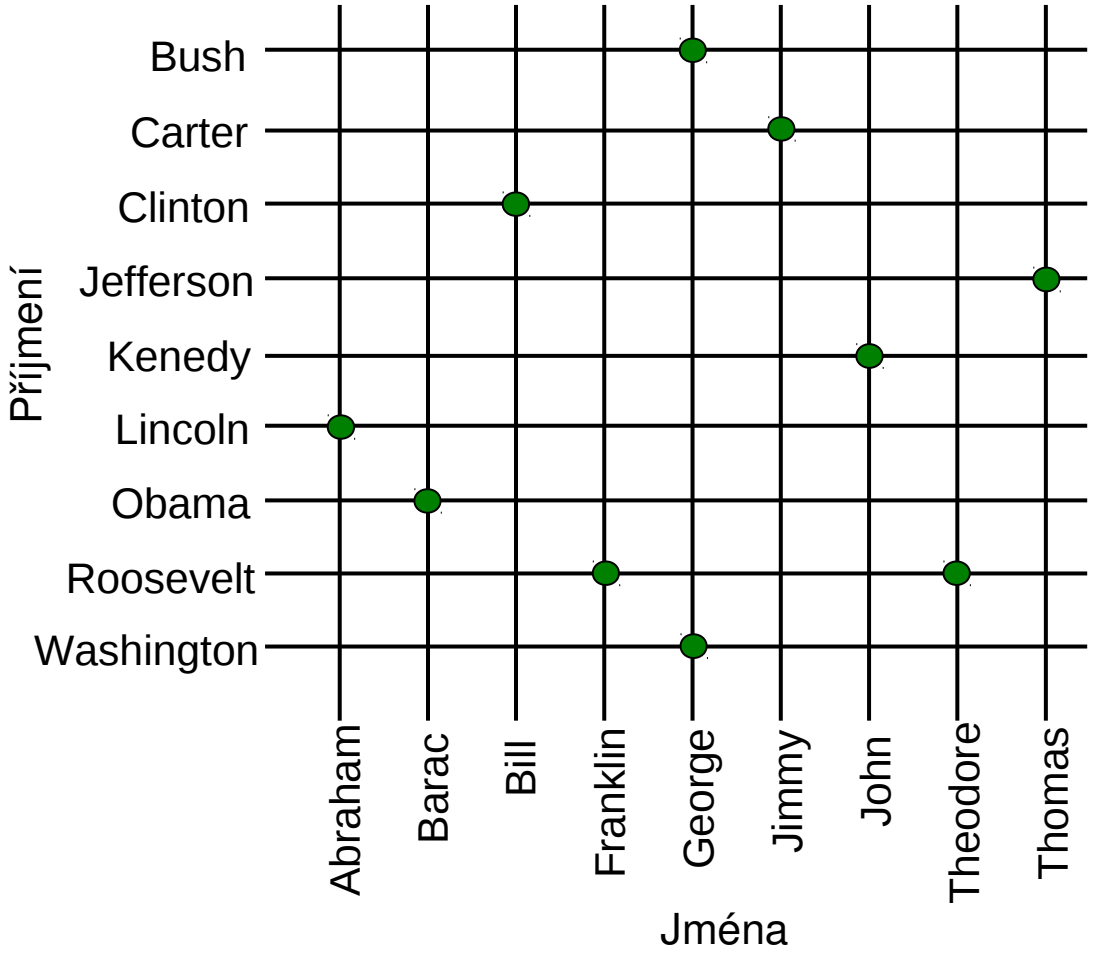
Co to je *relace*?

- Matematicky: Jsou dány množiny D_1, D_2, …, D_n, pak relací R rozumíme podmnožinu kartézského součinu $D_1 \times D_2 \times \ldots \times D_n$. Relace tedy je množina *n*-tic (a_1, a_2, …, a_n), kde $a_i \in D_i$
- Příklad:
  - *klient_jmeno* = {Novák, Mates, Braun, Novotný …}
    /* množna jmen klientů */
  - *klient_ulice* = {Spálená, Hlavní, Horní, …} /* množina jmen ulic*/
  - *klient_mesto* = {Praha, Brno, Nymburk, …} /* množina jmen měst */
  - relace *r* = {
    (Novák, Spálená, Praha),
    (Mates, Horní, Brno),
    (Braun, Hlavní, Brno),
    (Novotný, Horní, Nymburk)
    }

    je relace, tj. podmnožina *klient_jmeno* x *klient_ulice* x *klient_mesto*
- Vzhledem k tomu, že jde vždy o konečné množiny, lze je vyjádřit výčtem, tedy tabulkami

Co je relace?

