1 DATABÁZE

1.1 ÚVOD

Moderní doba se databázemi jen hemží. Například: Management firmy musí mít přehled přinejmenším o svých zaměstnancích, výrobcích, majetku a zákaznících. Přitom mezi těmito údaji existují různé vztahy, třeba zaměstnanci mají přidělen firemní automobil k používání nebo zákazníci si objednávají různé výrobky atd.. Manažeři firmy potřebují vědět, který automobil je momentálně k dispozici, o který výrobek není mezi zákazníky zájem, který zaměstnanec má nejvyšší plat a podobně. Je vidět, že je nutné uspořádat veškerá data do databáze tak, aby z ní šly efektivně získávat správné informace. Co je tedy databáze?

Databáze je sada vzájemně souvisejících dat, s nimiž pracujeme jako s ucelenou jednotkou.

Úkoly: Uveďte příklady použití databází. (Třeba školní databáze pro evidenci studentů, tříd, učitelů, klasifikace – Bakaláři)

Databáze vzniká v několika krocích:

Návrhář databáze důkladně pozná požadavky uživatele a navrhne databázi.

Databázový administrátor návrh realizuje v databázovém systému.

Aplikační programátor doprogramuje příjemné uživatelské prostředí pro uživatele, vznikne databázová aplikace.

My a databáze: Tentokrát nebudeme uživateli databáze, budeme jejími návrháři a administrátory, připravíme jednoduchou databázi pro koncového uživatele.

1.2 ZÁKLADNÍ POJMY RELAČNÍ DATABÁZE

Během posledního půl století se z databází stala věda. Vystřídalo se několik databázových modelů, **nejběžnějším modelem** současnosti se staly **relační databáze**.

Abychom si rozuměli, uvedu **základní pojmy** relačních databází:

Data jsou logicky uspořádána do tzv. **relací** – reprezentovaných tabulkami.

Jedna tabulka se týká pouze jedné **entity**.

Entita je osoba, místo, věc, událost nebo myšlenka o které shromažďujeme nějaká data (např. zaměstnanci, knihy, objednávky, výrobky...).

Jednu entitu charakterizují **atributy** (reprezentované sloupci tabulky).

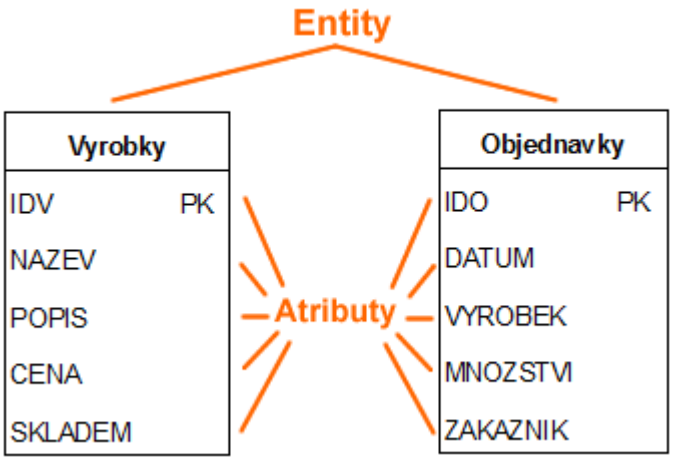
Atribut je jednotka informace popisující entitu (plat zaměstnance, cena výrobku, autor knihy, adresa zákazníka, rodné číslo studenta...)

Každá tabulka musí mít atribut, který jednoznačně identifikuje jednotlivé záznamy - řádky tabulky. Tento atribut se nazývá **primární klíč** (PK) (obvykle nějaké identifikační číslo).

