Úvod do databází

Tradiční a relativně nejčastější použití počítačů je v oblasti zpracování dat, dříve také nazývané hromadným zpracováním dat (HZD). Rozsáhlejší systémy pro zpracování dat se nazývají informační systémy. Rozumíme jimi systémy pro sběr, uchování, vyhledání a zpracování dat za účelem poskytnutí informací. Data jsou údaje získané pozorováním nebo měřením a informace interpretací těchto dat a vztahů mezi nimi. Informační systémy zajišťují následující činnosti:

- výběr informace,
- prognózy vývoje,
- plánování,
- rozhodování,
- použití pro automatizaci inženýrských prací (AIP),
- použití pro zpracování ekonomických agend (mzdy, faktury, skladové hospodářství, účetnictví apod}.

Činnost informačního systému může probíhat jako

- zpracování v dávkách,
- konverzace se systémem.

První způsob je vhodný zvláště pro soubory, kde se zpracování účastní všechny nebo většina záznamů jako např. mzdová agenda. V přípravě podkladů je ovšem potřeba rozsáhlá administrativa a interval mezi počátkem zpracování a časem, kdy jsou připraveny výstupní sestavy, může být dosti dlouhý. Konverzační systémy byly původně vytvořeny pro aplikace vyžadující rychlou zpětnou vazbu jako je např. rezervace letenek, zpracování dat v bankovnictví apod. Výhodou je to, že vstup dat se provádí přímo tam, kde data vznikají, a odpadá tak mnoho administrativy a kódování, které je typické pro dávkové zpracování.

Objekty (lidi, zvířata, věci, jevy) obvykle popisujeme pomocí jejich **vlastností** *(zaměstnanec firmy má jméno, adresu, funkci, plat; kniha v knihovně má název, autora, rok vydání, cenu*…).

Při evidencích se pak předem rozhodneme, které vlastnosti potřebujeme sledovat. Vybrané vlastnosti budeme nazývat **atribut**y.

Daty nazýváme údaje získané měřením, pozorováním nebo jen pouhým zaznamenáním z reálné skutečnosti. **Informace** jsou smysluplné interpretace dat a vztah ů mezi nimi. **Zpracováním dat** nebo také hromadným zpracováním dat nazýváme zpracování velkého množství údaj ů (obvykle desítky až stovky) o velkém množství objektů (obvykle od desítek po miliony i víc).

Vést evidenci o objektech znamená

- zaznamenat vhodně organizované údaje na nějaké médium;
- provádět změny údajů při změně evidované reality;
- provádět výběry informací podle různých kritérií;
- odvozovat a počítat z uložených údajů další;
- třídit údaje dle různých kritérií;
- zaznamenávat vztahy mezi údaji o objektech různých druhů ;

Informačním systémem obecně nazýváme organizaci údajů vhodnou pro systémové zpracování dat: pro jejich sběr, uložení a uchování, zpracování, vyhledávání a vydávání informací o nich, to vše pro účely rozhodování.

Vlastnosti databázové technologie:

1. Paradigma databázové technologie – **oddělení datových struktur od programů**.
2. Součástí SŘBD je soubor prostředků, pomocí nichž se datové struktury definují a který nazýváme **jazykem pro definici dat** (JDD).
3. Existuje soubor instrukcí, které nad definovanými daty provádějí jednotlivé operace; každá instrukce je vlastně mohutnou procedurou, v níž je řešen fyzický přístup k datům i realizace vlastní operace; jinak než prostřednictvím systému není možno s daty pracovat. Tento soubor instrukcí nazýváme **jazykem pro manipulaci s daty** (JMD).
4. SŘBD řeší způsob, jak zaznamenat **vztahy mezi objekt**y.
5. Data je možno zpracovávat libovolným i předem nepředpokládaným způsobem. Pro zodpovězení dotazů náhodných uživatelů je v SŘBD další typ jazyka – **dotazovací jazy**k. Ten umožňuje formulovat většinu dotazů na informace v databázi uložené bez nutnosti psát program pro jejich vyhledání.