- Relace je jakákoliv podmnožina kartézského součinu
- V množinách neexistuje duplicita
  - Velmi důležité pro databázové aplikace
- Prvky množiny mohou být v jakémkoliv pořadí
  - neexistuje uspořádání



Vybraní američtí prezidenti

Relační schéma a instance

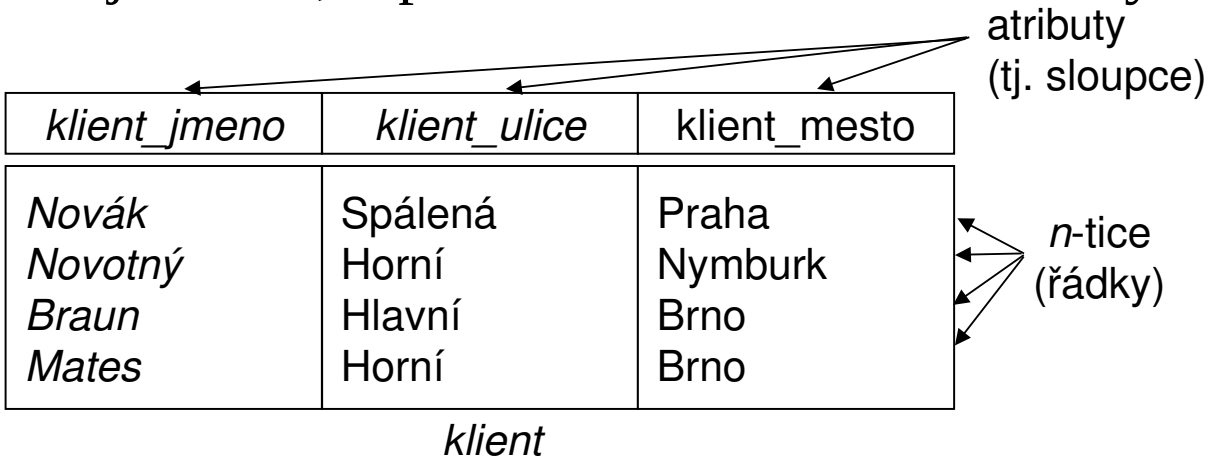
- Relační schéma
  - $A_1, A_2, \ldots, A_n$ jsou *atributy*
  - $R = (A_1, A_2, \ldots, A_n)$ je ***relační schéma***

    Příklad:

    Klient_schema = (*klient_jmeno*, *klient_ulice*, *klient_mesto*)
  - *r*(*R*) značí *relaci r* nad *relačním schématem R*

    Příklad:

    klient (*Klient_schema*)
- Instance relace (relační instance)
  - Skutečné hodnoty (*relační instance*) jsou definovány výčtem, tj. tabulkou
  - Prvek *t* relace *r* je *n*-tice, reprezentovaná *řádkem* tabulky



Typy atributů

- Každý atribut v relaci má své jméno
- Množina přípustných hodnot atributu je definiční doménou atributu
- Hodnoty atributu jsou (téměř vždy) atomické, tj. dále nedělitelné
  - Např. hodnotou atributu „číslo_účtu“ smí být číslo jednoho účtu, nikoliv množina čísel účtů
- Speciální hodnota *null* patří do každé domény
  - prázdná (nezadaná) hodnota
  - null značně komplikuje definici mnoha množinových operací, a proto zpočátku tuto hodnotu budeme ignorovat
    - důsledky uvedeme později

Datově orientované „programovací“ jazyky

- Jazyky pro definici dat
  - *Data Definition Language* (**DDL**)
  - Popis definice databázového schématu
    - Příklad:
      create table *ucet* (*cislo_uctu*: char(10), *zustatek*: integer)
  - Překladač DDL generuje množinu tabulek uložených v tzv. slovníkem dat (*data dictionary*)
  - Slovník dat obsahuje metadata (tj. data o datech – „znalosti“)
    - Databázové schéma
    - Způsob uložení dat a datové typy
      - Specifikují se struktury „datových souborů“ a způsoby přístupu
    - Integritní omezení
      - Doménová omezení (např. pouze nezáporná čísla)
      - Referenční integrita (odkazy a vazby dat)
    - Tvrzení (a*ssertion*)
      - Obecná omezení ve tvaru predikátů popisující povinné podmínky, které nelze zahrnout do omezení integritních (např. *katedra* musí v každém semestru nabízet aspoň 5 předmětů)
    - Autorizační informace
      - Např. *rodné číslo* je diskrétní údaj, který mohou vidět jen oprávnění uživatelé
- Příklady: SQL, ER model, UML, XML

Relační model - operace

- Vytvoř novou relaci (tabulku) podle zadaného schéma
- Přidej novou n-tici (řádek) do existující relace (tabulky)
- Vymaž n-tice (řádky) zadaných vlastností
- Ve vybraných n-ticích (řádcích) změň hodnoty atributů (sloupců)
- Vytvoř novou relaci (tabulku) z existující relace:
  - Výběrem n-tic(řádků) zadaných vlastností – selekce
  - Výběrem zadaných atributů (sloupců) – projekce
  - Množinovou operací z existujících relací (tabulek) – sjednocení, průnik, rozdíl, kartézský součin
  - Operací spojení ze zadaných existujících relací (tabulek)
- Odstraň relaci (tabulku)

DBMS pomocí relačních operací vytvoří novou relaci jako odpověď na zadanou otázku.

