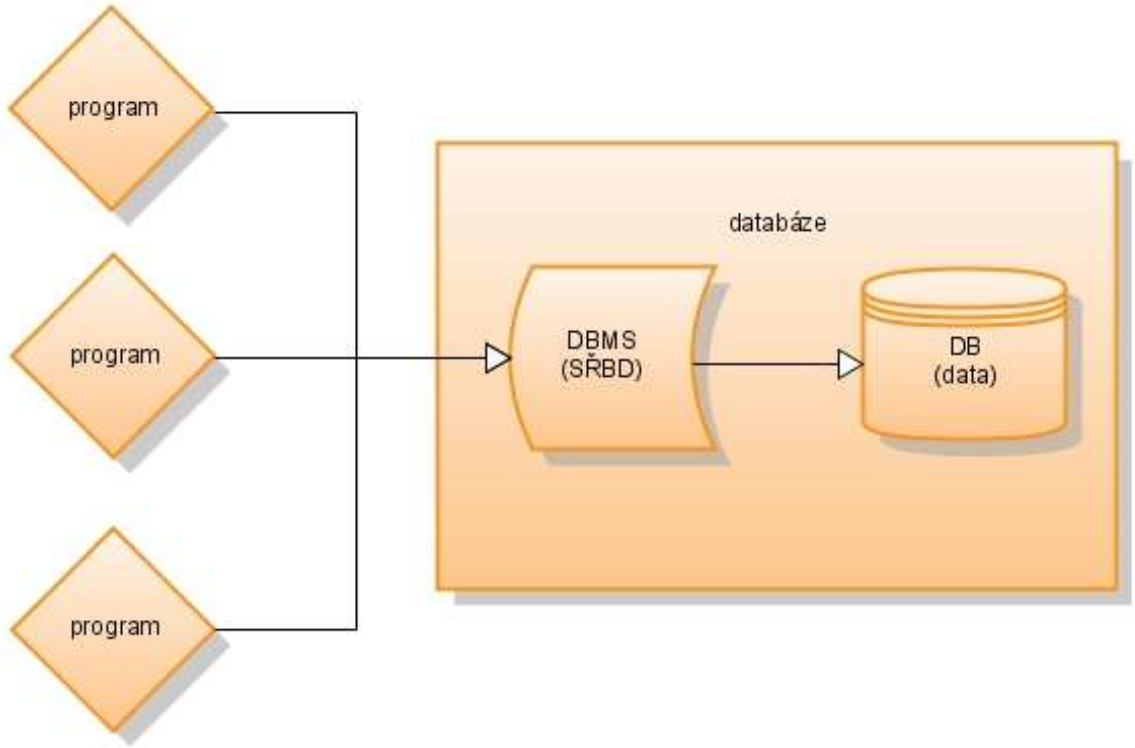
Relační databáze

Pojem databáze, druhy databází

Databází se myslí uložiště dat. V době začátků využívání databází byly tyto členěny hlavně **hierarchicky, případně síťově** (rozšíření hierarchického modelu). V roce 1970 položil E.F.Codd základy pro dnes nejrozšířenější **relační databáze.** V dnešní době se kromě nejčastějších relačních databází začínají prosazovat také databáze objektové a **objektově-relační.** Tyto databáze na sebe berou řadu vlastností z oblasti objektového programování (nové datové typy, možnost vykonávat nad sebou metody, ... , viz. http://nb.vse.cz/~zelenyj/it380/eseje/xlavj01/od.htm).

Pokud se hovoří o databázích, obvykle se rozlišují pojmy báze dat (**DB**) – samotná data – a pojem DataBase Management System (**DBMS**, nebo také česky SŘBD – systém řízení báze dat), který se stará o jejich fyzické uložení, správu a požadované operace.



V dnešní době existuje řada komerčních DBMS, nejznámější jsou:

MySQL – v základní verzi zdarma, využívá se především pro menší webové projekty. Před nedávnem jej pod své portfolio začlenila firma Sun. http://www.mysql.com

MSSQL – DBMS od společnosti Microsoft, využívá se při tvorbě drobnějších projektů především v programovacích jazycích podporovaných společností Microsoft, kde má rozsáhlou podporu. http://www.microsoft.com/sqlserver/2008/en/us/default.aspx

Oracle – DBMS pro rozsáhlé databázové systémy, nabízí především velice široké možnosti správy, ladění výkonu, řadu nadstandartních funkcí. http://en.wikipedia.org/wiki/Oracle_database

Všechny tyto databázové systémy pracují na základě jazyka **SQL (Simple Query Language)**, mají však obvykle různá vlastní vylepšení.