6. SŘBD umožňuje **víceuživatelský přístup** k informacím
7. SŘBD umožňuje **ochranu dat** před zneužitím použitím hesel, definováním přístupových práv na úrovni souborů, záznamů, položek, rozlišením práv pro zápis, čtení, modifikace. Tak ani programátor, znalý struktury dat, nemá přístup k reálným datům.

Uživatelé databázové technologie

Se SŘBD pracuje na různých úrovních několik typů uživatelů, dělících se podle způsobu komunikace s databází:

- **Správce nebo administrátor báze dat** je profesionální analytik a systémový programátor, který rozhoduje o tom, která data a jak budou v bázi uložena.
- **Aplikační programátor** je profesionální programátor, který programuje aplikační úlohy nad definovanými datovými strukturami pomocí programových prostředků SŘBD. Nemusí znát strukturu celé databáze, stačí mu znalost struktur, se kterými bude pracovat a které mu zadá správce
- **Příležitostný uživatel** je jakýkoliv uživatel, který umí prostřednictvím dotazovacího jazyka formulovat svůj dotaz.
- **Naivní uživatel** je takový uživatel (obvykle neprogramátor), který prostřednictvím aplikačních programů pracuje s databází a používá tak databázi jako informační systém pro ukládání, zpracování a vyhledávání informací.

.

Entitou rozumíme libovolnou existující osobu, zvíře, věc či jev (obecně objekt) reálného světa. Entita musí být rozlišitelná od ostatních entit a existovat nezávisle na nich.

Atribut je charakteristika, vlastnost entity, údaj o objektu. Atribut přiřadí každé entitě z množiny entit hodnotu z nějaké neprázdné množiny hodnot, nazvané **doména atributu** (obor hodnot atributu). Atribut je tedy zobrazení množiny entit do domény atributu. Je zadán svým názvem (identifikátorem) a datovým typem. **Typem entity** nazýváme množinu objektů stejného typu, charakterizovaných názvem typu a strukturou jejich atributů. Jednotlivé entity nazýváme také výskyty nebo **instancemi** objektů entitního typu. Instance entity je tedy konkrétní n-tice hodnot atribut ů jedné konkrétní entity.

Jeden atribut nebo množinu atributů, které jednoznačně určují entitu v množině entit, nazveme **klíčovým atributem.** Kandidátů na klíčový atribut může být mezi atributy.

Stupně tvorby databáze

- *Fáze shromažďování požadavků*. Bez ní můžeme opomenout mnohé důležité stránky budoucí databáze. Požadavky získáme především pomocí rozhovorů se zadavatelem vytvoření databáze a s budoucími uživateli databáze: v našem případě např. s učitelem, který chce knihovnu používat.
- *Fáze analýzy požadavků, vznik konceptuálního modelu, který znázorníme ER diagramem*. Analýzou požadavků se snažíme najít základní součásti (entity) budoucí databáze (částečně odpovídají budoucím tabulkám), vlastnosti entit (neboli atributy, budoucí sloupce tabulek) a také vazby mezi entitami (ze kterých vzniknou vazby mezi tabulkami).
- *Fáze tvorby logického modelu* – v této fázi přejdeme od ER diagramu k vlastním tabulkám databáze a k vazbám mezi nimi. Návrh ještě nezávisí na použitém SŘBD.
- *Fáze tvorby fyzického modelu* – zefektivnění a přizpůsobení použitému SŘBD. Touto fází se v tomto seriálu nebudu zabývat.
- *Fáze testování, tvorby dokumentace apod.* – závěrečné činnosti, ani jim se zde nebudeme věnovat.