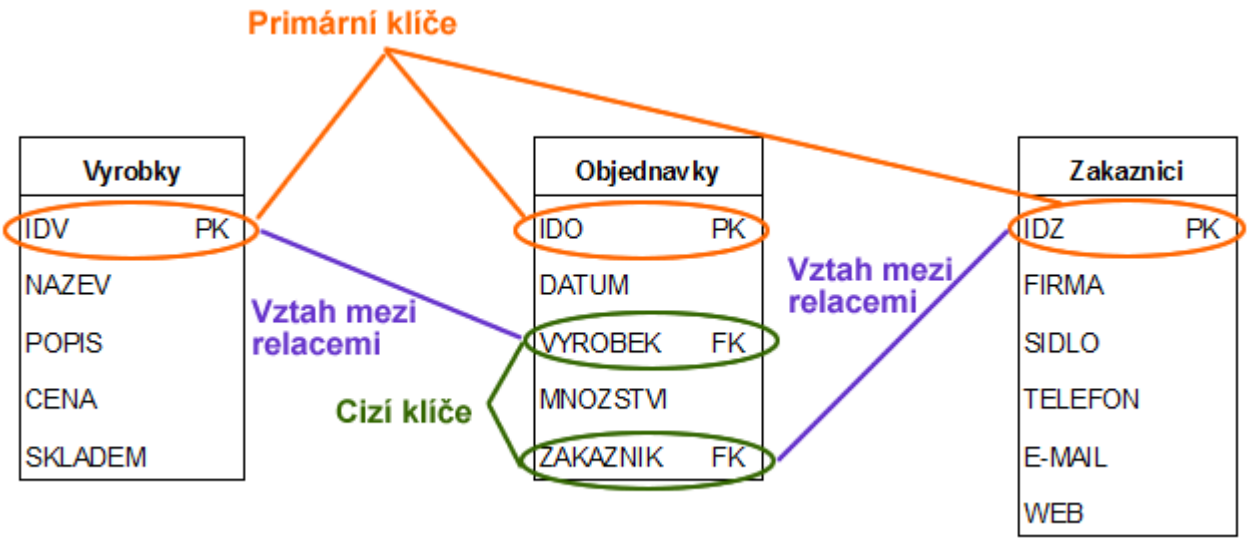
Mezi relacemi (tabulkami) existují vztahy (bohužel též nazývané relace), které reprezentují **cizí klíče**.

Cizí klíč (FK – Foreign key) je atribut v tabulce, který tvoří primární klíč v jiné tabulce.

Zadáme-li do databáze vztahy mezi tabulkami pomocí cizích klíčů, zamezíme tím uživateli zadávat do databáze nesprávná data (tomu se říká **referenční omezení**). Např. mezi entitami *Vyrobky* a *Objednavky* existuje vztah – do objednávky se zadává číslo objednávaného výrobku, což je primární klíč v entitě *Vyrobky* a zároveň cizí klíč v entitě *Objednavky*. Uvedený vztah pak zamezí uživateli zadat do objednávky nesprávné číslo výrobku.



Entity Vyrobky a Objednavky s atributy



Vztahy mezi entitami

Úkoly: Navrhněte pět entit, u každé z nich určete několik vhodných atributů včetně primárního klíče.

Úkoly: Navrhněte dvě dvojice (trojice) entit včetně vhodných atributů, mezi nimiž existuje vztah. Zdůrazněte primární a cizí klíče.

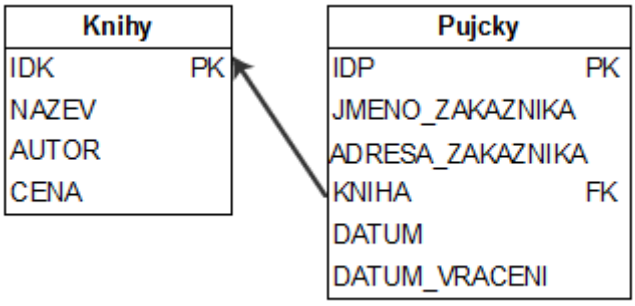
1.3 NÁVRH RELAČNÍ DATABÁZE

Sestavíme návrh jednoduché databáze pro knihovnu.

Knihovník potřebuje mít přehled o:

- svých knihách (název knihy, autor knihy, pořizovací cena, vydavatelství)
- výpůjčkách knih (název půjčené knihy, jméno, adresa zákazníka, který si ji půjčil, kdy si ji půjčil a kdy jí má vrátit)

Z potřeb knihovníka vytvoříme nenormalizovaný návrh databáze **Knihovna**:



Hrubý návrh

Z prvního hrubého návrhu vzešly dvě entity - *Knihy* a *Půjčky*, s atributy, primárními klíči (PK), a cizími klíči (FK) vypsanými ve schematu. Šipka značí vztah mezi relacemi.