Relační databáze

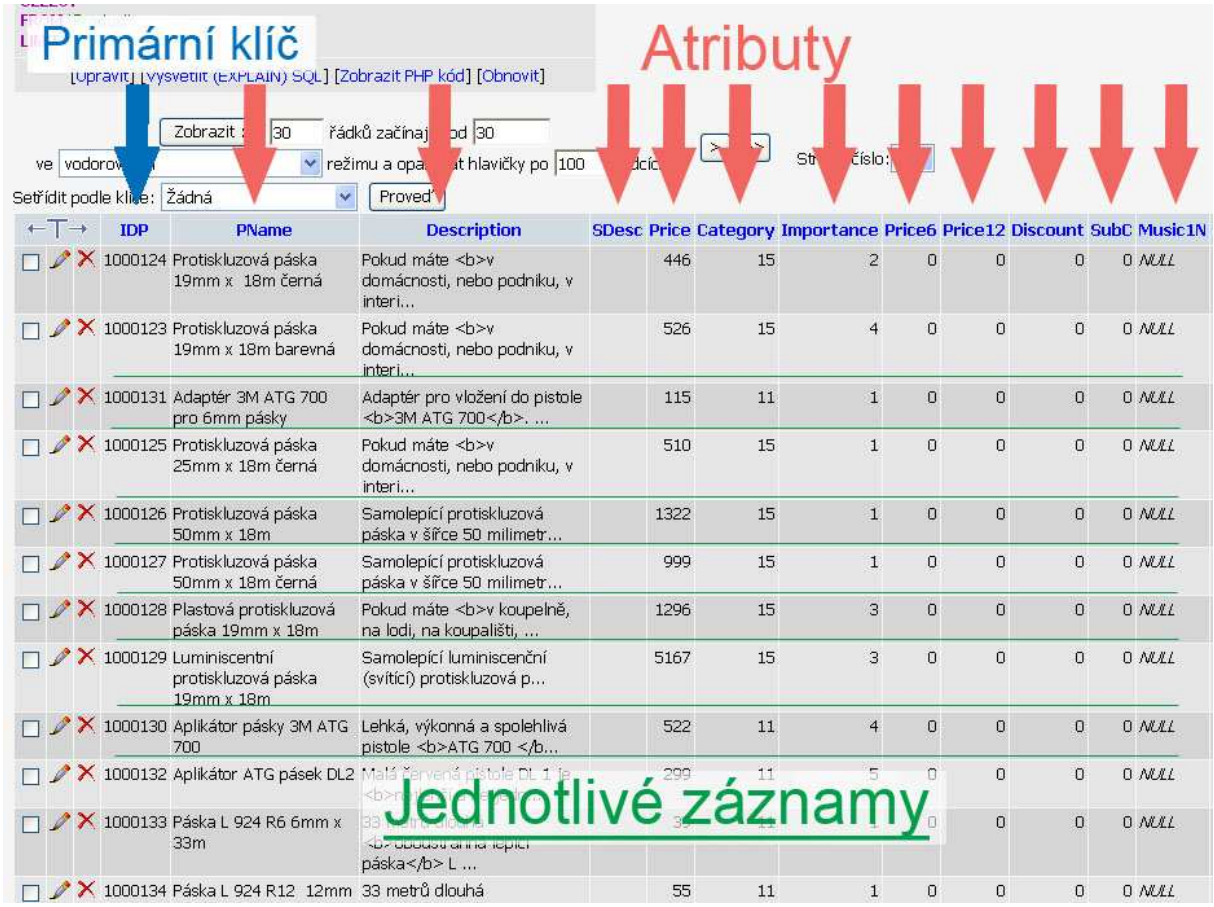
Relační databáze využívají jako základu **tabulek**, které jsou propojeny předem nastavenými vztahy. Tabulka obsahuje jednotlivé sloupce – **atributy** a řadky – **záznamy**. Atributy tabulky určují vlastnosti objektů, které se do tabulky budou vkládat – do tabulky se vkládají objekty stejného druhu, nicméně každý vždy jen jednou (viz integrita databáze). Atributy při návrhu tabulky však rozhodně neobsahují samotné hodnoty (!), pouze určují jaké vlastnosti budou vkládané objekty mít v databázi uložené.