Datově orientované „programovací“ jazyky

- Jazyky pro manipulaci s daty
  - Data Manipulation Languages (**DML**)
  - Jazyk pro přístup k datům a manipulaci s daty (modifikaci) organizovanými podle příslušného modelu
    - též dotazovací jazyk (*query language*)
  - Dvě třídy DML
    - Procedurální – uživatel (programátor) udává jaká data chce a jak je získat
    - Deklarativní (neprocedurální) – uživatel (programátor) udává jaká data chce, aniž by zadával způsob jejich získání
  - Nejrozšířenější dotazovací jazyk je SQL
    - Structured Query Language
    - deklarativní jazyk
    - Příklad:

      select *st_jmeno*, *st_prijmeni* **from** *studenti* **where** *st_login* = ‘xnovak’

SQL

- SQL – nejrozšířenější neprocedurální dotazovací jazyk
  - Příklad: Najdi jméno zákazníka s *klient_id* 12-345

```
select    klient.klient_jmeno
from      klient
where     klient.klient_id = '12-345'
```

  - Příklad: Najdi zůstatky všech účtů patřících klientovi s *klient_id* 12-345

```
select    ucet.zustatek
from      vkladatel, ucet
where     vkladatel.klient_id = '12-345' and
          vkladatel.ucet_id = ucet.ucet_id
```

  - SQL standard je primárně DML, avšak má i DDL příkazy
- Aplikační programy obvykle přistupují k databázím přes
  - rozšíření příslušného programovacího jazyka, které umožňuje využívat SQL jako jazykový konstrukt
    - např. propojení PHP s MySQL – časté pro webové aplikace
  - API knihoven schopných zaslat SQL příkazy databázovému (sub)systému
    - např. ODBC = *Open Database Connectivity* či JDBC = *Java DataBase Connectivity*

Návrh relačního modelu

- Příklad tabelovaných dat v relačním modelu
  - Relace je množina (➔) zobrazená výčtovou tabulkou
  - Sloupce se nazývají atributy

klient_id	klient_jmeno	klient_ulice	klient_mesto	ucet_id	zustatek
12-345	Novák	Spálená 22	Praha 1	A-101	500
12-345	Novák	Spálená 22	Praha 1	A-201	700
12-346	Kovář	Hlavní 10	Brno	A-102	450
12-358	Mates	Horní 135	Benešov	A-118	800
25-836	Braun	Karlovo n. 13	Nymburk	A-249	550
35-795	Soukup	Hlavní 25	Praha 4	A-357	635
45-678	Novotný	Česká 1	Ostrava	A-201	700

  - Špatně navrženo – data se opakují, problémy s aktualizací
- Ukázka relační databáze

klient_id	klient_jmeno	klient_ulice	klient_mesto
12-345	Novák	Spálená 22	Praha 1
12-346	Kovář	Hlavní 10	Brno
12-358	Mates	Horní 135	Benešov
25-836	Braun	Karlovo n. 13	Nymburk
35-795	Soukup	Hlavní 25	Praha 4
45-678	Novotný	Česká 1	Ostrava

Tabulka *klient*

ucet_id	zustatek
A-101	500
A-201	700
A-102	450
A-118	800
A-249	550
A-357	635

Tabulka *ucet*

klient_id	ucet_id
12-345	A-101
12-345	A-201
12-346	A-102
12-358	A-118
25-836	A-249
35-795	A-357
45-678	A-201

Tabulka *vkladatel*
uvádí **vztah** mezi tabulkami *klient* a *ucet*

Návrh databáze

- Obecný návrh databáze je složitý několikastupňový proces
- Logický návrh
  - Vytváří se logické schéma databáze. Cílem je nalézt dobře koncipovanou sadu datových tabulek a jejich vzájemných vazeb tak, aby schéma obsahovalo minimum duplicit a bylo co nejotevřenější pro případné modifikace struktury.
    - Obecně úloha softwarového inženýrství
  - Logický návrh zahrnuje dvě klíčová rozhodnutí:
    1. Rozhodnutí dle účelu: Jaká data mají být zaznamenávána v databázi
    2. Rozhodnutí o struktuře: Jak potřebná data rozdělit mezi tabulky dat (relace) a jak koncipovat příslušné databázové schéma.
- Fyzický návrh
  - Rozhodnutí o zobrazení datových tabulek do samostatných komponent v databázi
    - Důležité z pohledu efektivity a údržby dat

Entity-Relationship model

- E-R model je jedním z nejčastěji používaných návrhových prostředků
  - Modeluje „oblast zájmu“ (např. podnik) jako kolekci ***entit*** and a ***vztahů*** (*relationships*) mezi nimi
  - Entita: je nějaká “věc” nebo “objekt” jednoznačně odlišitelná od ostatních
    - Entita je popsána množinou svých atributů
  - Vztah: propojení mezi dvěma či více entitami
  - POZOR: Nezaměňovat relace (*relation*) a vztah (*relationship*)!
  - Reprezentuje se graficky tzv. *Entity-Relationship diagramem* (E-R diagram)
  - Volné nástroje Oracle Data Modeller, MySQL Workbench a řada dalších