1.4 NORMALIZACE NÁVRHU RELAČNÍ DATABÁZE

Od dobré databáze očekáváme, že bude fungovat **rychle a správně**. Při podrobnějším pohledu na náš hrubý návrh zjistíme jisté neefektivnosti:

1. Není příliš pracné při každé výpůjčce vyplňovat jméno a adresu zákazníka, který bude chodit pravidelně do knihovny?
2. Atribut *DATUM_VRACENI* v relaci entity *Půjčky* je zbytečné vyplňovat, dá se lehce vypočítat z data zapůjčení, je tedy nadbytečný.
3. Co když si jeden zákazník půjčí 20 knížek. Má smysl zadávat 20 záznamů pokaždé se stejným jménem, adresou a datem a odlišným identifikačním číslem do relace *Půjčky*?
4.

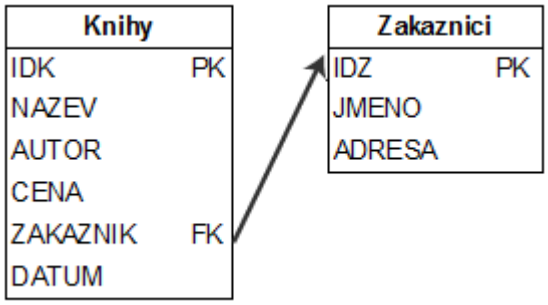
Pro rychlé a správné fungování databáze musí její návrh splňovat určité normy. Databáze, jejíž návrh příslušné normy splňuje se nazývá **normalizovaný návrh**. Pro vytvoření normalizovaného návrhu existují postupy, avšak dobří návrháři se kromě těchto postupů řídí svými zkušenostmi a citem pro strukturu dat. Nám snad postačí jedno pravidlo:

V relaci, která je normalizovaná, musí být každý neklíčový atribut jednoznačně určený svým primárním klíčem a ničím jiným než primárním klíčem.

Tedy k našemu hrubému návrhu:

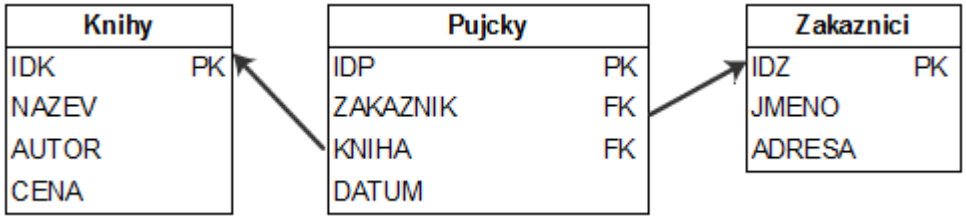
1. Atributy *JMENO_ZAKAZNIKA* a *ADRESA_ZAKAZNIKA* určitě nejsou jednoznačně určeny identifikačním číslem Půjčky. Proto pro ně vytvoříme novou relaci *Zákazník* a v relaci *Půjčka* bude vystupovat atribut *ZAKAZNIK* jako cizí klíč.
2. Atribut *DATUM_VRACENI* se dá vypočítat z atributu *DATUM*, proto jej z relace vypustíme.
3. Atributy *ZAKAZNIK*, *KNIHA* a *DATUM* v relaci *Půjčky* klidně mohou být atributy v relaci *Knihy*, proto je tam přesuneme a relaci *Půjčky* zrušíme.