Příklad:

Vytvořte databázi Knihovna s následujícími vlastnostmi:

1. Databáze uchová ke knize údaj o názvu, oboru, autorovi knihy, roku vydání
2. Databáze uchová o autorovi jen jméno a příjmení
3. Databáze umožní zadat několik oborů, při zadávání knih do databáze bude zadavatel moci z těchto oborů vybírat čili kniha má jeden (nebo více) obor
4. S databází bude pracovat zadavatel (zapisuje nové knihy, zapisuje výpůjčky, návrat knihy, z databáze odstraňuje odepsané knihy, tiskne seznamy knih, tiskne upomínky, seznamy čtenářů s upomínkou, seznamy knih, které má půjčené čtenář, může nastavit interval výpůjčky – např. jeden měsíc)
5. Záznam o výpůjčce bude obsahovat jméno, příjmení, u studenta třídu, datum výpůjčky, datum předpokládaného vrácení (vygeneruje se samo dle nastavení)
6. Databáze umožní při zadávání knih zadavateli vybírat z autorů, popř. dopsat nového autora
7. Databáze umožní, že kniha může mít více autorů (a samozřejmě autor může mít více knih)

8. Databáze umožní, že kniha může mít více oborů (a samozřejmě daný obor může být přiřazen více knihám)
9. Databáze umožní zobrazení nebo výpis knih podle oborů, podle data vydání, podle autorů, podle abecedy – názvů
10. Databáze umožní zobrazit nebo i vypsat záznam výpůjček knih vybraného půjčovatele (třídy, ze které vypůjčovatel je, knihy, data výpůjčky a data, kdy skončila regulérní výpůjční lhůta)
11. Databáze umožní zobrazit všechny vypůjčovatele z jedné třídy
12. Databáze umožní, aby se třída vypůjčovatele v novém školním roce zvýšila, umožní zobrazení nebo výpis studentů, kteří budou v daném školním roce končit studium
13. Databáze umožní nastavit výpůjční dobu
14. Databáze umožní výpis těch vypůjčovatelů, kteří výpůjční dobu překročili
15. Databáze zaznamená historii upomínek každého čtenáře
16. Záznam o upomínce bude obsahovat jméno a příjmení čtenáře, název knihy, datum výpůjčky, datum vrácení a dobu překročení termínu výpůjčky

Fáze analýzy požadavků

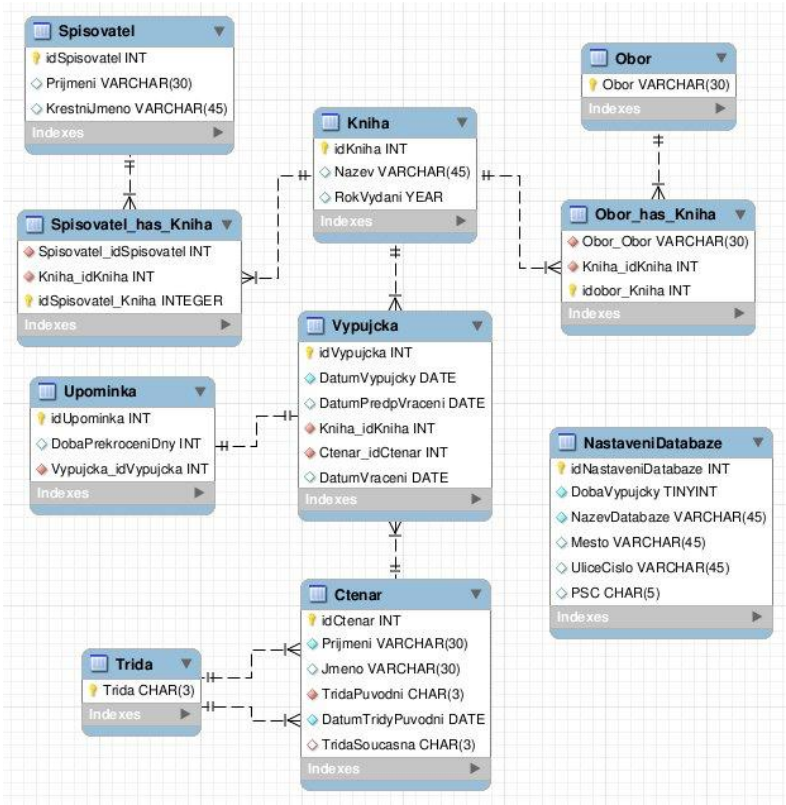
Nejdřív vyznačíme v textu zvýrazněním vhodných podstatných jmen možné *entity* (budoucí tabulky) a jejich*vlastnosti* (atributy, budoucí sloupce tabulek). Slova, která by mohla představovat entity, označím tučným písmem, atributy těchto entit značím tučnou kurzivou.