Atribut

Atribut je sloupec v tabulce, vlastnost záznamů v tabulce. U atributu určujeme při návrhu tabulky jeho **doménu** – rozsah hodnot, kterých může nabývat (doména může být například u atributu datum narození zapsána jako *číslo od 1900 do 3000*). Při vlastní implementaci (sestavení) navržené databáze do DBMS se pak z domény atributů určí jeho **datový typ** (například pro doménu výše je to Integer = celé číslo). DBMS se pak postará o to, že do daného sloupce nebude možné zapsat hodnotu, která nevyhovuje datovému typu (v předchozím případě tedy nezapíšu třeba *nevím*).

Atomicita atributu –atribut je atomický, pokud jej již nelze rozložit na dílčí atributy (jméno na jméno, příjmení a titul například).

Atribut u jednoho záznamu může obsahovat vždy jen jednu hodnotu, v případě že potřebujeme vložit do atributu u jednoho záznamu více hodnot (například telefonních čísel), musíme pro ně vytvořit novou tabulku, ve které specifikujeme ke kterému záznamu se telefonní čísla váží.



Existují speciální druhy atributů, které plní v relační databázi specifické funkce, jsou to:

Primární klíč – tabulky nemají nijak pevně zafixováno pořadí záznamů (řádků), aby se tedy dalo jednoznačně ukázat na jeden záznam (pro editaci, nebo jeho vybrání), je nutné zvolit jeden atribut tabulky jako unikátní. Takovému atributu říkáme primární klíč – **každý záznam v tabulce musí mít hodnotu tohoto atributu unikátní pro celou tabulku** (nesmí se vyskytnout dva lidé se stejným RČ). Každá tabulka **musí** mít vytvořen primární klíč.

Cizí klíč – záznamy v tabulkách, které jsou propojeny vztahy musí být nějakým způsobem provázány. Provázání se docílí vytvořením speciálního atributu – tzv. cizího klíče – v jedné z tabulek. **Do tohoto atributu se pak bude u jednotlivých záznamů kopírovat hodnota cizího klíče navázaného záznamu z druhé tabulky.**

Datové typy, které byste měli znát:

Integer – celé číslo

Varchar – krátký text (do 255 znaků)

Text – dlouhý text

Boolean – logická hodnota (ano/ne, 1/0, true/false)

Propojení tabulek

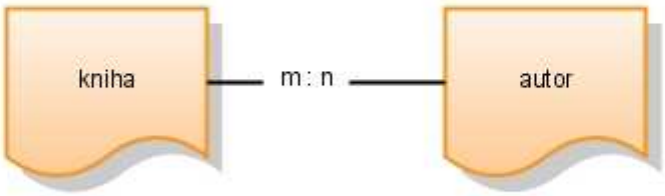
Některé tabulky v relační databázi mají mezi sebou logické vztahy, ty určujeme při návrhu databáze. Například v databázi knihovny budou mít mezi sebou vztah tabulky *spisovatel a kniha*. V rámci návrhu databáze musíme tyto vztahy nejprve popsat (jak ten vztah vypadá, tedy například *spisovatel* **napsal** *knihy*, určit multiplicitu) a pak navrhnout i jejich fyzické vytvoření.

Multiplicita vztahu

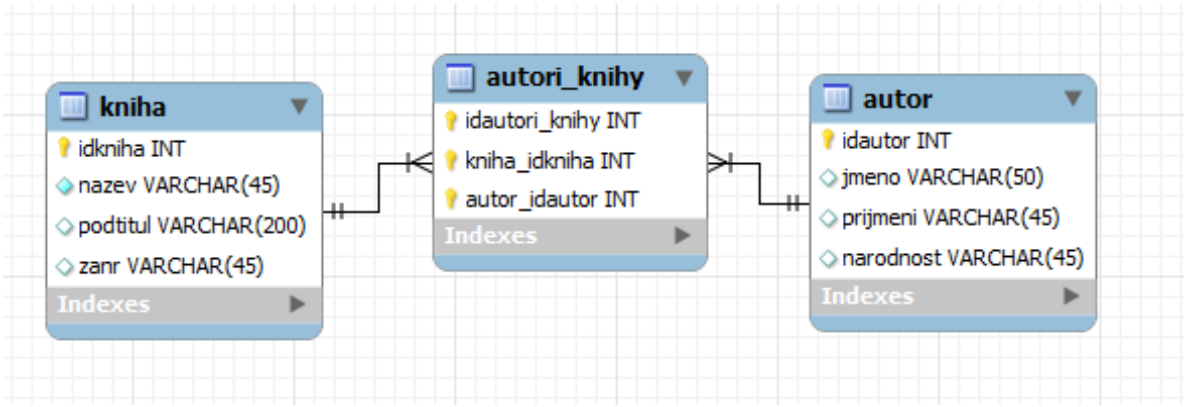
Po slovním označení vztahu je nutné určit jeho multiplicitu – **multiplicita určuje, kolik záznamů v první tabulce může být spojeno s jedním záznamem v tabulce druhé a naopak**.