Náš upavený, normalizovaný návrh může (ale nemusí) vypadat takto:



Normalizovaný návrh 1

Poznámka: Zrušení relace Půjčky a přesun evidence půjček do relace Knihy (atributy ZAKAZNIK a DATUM) má za následek to, že knihovník nemůže sledovat historii půjčování (u každé knihy se eviduje pouze aktuální situace – je-li půjčena, kým a od kdy, nebo není). Trvá-li knihovník na sledování historie půjček (např. chce vědět, o kterou knihu je zájem a o kterou nikoliv), je tento zjednodušený návrh sice normovaný, ale pro zákazníka nepoužitelný. V takovém případě musíme návrh upravit takto:



Normalizovaný návrh 2

2 OPENOFFICE.BASE A ZÁKLADY JAZYKA SQL

http://www.inpics.net/base.html

Logický návrh databáze máme na papíře hotový. Nyní nezbývá, než jej realizovat v nějakém databázovém systému. Na výběr máme:

- komerční systémy (Oracle, Microsoft SQL Server, DB2, INGRES, Microsoft Acces...)
- volně šiřitelné systémy (MySQL, OpenOffice.Base)

Bez ohledu na zvolený systém, databázový administrátor obvykle pracuje s databázovým systémem pomocí jazyka SQL (Microsoft Acces a OpenOffice.Base mají k dispozici i grafické (klikací) rozhraní). My budeme pracovat s modulem kancelářského balíku **OpenOffice.Base.** Většinu operací zvládneme pomocí grafického rozhraní, některé jsou ale mnohem rychlejší a snazší s pomocí jazyka SQL.

Náš jednoduchý normalizovaný **Návrh 1** databáze **Knihovna** převedeme do databázového systému OpenOffice.Base. V kostce budeme postupovat takto:

1. **Připravíme prázdné tabulky** odpovídající relacím *Knihy* a *Zákazníci*, které později koncový uživatel naplní daty. (No, koncové uživatele budete muset předstírat vy :))
2. **Nachystáme formuláře pro zadávání dat** do tabulek *Knihy* a *Zákazníci*, které koncovému uživateli usnadní zápis nových knih a zákazníků a znesnadní mu poškození databáze.
3. **Připravíme dotazy**. Dotaz je výběr jisté podmnožiny dat, z níž uživatel vyčte informace, které často potřebuje (např. zobrazení všech momentálně půjčených knih či těch knih a zákazníků, kterým již vypršela zápůjční lhůta...)
4. **Nachystáme sestavy**. Sestava je sada dat z databáze připravená pro tisk.

2.1 TABULKY

V grafickém rozhraní:

- **Vytvoříme novou prázdnou tabulku** odpovídající relaci *Knihy*. (Přístup k funkci: *Tabuky/ Vytořit tabulku v režimu návrhu)*
- zadáme **názvy polí** – budoucích sloupců tabulky (odpovídají atributům; **konvence** – **velkými písmeny bez mezer a diakritiky**), takže:
  Tabulka *Knihy* bude mít atributy: *IDK, NAZEV, AUTOR, ZAKAZNIK, DATUM*
- pro každé pole definujeme jeho **datový typ**:
  Text (text proměnné délky)
  Integer (celé číslo v rozsahu -2^{31} až $+2^{31}$)
  Number (číslo se zadaným počtem číslic před a za desetinnou čárkou)
  Date (datum)...
- pro každé pole doplníme políčka dialogového okna
- určíme, které pole představuje **primární klíč** (*klik PT nad prvním sloupcem tabulky* – objeví se klíček)
  Tabulka *Knihy* bude mít primární klíč: *IDK.*
- Návrh tabulky uložíme.