1. Databáze uchová ke **knize** údaj o ***názvu***, ***oboru***, **autorovi** knihy, ***roku vydání***
2. Databáze uchová o **autorovi** jen ***jméno*** a ***příjmení***
3. Databáze umožní zadat několik **oborů**, při zadávání knih do databáze bude zadavatel moci z těchto oborů vybírat
4. S databází bude pracovat zadavatel (zapisuje nové **knihy**, zapisuje **výpůjčky**, návrat knihy, z databáze odstraňuje odepsané knihy, tiskne seznamy knih, tiskne **upomínky**, seznamy **čtenářů** s**upomínkou**, seznamy knih, které má půjčené čtenář, může nastavit ***interval výpůjčky*** – např. jeden měsíc)
5. Záznam o **výpůjčce** bude obsahovat ***jméno***, ***příjmení***, u studenta ***třídu***, ***datum výpůjčky***, ***datum předpokládaného vrácení*** (vygeneruje se samo dle nastavení)
6. Databáze umožní při zadávání knih zadavateli vybírat z **autorů**, popř. nového autora dopsat
7. Databáze umožní, aby **kniha** mohla mít více **autorů** (a samozřejmě aby **autor** mohl mít více **knih**)

8. Databáze umožní, aby **kniha** mohla mít více **oborů** (a samozřejmě daný **obor** může být přiřazen více **knihám**)
9. Databáze umožní zobrazení nebo výpis knih podle oborů, podle ***data vydání***, podle autorů, podle abecedy – názvů
10. Databáze umožní zobrazit nebo i vypsat záznam **výpůjček** knih vybraného **půjčovatele** (***třídy***, ze které vypůjčovatel je, ***knihy***, ***data výpůjčky*** a ***data, kdy skončila regulérní výpůjční lhůta***)
11. Databáze umožní zobrazit všechny **vypůjčovatele** z jedné ***třídy***
12. Databáze umožní, aby se třída **vypůjčovatele** v novém školním roce zvýšila, umožní zobrazení nebo výpis **studentů**, kteří budou v daném školním roce končit studium
13. Databáze umožní nastavit ***výpůjční dobu***
14. Databáze umožní výpis těch **vypůjčovatelů**, kteří výpůjční dobu překročili
15. Databáze zaznamená historii **upomínek** každého čtenáře
16. Záznam o **upomínce** bude obsahovat ***jméno*** a ***příjmení*** **čtenáře**, ***název knihy***, ***datum výpůjčky***, ***datum vrácení*** a ***dobu překročení*** termínu **výpůjčky**