V našem případě se tedy zamyslíme, kolik spisovatelů mohlo napsat jednu knihu (jeden i více) a počet zapíšeme nad tabulku spisovatelů. Poté provedeme to samé z opačného pohledu – kolik knih mohl napsat jeden spisovatel (jednu i více) a počet opět zapíšeme, pro změnu nad tabulku knih. Vidíme tedy, že mezi tabulkami bude vztah **m:n** (více ku více). Vztah s takovou multiplicitou nelze rovnou fyzicky zapsat do DB a **je nutno jej rozložit** – viz níže.

Pokud bychom měli databázi fotbalových klubů a jejich hráčů, vztah mezi tabulkami bude **1:n**, neboť jeden tým má jistě více hráčů, ale jeden hráč



může být pouze v jednom týmu. Vtakovém případě je fyzické propojení jednoduché – v **tabulce nad kterou je N** (fotbalisté) **vytvoříme nový atribut – cizí klíč** – do kterého budeme vkládat **identifikaci klubu**, ve kterém hráč hraje (kopie primárního klíče hráčova klubu z klubové tabulky).



Rozdělení vztahu m:n na dva vztahy 1:n

Pokud máme mezi tabulkami vztah **m:n** (autor napsal X knih, ale jedna kniha může být napsána Y autory), je nutné jej **rozložit na dva vztahy 1:n** (protože nemůžeme třeba do cizího klíče u knihy zapsat více autorů – atribut může obsahovat jenom jednu hodnotu). Vytvoříme tedy separátní tabulku, ve které se budou objevovat jen cizí klíče z obou tabulek (tedy identifikace knihy a identifikace autora). V takové tabulce se pak může vícekrát objevit jeden autor i jedna kniha.

V některých případech je možné se setkat i s **multiplicitou 1:1**, tedy jeden záznam v první tabulce odpovídá přesně jednomu záznamu v tabulce druhé. Příkladem může být tabulka top manažerů a tabulka firemních limuzín, kdy vždy jedna limuzína je přidělena jednomu manažerovi. V takovém případě **je jedno do které tabulky umístíme cizí klíč**.

Možné multiplicity: 1:1, 1:n, m:n. Vztah m:n se musí rozepsat na dva vztahy 1:n.

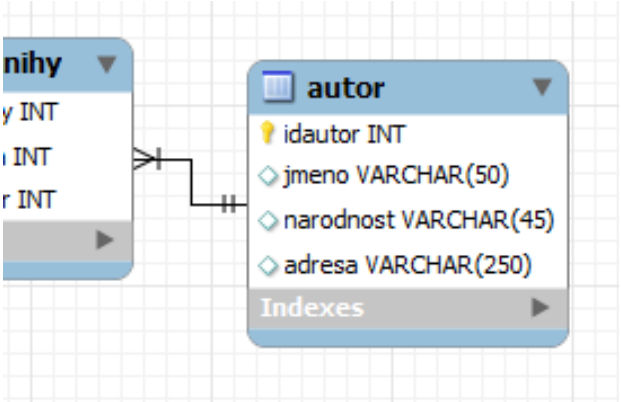
Normalizace návrhu databáze

Normalizace návrhu databáze by měla databázi zefektivnit, zmenšit její datovou náročnost a zajistit menší náchylnost k nejrůznějším chybám.

http://interval.cz/clanky/databaze-a-jazyk-sql/

První normální forma

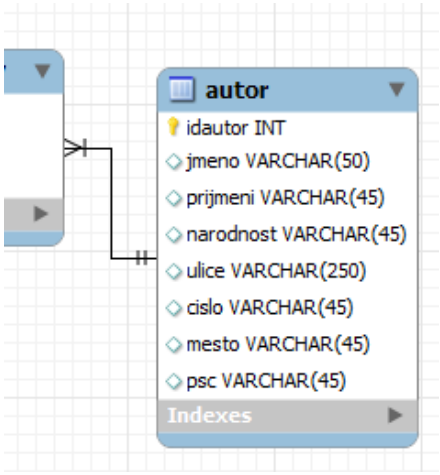
Všechny atributy mají být **atomické**, tedy obsahovat pouze jednu **nedělitelnou hodnotu**. Například pokud bychom měli atribut adresa (Tupolevova XYZ, Praha 9, 19900), je vhodné jej rozdělit do jednotlivých atmických atributů (ulice, číslo, město, psč) – důvodem je v případě neatomického atributu omezená možnost práce s databází – nenajdeme například všechny osoby s psč 19900 apod.