Stejně tak vytvoříme i tabulku **Zakaznici**

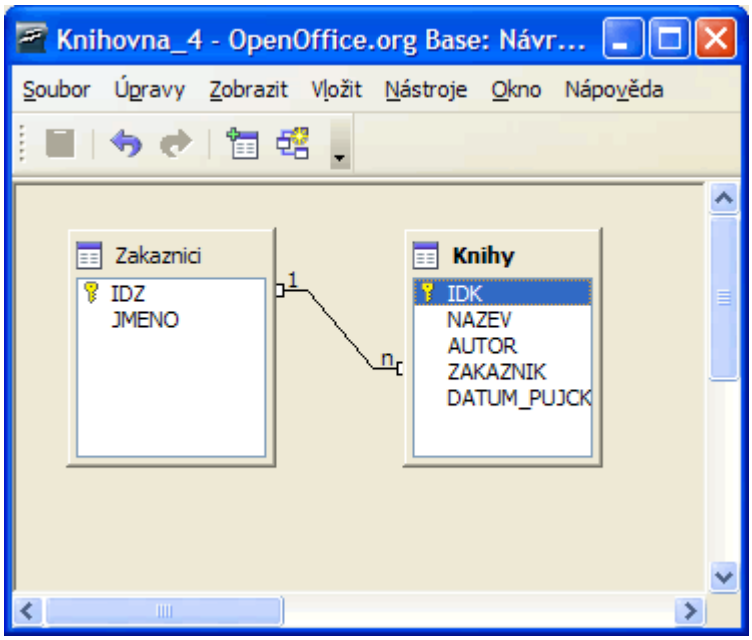
- Tabulka *Zakaznici* bude mít atributy: *IDZ, JMENO, ADRESA*
- Tabulka *Zakaznici* bude mít primární klíč: *IDZ*

2.2 VZTAHY MEZI TABULKAMI

Podle našeho návrhu je mezi tabulkami *Knihy* a *Zakaznici* vztah:

- Primární klíč *Zakaznik.IDZ* je cizím klíčem *Knihy.ZAKAZNIK*.

Tento vztah zadáme: *Nástroje/ vztahy*, pečlivě vyplníme dialogová okna a uložíme. V náhledu uvidíme náš návrh: (znaky "1" a "n" znamenají, že uvedený vztah je "jadna ku n", tedy jeden zákazník může mít půjčeno i více knih. Pokud se váš náhled liší, zle jej upravit: klik *PT/ upravit*...)



2.3 FORMULÁŘE

Pro knihovníka nachystáme formuláře:

1. **Vkládání nově koupené knihy** do tabulky *Knihy*:
   Přístup k funkci: *Formuláře/ Použít průvodce pro vytvoření formuláře*,
   - atributy, které se objeví ve formuláři: IDK, NAZEV, AUTOR;
   - bez podformuláře;
   - v režimu zadávání dat zatrhneme možnost "Formulář se bude používat jen pro zadávání dat, existující data nebudou zobrazena");
2. **Vkládání nového zákazníka** do tabulky *Zakaznici:*
   (podobné jako bod 1.)
3. **Půjčování knih**.
   (podobné jako bod 1., ale:
   - formulář slouží nikoliv k zadávání nových dat, ale k úpravě dat v tabulce *Knihy*;
   - atributy, které se objeví ve formuláři: IDK, NAZEV, AUTOR, ZAKAZNIK, DATUM;
   - k poli ZAKAZNIK se bude vázat podformulář zobrazující jméno zákazníka, který má knihu půjčenou;
   - v režimu zadávání dat zatrhneme možnost: "Formulář bude zobrazovat všechna data" a zakážeme odstraňování existujících a vkládání nových dat;)

2.4 ZADÁVÁNÍ DAT DO DATABÁZE

V této kapitole se na chvíli oprostíte od role databázového návrháře, administrátora a vyzkoušíte si svoji databázi jako koncový uživatel. Vaším úkolem bude vámi navrženou databázi naplnit daty. Využijte vámi navržené formuláře pro vkládání dat.

Úkol: Pomocí formulářů zadejte 10 knih, 10 zákazníků a "půjčte" 5 různých knih několika zákazníkům.

*Poznámka: Data lze snadno do databáze OpenOffice Base také importovat ze sešitu *.ods. Tabulku v sešitě označíme a prostým přetažením myší do tabulek databáze ji importujeme. Pozor: přetažením tabulky tuto zároveň definujeme – buňky prvního řádku budou použity jako atributy nové relace (tabulky) v Base.*

2.5 DOTAZY

Koncový uživatel vaší databáze nemá ani potuchy o tom, jakou strukturu jste vnesli do jeho dat. Ale ví, jaké informace chce z databáze získávat. Tyto informace vyčte z **dotazů**, které mu nachystáte.
Konkrétně náš knihovník chce vědět:

- Které knihy jsou momentálně k dispozici (jejich seznam).
- Vypůjčené knihy.
- Vypůjčené knihy i se jménem zákazníka, který je má půjčené.