ER diagram

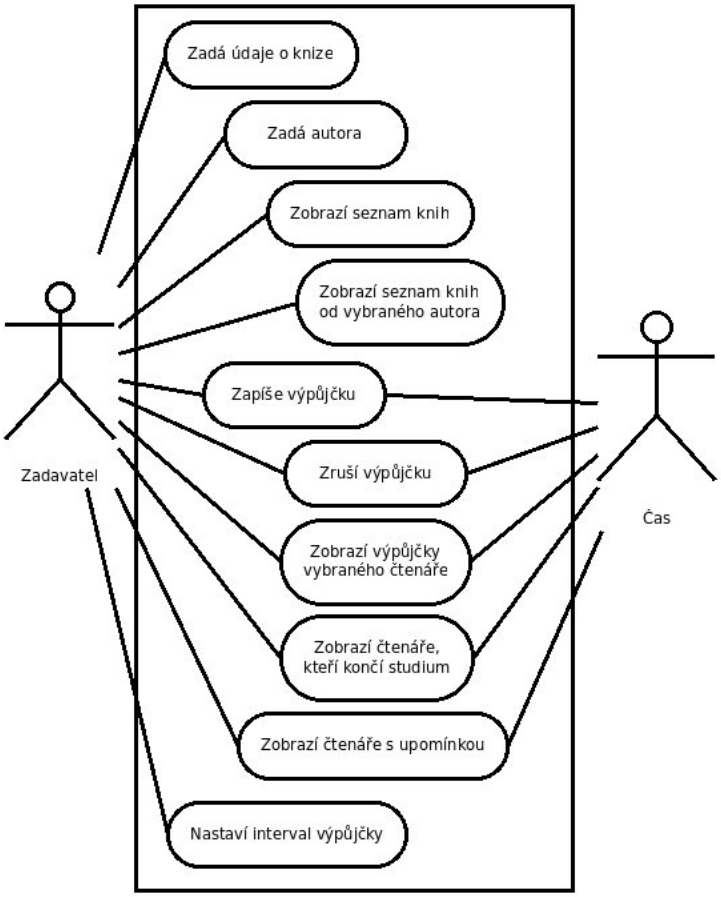
Výsledkem analýzy požadavků (entit, atributů a vazeb) bude nový ER (entitně – relační) diagram. Mohl by vypadat následovně:



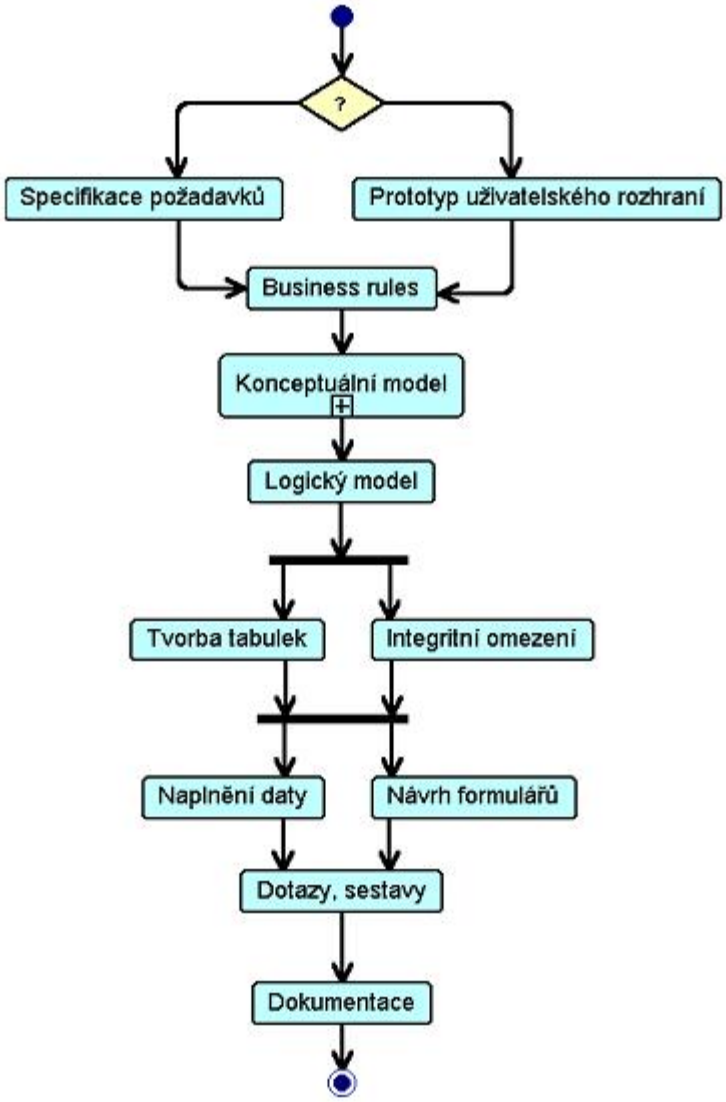
V diagramu se už setkáváme s řešením problému vazby M – M mezi dvěma tabulkami. Tato vazba se uskuteční rozdělením na dvě vazby 1 – M s pomocí třetí tabulky, vložené mezi dvě původní. Takovou pomocnou tabulkou je Spisovatel_has_Kniha mezi tabulkami Kniha a Spisovatel a tabulka Obor_has_Kniha mezi tabulkami Kniha a Obor. Poslední tabulky tohoto typu v diagramu se týká otázka na konci dílu.