E-R modelování: Entity

- Entita je objekt, který existuje a je odlišitelná od ostatních objektů
  - Příklad: určitá osoba, podnik, kulturní akce, technické zařízení
  - Entity jsou obecné řeči obvykle vyjádřeny jako podstatná jména
- Entity mají vlastnosti označované jako *atributy*
  - Příklad: lidé mají *jména* a *adresy*
- Množinou entit rozumíme množinou tvořenou entitami stejného typu, tj. entitami s týmiž vlastnostmi
  - Často se místo pojmu množina entit používá entitní typ
  - Příklad: množina osob, množina stromů, množina bankovních půjček, množina údajů sbíraných z čidel

klient_id	klient_jmeno	klient_ulice	klient_mesto
12-345	Novák	Spálená 22	Praha 1
12-346	Kovář	Hlavní 10	Brno
12-358	Mates	Horní 135	Benešov
25-836	Braun	Karlovo n. 13	Nymburk
35-795	Soukup	Hlavní 25	Praha 4
45-678	Novotný	Česká 1	Ostrava

klient

pujcka_id	castka
L-101-A	1500
L-201-A	2700
L-102-C	1450
L-118-D	3800
L-249-B	2550
L-157-A	6350

pujcka

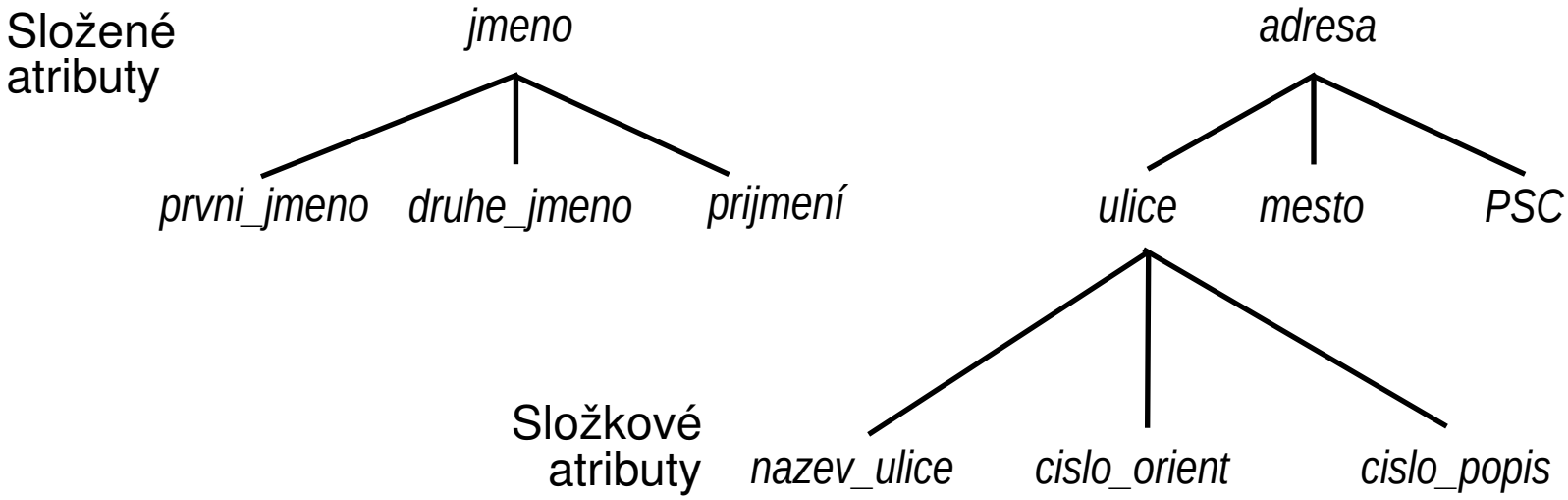
E-R modelování: Atributy

- Entita je reprezentována množinou atributů popisujících vlastnosti všech prvků patřících do příslušné množiny entit.
  - Fakticky je tím definována struktura datového typu (záznamu), který nese informaci o jednom každém prvku množiny

    Příklad:
    klient = (*klient_id*, *klient_jmeno*, *klient_ulice*, *klient_mesto*)
    pujcka = (*pujcka_id*, *castka*)

- Doména atributu
  - množina přípustných hodnot pro každý atribut
- Typy atributů
  - *Jednoduché* a *složené* atributy
  - *Jedno-* nebo *vícehodnotové* atributy
    - Příklad vícehodnotového atributu: *telefonni_cisla*
  - *Odvozené* (též *počítané*) atributy
    - Dají se vypočítat z hodnot jiných atributů
    - Např. věk, je-li známo *datum_narození*

Složené a složkové atributy



E-R modelování: Vztahy

- Vztah definuje propojení mezi dvěma či více entitami

  Příklad:

  Novák — *ukládá na* — A-102

  entita *klient* — vztah *vkladatel* — entita *ucet*

  (zobrazeno jako entita)