V závěru kapitoly nachystáme tři dotazy, s pomocí dotazovacího jazyka SQL. Nejdříve se ho ale naučíme:

slovníček:

select	vyber (zobraz) sloupce ...
from	z tabulky ...
where	jejichž řádky splňují podmínku ...
order by	a seřaď je podle …
and	logický operátor "a" pro složené podmínky
or	logický operátor "nebo" pro složené podmínky
like	operátor pro porovnávání textů
%	zástupný symbol pro libovolný řetězec znaků, cokoliv, všechno (někdy též *)
null	prázdné políčko tabulky
is, is not	je, není

přístup k funkci v OOO.Base: *Dotazy/ Vytvořit dotaz v SQL pohledu...* pak napíšeme příkaz v SQL a stiskneme *Spustit dotaz*.

například:

1. **select NAZEV from Knihy**
   příkaz zobrazí seznam všech knih.

2. **select IDK, NAZEV from Knihy**
   příkaz zobrazí seznam všech knih i s identifikačním číslem

3. **select * from Knihy**
   příkaz zobrazí celou tabulku Knihy.

4. **select NAZEV from Knihy where IDK > 10**
   příkaz zobrazí seznam knih s identifikačním číslem větším než 10

5. **select NAZEV, AUTOR from Knihy order by NAZEV**
   příkaz zobrazí seznam knih seřazený abecedně podle názvu knihy.

6. **select JMENO from Zakaznici where JMENO like 'D*' order by IDZ**
   příkaz zobrazí seznam zákazníků, jejichž jméno začíná na D a seřadí jej vzestupně podle identifikačního čísla.

7. **select IDK, NAZEV, AUTOR from Knihy where ZAKAZNIK is NULL**
   příkaz zobrazí seznam knih (identifikační číslo, název a jméno autora), které mají prázdné políčko zákazník, což znamená, že nejsou půjčené. Tedy seznam nevypůjčených knih.

8. **select NAZEV from Knihy where ZAKAZNIK is NULL and AUTOR = 'Čapek Karel'**
   příkaz zobrazí seznam nevypůjčených knih od Karla Čapka.

Úkoly: Napište příkazy pro zobrazení:
1. Seznamu knih, které jsou vypůjčené.
3. Seznamu nevypůjčených knih od Karla Čapka.
4. Seznamu knih od Karla Čapka nebo od Jana Nerudy.

Dosud se naše dotazy týkali pouze dat z jedné tabulky (pouze Knihy, nebo pouze Zakaznici). Přitom užitečnost relačních databází tkví v tom, že mezi tabulkami existují vztahy (relace), které umožňují získávat správné informace z více tabulek. Využijeme tedy propojení tabulek pomocí cizího klíče:

9. **select NAZEV, AUTOR, ZAKAZNIK, JMENO from Knihy, Zakaznici where Knihy.ZAKAZNIK = Zakaznici.IDZ**
   příkaz zobrazí seznam vypůjčených knih (název, autor, číslo zákazníka) včetně jmen zákazníků, jež mají knihu půjčenou.

   Všimněte si použité konvence **Tabulka.SLOUPEC** pro vyjádření sloupce "ZAKAZNIK" z tabulky "Knihy" – **Knihy.ZAKAZNIK** a podobně i sloupce "IDZ" z tabulky "Zakaznici" – **Zakaznici.IDZ**.

Dotazy, které knihovník často používá, uložíme. Je to seznam nevypůjčených knih (č. 7), seznam vypůjčených knih (Úkol č. 1) a rozšířený seznam vypůjčených knih včetně jmen zákazníků, kteří mají danou knihu půjčenou (č. 9).

2.6 SESTAVY

Sestavy (reporty) jsou sady dat z databáze připravené pro tisk.

Úkoly: připravte sestavu pro vytištění seznamu vypůjčených knih včetně jmen příslušných zákazníků (odpovídá dotazu č. 9 z předchozí kapitoly).