Diagram případů užití (Use case diagram)

Analýze požadavků může pomoci také grafické znázornění činností, které budou uživatelé (tzv. aktéři,actors) provádět. Více pohledů na požadavky umožňuje totiž jejich vzájemné opakované vylepšování a doplňování. Zde uvádím tzv. diagram případů užití (Use Case Diagram), všimněte si, že jako aktér může být označen i čas, pokud nějaká činnost v systému má nastat v jistém čase. Diagram jistě není úplný, ale mohl by nás přivést k vylepšení seznamu požadavků a naopak v průběhu shromažďování požadavků bychom mohli doplňovat tento diagram.



Metodika návrhu database



Základy jazyka SQL

SQL (Structured Query Language)

- strukturovaný dotazovací jazyk pro výběr dat z databáze
- je standardizovaný, může se použít v různých databázových systémech téměř stejně
- je to vnitřní nástroj, který databáze používá, jeho příkazy jsou pro běžného uživatele skryté za obvykle za webovým formulářem

Příkazy pro práci s databází

- vytvořit tabulku, vyprázdnit tabulku, zrušit tabulku, změnit tabulku
- vložit data do tabulky, změnit data v tabulce, vymazat data z tabulky, zobrazit data z tabulky

Příkaz SELECT

- příkaz pro výběr dat z databáze
- struktura: **SELECT** názvy polí **FROM** názvy tabulkek **WHERE** podmínky pro vybraná data **ORDER BY** řazení dat
- popř.: **SELECT** názvy polí **FROM** názvy tabulkek **AS** alias pole **WHERE** podmínky pro vybraná data **ORDER BY** řazení dat

Názvy polí

- pokud chceme vypsat **hodnoty všech polí,** nevypisujeme za klíčové slovo FROM všechny názvy atributů, ale použijeme symbol *
  Př.: vypište všechny údaje z tabulky auta, která byla vyrobena v roce 2001:
  SELECT * **FROM** auta **WHERE** rok_vyroby = 2001
- pokud vypisujeme data z více provázaných tabulek, které mají některé názvy polí (atributů) stejné, musíme před název atributu uvést název tabulky oddělený tečkou:
  Př.: Vypište jména klientů, názvy knížek a jména autorů, těch klientů, kteří si půjčili knížky před 1. květnem 2010:
  SELECT klienti.jmeno, knihy.nazev, knihy.jmeno **FROM** klienti,
  knihy **WHERE** datum_vypujcky < 2010-05-01 AND knihy.klient_ID = klienti.ID

Agregační funkce

- **AVG** - aritmetický průměr
  průměrný plat technického pracovníka:
  SELECT AVG(plat) FROM klienti WHERE zarazeni = "technický pracovník"
- **COUNT** - počet hodnot ve sloupci
  počet technických pracovníků:
  SELECT COUNT(*) FROM klienti WHERE zarazeni = "technický pracovník"
- **MAX** - maximální hodnota z množiny
  maximální plat technického pracovníka:
  SELECT MAX(plat) FROM klienti WHERE zarazeni = "technický pracovník"
- **MIN** - minimální hodnota z množiny
- **SUM** - součet hodnot
  celkové mzdové náklady na technické pracovníky:
  SELECT SUM(plat) FROM klienti WHERE zarazeni = "technický pracovník"

Modifikátory agregačních funkcí

- **DISTINCT** - z množiny jsou hodnoty vybrány tak, aby se žádná neopakovala
  počet různých titulů v knihovně:
  SELECT COUNT(DISTINCT knihy.nazev) FROM knihy
- **ALL** - z množiny jsou vybrány všechny hodnoty, bez ohledu na to, jestli se vyskytují vícekrát
  SELECT FROM WHERE