  – Vztahy obvykle představují slovesné fráze

- Množina vztahů je množina propojení mezi dvěma (či více) množinami entit
  – matematicky zapsáno: množina vztahů mezi $n \geq 2$ entitami je množina n-tic, kde každý prvek n-tice je vybrán z jedné množiny entit

  $$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

  kde $(e_1, e_2, \ldots, e_n)$ je vztah

  - matematicky jde o relaci ➔

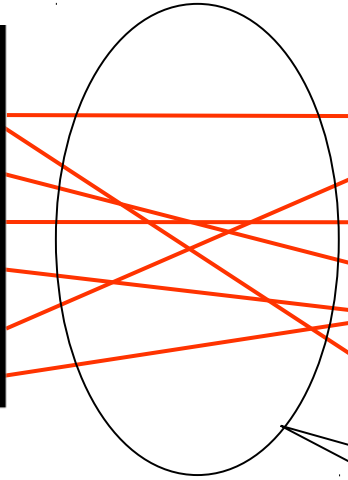
  – Příklad:

  $$(\text{Novák}, \text{A-102}) \in \textit{vkladatel}$$

Množina vztahů

klient_id	klient_jmeno	klient_ulice	klient_mesto
12-345	Novák	Spálená 22	Praha 1
12-346	Kovář	Hlavní 10	Brno
12-358	Mates	Horní 135	Benešov
25-836	Braun	Karlovo n. 13	Nymburk
35-795	Soukup	Hlavní 25	Praha 4
45-678	Novotný	Česká 1	Ostrava

klient



pujcka_id	castka
L-101-A	1500
L-201-A	2700
L-102-C	1450
L-118-D	3800
L-249-B	2550
L-157-A	6350

pujcka

množina vztahů *dluzi*

- Množiny vztahů vyjádřeny tabulkami
  - podobně jako entity
  - Mohou mít dokonce připojené atributy
    - např. datum poslední splátky
- Množiny vztahů mohou propojovat více množin entit
    - např. (*zaměstnanec*, *pozice*, *oddělení*)
  - většinou se ale používají binární množiny vztahů
    - propojení dvou množin entit

klient_id	pujcka_id	datum
12-345	L-101-A	01.04.2011
12-345	L-157-A	25.03.2011
12-346	L-118-D	27.02.2011
12-358	L-102-C	15.03.2011
25-836	L-249-B	12.04.2011
35-795	L-201-A	01.04.2011
45-678	L-249-B	28.03.2011

dluzi

Kardinalita vztahů

- Kardinalita určuje počet prvků množiny entit přidružené prostřednictvím množiny vztahů
  - Pro binární množiny vztahů jsou možné 4 typy

$1 : 1$

$1 : N$

$N : 1$

$M : N$

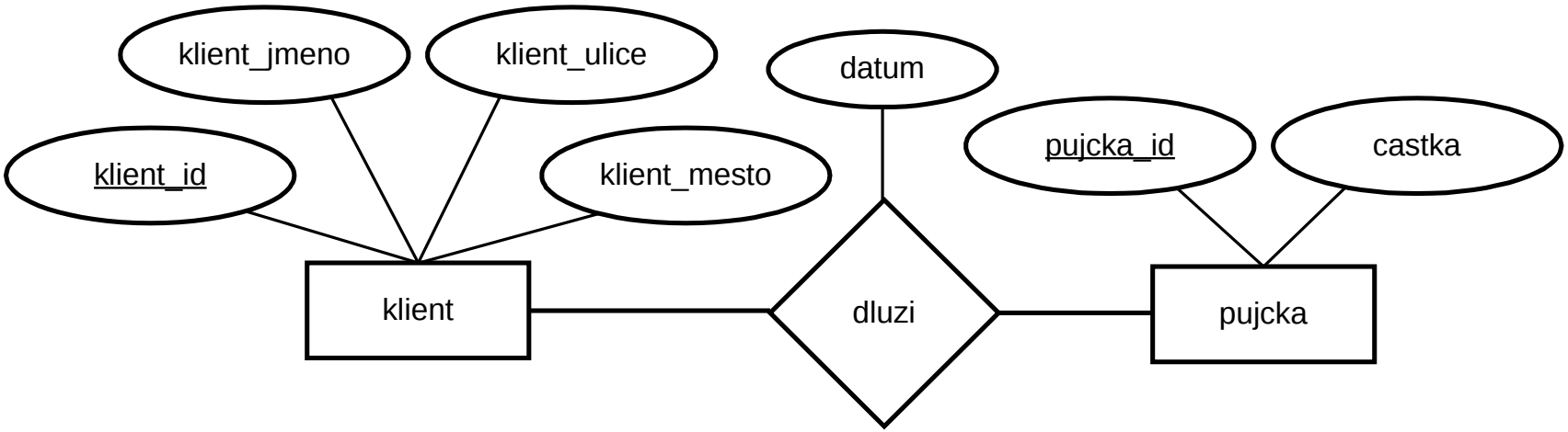
Poznámka: V množinách entit mohou existovat i prvky, které nejsou propojeny s žádným prvkem v druhé množině.