Neatomické hodnoty (jmeno, adresa)

Druhá normální forma

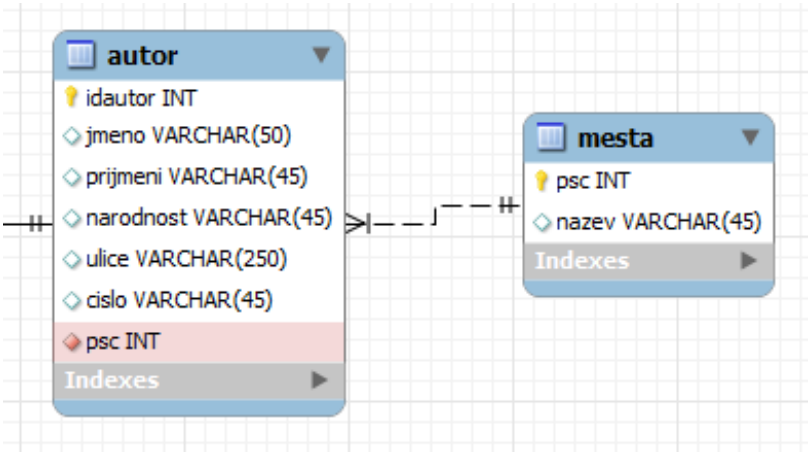
Všechny atributy mají být **závislé na primárním klíči**. Nelze tedy mít například tabulku automobilů s atributy (SPZ, značka, typ, motor, **servis, telefon_servis**), neboť atribut **telefon_servis je závislý nikoliv na SPZ auta, ale na atributu servis**. Problém by vznikl například při změně telefonního čísla servisu, které by se muselo měnit u mnoha vozů najednou. Je proto nutné vytvořit speciální tabulku servisů s jejich detaily.



Autoři v druhé normální formě

Třetí normální forma

Atributy musí být závislé přímo na primárním klíči tabulky. Typickým příkladem je tabulka zaměstnanců, jejich pozic a platů. Pokud je plat odvozen od pozice zaměstnance, je závislý na tomto atributu a nikoliv na primárním klíči zaměstnance. Opět se tedy vytvoří samostatná tabulka ohodnocení jednotlivých pracovních pozic zaměstnanců.



Tabulka autorů vyhovující 3. NF

Referenční integrita

Pokud databázi správně navrhneme, DBMS nám může zajistit mezi propojenými tabulkami takzvanou referenční integritu. V databázi knihovny nám referenční integrita **znemožní odstranit autora**, který je uveden u některé v databázi uložené knihy. Také nám n**eumožní ke knize určit autora, který neexistuje v tabulce autorů**. Také nám neumožní **změnit primární klíč u autora, který je používán jako cizí klíč v jiné tabulce**.

SQL – Simple Query Language – jazyk pro práci s DB

Primární klíč – atribut jednoznačně identifikující jeden záznam v tabulce

Cizí klíč – ukazuje, ke kterému záznamu v druhé tabulce se záznam vztahuje

DBMS – systém spravující data

Atribut – sloupeček v tabulce, vlastnost záznamů

Atomicita – atribut nelze rozdělit na dílčí atributy

Doména – rozsah hodnot, které může atribut dosahovat

Datový typ – druh dat, která může atribut obsahovat (číslo, text,...)

Multiplicita – kolik záznamů v jedné tabulce se váže na kolik záznamů v tabulce druhé

Referenční integrita – udržování vztahu mezi tabulkami

Normalizace databáze – odstranění redundance dat, zlepšení struktury databáze, lepší práce s DB v budoucnu

Některé databázové systémy pak umožňují takzvané **kaskádové odstraňování a update** – nastavuje se obvykle samostatně pro jednotlivé vztahy mezi tabulkami. Pokud bysme jej zprovoznili u databáze knihovny, smazaly by se v případě odstranění autora z DB všechny knihy které napsal. V případě změny primárního klíče autora (RČ) by se pak změnily cizí klíče všech autorových knih.

Zdroje

Wiki – http://www.wikipedia.com

MySQL – http://www.mysql.com

Grafické návrhy – http://www.gliffy.com , http://dev.mysql.com/workbench/