Klíče

- Klíčem množiny entit je atribut nebo skupina atributů, jejichž hodnota určuje jednoznačně konkrétní entitu
  - Takových skupin atributů může být více – někdy se říká „superklíč“
  - Minimální superklíče jsou kandidáti na to, aby se stali klíčem
  - Mezi potenciálními kandidáty bude zvolen jeden primární klíč
    - Např. *klient_id* bude zvolen za primární klíč množiny entit *klient*
  - Někdy se zavádí samostatný (syntetický) atribut, aby sloužil jako klíč
- Superklíčem množiny vztahů je kombinace (zřetězení) primárních klíčů participujících množin entit
  - též značené jako „složkové klíče“
    - (*klient_id*, *pujcka_id*) je superklíčem vztahu *dluzi*
    - Pak ovšem nelze mít ve vztahu *dluzi* více údajů o splátkách půjčky
      - Logickou strukturu pro tato data je nutno navrhnout jinak
  - Volba kandidátů a výběr primárního klíče vztahu závisí na kardinalitě vztahu
    - Pro 1:1 může být primárním klíčem kterýkoliv složkový klíč
    - Pro ostatní případy „zřetězení“ složkových klíčů

Přehled hlavních symbolů E-R diagramů

E-R Diagramy



- Obdélníky – množiny entit
- Kosočtverce – množiny vztahů
- Ovály – atributy
  - zdvojené ovály se užívají pro vícehodnotové atributy
  - čárkované ovály značí odvozené (počítané) atributy
- Podtržené atributy značí primární klíče

E-R diagram s vícehodnotovými, složenými a odvozenými atributy

Role

- Vztahy nemusí propojovat různé množiny entit
  - Pak je vhodné zavést role a označit tak význam částí vztahu
  - Označení "*řídí*" a "*je_řízen*" specifikují, jak jsou jednotliví zaměstnanci vzájemně podřízeni přes vztah *pracuje_pro*
  - Role v ER diagramech jsou nepovinné, zlepšují čitelnost a vyjadřují sémantiku

Vyjádření kardinality vztahů

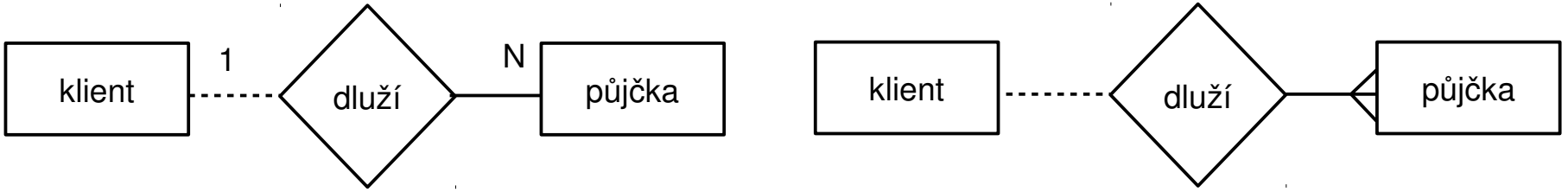
- Značení je nejednotné
- Základní metoda:
  - Pokud se všechny entity z dané množiny musí účastnit množiny vztahů, znázorňuje se to tlustou nebo dvojitou čarou a nazývá se to omezení účasti ve vztahu (*participation constraint*).
  - Pokud se každá entita z množiny může účastnit maximálně jednoho vztahu z množiny, je to znázorněno šipkou od množiny entit k množině vztahů, je to nazýváno klíčové omezení (*key constraint*).
  - Ke znázornění vztahu, kdy každá entita z množiny se účastní právě jednoho vztahu z množiny, se používá tlustá šipka.

Kardinalita a omezení - příklady

- Každý klient má právě jednu půjčku
- Několik klientů má jednu společnou půjčku
- Jeden klient má několik půjček
  (nebo také žádnou)
- Několik klientů participuje na několika půjčkách
- Ke každé půjčce musí existovat aspoň jeden klient
  - avšak ne každý klient musí mít půjčku

Alternativní vyjádření omezení kardinalitou vztahů

Klient může mít libovolný počet půjček (nebo také žádnou), avšak každá existující půjčka musí mít svého klienta



- Alternativně se též používá značení Crow‘s feet

Otázky návrhu modelu

- Entita nebo atribut?
  - Volba závisí na struktuře modelované reality a na sémantice příslušného atributu
  - Často entita místo vícehodnotového nebo složeného atributu
- Použít množinu entit nebo množinu vztahů?
  - Možným vodítkem je právě význam: vztahy jsou slovesné fráze popisující akce, které se dějí mezi entitami
    - Např. *klient* ukládá_na *účet*
- Binární nebo *n*-ární množiny vztahů?
  - Kvůli zlepšení čitelnosti jsou často vhodnější *n*-ární množiny vztahů, neboť je zřejmé, že více entit participuje na jednom vztahu
    - Z implementačního pohledu jsou binární vztahy efektivnější
    - *n*-ární vztahy lze převést na sadu binárních ➔
- Připojovat atributy k množinám vztahů?
  - Připojení jednoduchého atributu k množině vztahů je sice efektivní, může však způsobit komplikace při modifikaci modelu
    - složitější atributy ke vztahům jsou nevhodné

Binární vs. *n*-ární vztahy

- Některé vztahy vypadající „ne-binárně“ je lépe reprezentovat binárními vztahy
  - Příklad: Ternární vztah *rodiče* propojující dítě s jeho matkou a otcem je vhodnější nahradit dvojicí binárních vztahů *je_otcem* a *je_matkou*
    - To dokonce přinese výhodu reprezentace možnosti zakotvené již ve starořímském právu: „Jistá je vždy jen matka“
  - Existují ale vztahy, které jsou ze své sémantiky ne-binární
    - Např.: Máme množiny entit:

      studentske_projekty
      vedouci_ucitele a
      studenti
      a k nim množinu vztahů *pracuje_na_a_vede_ho*, které popisují, který student pracuje na kterém projektu pod vedením kterého učitele.

      Kdybychom nahradili takovýto ternární vztah binárními vztahy *ucitel_projekt* a *ucitel_student*, tak sice zachováme informaci o tom, že učitel Novák vede studenty Karla a Petra a že učitel Novák řídí projekty A a B, ale nebudeme mít informaci o tom, že Karel řeší projekt A a Petr řeší projekt B

Formální převod *n*-árních vztahů na binární

- Ne-binární vztahy lze reprezentovat binárními
  - Zavedeme abstraktní množinu entit E a nahradíme vztahy R mezi množinami entit A, B a C množinou entit E a třemi množinami vztahů:
    1. R_A, které propojují E a A
    2. R_B, propojují E a B
    3. R_C, propojují E a C
  - Vytvoříme klíč pro množinu E a případné atributy vztahů v R přesuneme do E
  - Pro každý jeden vztah (a_i , b_i , c_i) v R vytvoříme
    1. novou entitu e_i v množině E
    2. přidáme (e_i , a_i) do R_A
    3. přidáme (e_i , b_i) do R_B
    4. přidáme (e_i , c_i) do R_C
- Při tomto převodu mohou nastat problémy s omezeními souvisejícími s kardinalitou

Slabé množiny entit

- Při modelování reality se někdy vytváří množiny entit, které samy o sobě nemají smysl
  - Jsou součástí celku daného souvislostí s jinou množinou entit. Takové množiny entit se nazývají slabé (též slabý entitní typ)
  - Existence slabé množiny entit závisí na existenci identifikující množiny entit
    - Musí existovat vztah 1:*N* vedoucí z identifikující ke slabé množině entit
    - Identifikující vztah se označuje v E-R diagramu zdvojeným kosočtvercem
  - Primární klíč slabého entitního typu primární klíč identifikující množiny plus vhodný rozlišující atribut slabé množiny (tzv. diskriminátor či „rozlišovač“)
    - Slabé množiny entit se značí zdvojeným obdélníkem a její diskriminátor se podtrhává čárkovaně
    - Příkladem slabé entity jsou „položky faktury“, které samy o sobě nemají smysl, pokud nejsou vztaženy ke konkrétní faktuře. Identifikující množinou budou „faktury“ a diskriminátorem „číslo položky“
    - Jiný příklad

Přehled hlavních symbolů E-R diagramů

Převod E-R modelu na logické schéma

- Data uložená v tabulkách (relace ➔)
- Množina entit
  - Pojmenovaná tabulka
    - Pojmenované sloupce jsou atributy, každý atribut má svůj datový typ (datum, řetězec znaků dané délky, číslo integer, ...)
    - Řádky – jednotlivé entity (instance)
  - K prohledávání tabulky slouží primární klíč
- Množina vztahů
  - Pojmenovaná tabulka
    - Sloupce jsou primární klíče entitních tabulek, které jsou vztahem propojeny; mohou být přidány další sloupce popisující atributy vztahu
    - Řádky – jednotlivé dvojice (nebo *n*-tice) provázaných entit
  - Primárním klíč závisí na kardinalitě vztahu
    - pro vztah 1:1 stačí jediný atribut, jinak „zřetězení“ primárních klíčů propojených entit
- Slabá množina entit
  - Opět tabulka se sloupcem obsahujícím primární klíč identifikující entity a sloupcem obsahujícím diskriminátor
    - Zřetězení těchto dvou atributů je primárním klíčem slabého entitního typu

Převod E-R modelu na logické schéma (pokr.)

- Složené atributy
  - Zpravidla se převedou na skupinu jednoduchých atributů (několik sloupců v tabulce)
    - Např. Složený atribut *jmeno* se složkami *prvni_jmeno* a *prijmeni* se převede na dvojici jednoduchych atributů *jmeno.prvni_jmeno* a *jmeno.prijmeni*
- Vícehodnotové atributy
  - Pro takový atribut se vytvoří samostatná tabulka s dvěma sloupci (atributy)
    - První sloupec je primární klíč základní množiny entit a
    - druhý sloupec je konkrétní hodnota zobrazovaného vícehodnotového atributu

klient_id	telefon
12-345	602602602
12-345	224224224
12-345	776776776

Tabulka atributu *klient.telefon*

- Takto dekomponované tabulky s atomickými atributy tvoří schéma v tzv. první normální formě
  - Skvělý výklad je na http://en.wikipedia.org/wiki/First_normal_form

Kvalita schématu

- Uvažujme relaci:

  Vyrábí(Zaměstnanec_jméno, Výrobek_jméno, Výrobek_hmotnost, počet_kusů)
- Aktualizační anomálie (Codd)
  - Změní-li se hmotnost výrobku, pak je nutné ji měnit na více místech (pokud se neprovede úprava všude, pak jsou rozporuplné údaje v databízi)
  - Pokud se výrobek zrovna nikde nevyrábí, ztrácí se informace o jméně a hmotnosti výrobku
  - Pokud chcete přidat výrobek do databáze, lze to provést pouze když se výrobek bude vyrábět
- Jak to lze vyřešit?
  - Normalizace dekompozicí

    Vyrábí(Zaměstnanec_jméno, Výrobek_jméno, počet_kusů)

    Výrobky(Výrobek_jméno, Výrobek_hmotnost)
  - Dekompozicí jsme se zbavili aktualizačních anomálií

Normalizace databází

- Je to teorie umožňující rigorózní návrh databáze s ohledem na její korektní aktualizace a užití (Edgar F. Codd 1971)
- První normální forma 1NF – všechna data v relaci musí být atomická
  - Špatně atribut telefon s hodnotou 123456789,987654321
- 2NF Problém: Co zvolit jako klíč?
  - Ani jeden z atributů není klíčem => klíčem musí být zřetězení atributů
    - Nevhodně a redundantně se opakují data
- Příklad:

Zamestnanec	Dovednost	Pracoviště
Novák	Programátor Java	Spálená 22, Praha 1
Novák	Programátor .net	Spálená 22, Praha 1
Novák	Data architekt	Spálená 22, Praha 1
Kovářová	Programátor .net	Hlavní 10, Brno
Kovářová	Programátor PHP	Hlavní 10, Brno
Mates	Tester	Horní 135, Benešov

- Druhá normální forma
  - Rozklad tak, aby klíče byly jednoduché atributy
    - Avšak i nadále trvá problém: Když se Kovářová vdá, budou se měnit klíče a na ty mohou být navázány další vztahy.

Zamestnanec	Dovednost
Novák	Programátor Java
Novák	Programátor .net
Novák	Data architekt

- Další argumenty viz http://en.wikipedia.org/wiki/Second_normal_form

Funkční závislosti 3NF-5NF

- Další normální formy jsou založeny na tzv. funkčních závislostech:
  - Funkční závislostí rozumíme vazbu (omezení) platnou mezi dvěma množinami atributů v téže "tabulce" (relaci ➔) v databázi
    - Primitivní příklad: Rodne_cislo → Datum_narozeni (datum narození lze z rodného čísla odvodit). Nikoli však obráceně!
    - Podmínka, z níž lze odvodit závěr, se nazývá determinant závislosti
- Příklady:

Student_ID	Semestr	Předmět	Učitel
1234	6	Numerické metody	Jan
2380	4	Numerické metody	Petr
1234	6	Elektronické systémy	Alena
1201	4	Numerické metody	Petr
1201	4	Fyzika 2	Josef

  - Zde lze odvodit následující "funkční" závislosti:
    - Triviální Student_ID → Semestr (konkrétní student studuje konkrétní semestr)
    - Netriviální závislosti (méně zjevné):
      - {Student_ID, Předmět} → Učitel
      - {Student_ID, Předmět} → {Učitel, Semestr}
      - Z poslední závislosti vyplývá, že {Student_ID, Předmět} je superklíčem
- K čemu je to dobré?

Boyce-Coddova normální forma (BCNF)

- Definice BCNF:
  - Relace *R* je v BCNF právě tehdy, když pro každou netriviální závislost $X \rightarrow Y$, kde *X* a *Y* jsou množiny atributů a zároveň *Y* není podmnožinou *X*, platí, že *X* je nadmnožinou nějakého klíče, nebo *X* je klíčem relace *R*.
    - Jinak řečeno relace *R* je v BCNF tehdy a jen tehdy, když každý determinant funkční závislosti v relaci *R* je zároveň kandidátním klíčem relace *R*.
- Tabulky (relace) v BCNF umožňují jednoznačně odpovídat na "datové dotazy" (➔)
  - Např.:
    je v BCNF

Předmět	Učitel	Učebnice
Databáze	Jiří	DB Concepts
Databáze	Jiří	Ullman
Databáze	Petr	DB Concepts
Databáze	Petr	Ullman
Operační systémy	Petr	OS Concepts
Operační systémy	Petr	Tannenbaum MOS
Operační systémy	Jiří	OS Concepts
Operační systémy	Jiří	Tannenbaum MOS

  - Dotaz: "Kdo používá učebnici Tannenbaum MOS?" je jednoznačně zodpověditelný

BCNF a další normalizace

- Bohužel tabulky v BCNF trpí mnohdy tzv. aktualizačními problémy.
  - Přijde-li nový učitel databází Franta a bude používat tytéž dvě učebnice, bude nutno doplnit dva záznamy.
  - Bude proto vhodné rozložit naši tabulku na dvě:

Předmět	Učebnice
Databáze	DB Concepts
Databáze	Ullman
Operační systémy	OS Concepts
Operační systémy	Tannenbaum

Předmět	Učitel
Databáze	Jiří
Databáze	Petr
Operační systémy	Petr
Operační systémy	Jiří

  - To pak vede na tzv. 4. NF – složený primární klíč nesmí být tvořen z nezávislých dat,
  - I pokud databáze splňuje 4NF můžeme najít další drobné problémy.
- Závěr:
  - Teorie normalizace databází je matematicky hluboce propracovaná
  - V učebnicích lze najít devět stupňů normálních forem
  - Zájemci nechť se obrátí ke specializovaných publikacím nebo k předmětům, které se věnují výhradně databázím a jejich teorii
    - Např. A4B33DS, A7B36DBS apod.

Dotazy