2 IMPLEMENTÁCIA BÁZY DÁT

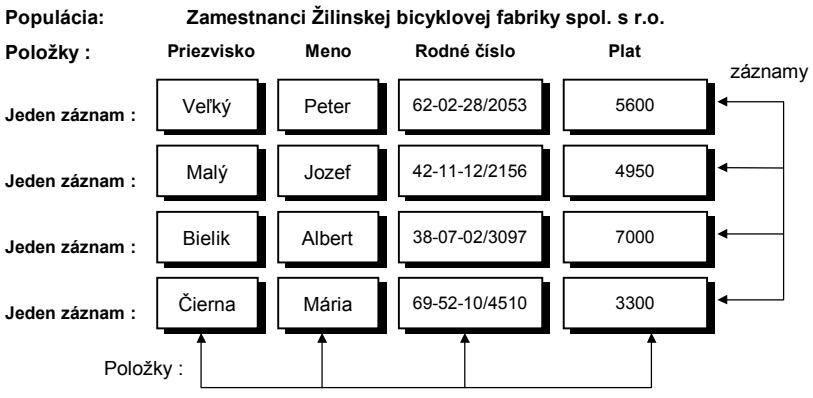
Základ činnosti informačných systémov predstavujú údaje, ktoré sú zviazané s objektmi reálneho sveta - popisujú ich vlastnosti, chovanie a podobne. Prvé informačné systémy používali tzv. ***agendový spôsob spracovania dát*** - spracovávali sa jednotlivé podnikové agendy (skladové hospodárstvo, mzdy, účtovníctvo a pod.). Agendový spôsob spracovania dát mal však určité nevýhody, ktoré obmedzovali rozvoj automatizácie takýchto informačných systémov. K ***nevýhodám*** patrili predovšetkým tieto :

- silná závislosť programov a údajov - presné opisy organizácie údajov boli len v programoch, ktoré s nimi pracovali a pri potrebe zmeniť organizáciu dát bolo potrebné zmeniť aj programy,
- nadmerná nadbytočnosť (redundancia) údajov - rovnaké údaje bolo často potrebné zadávať do rôznych aplikácií (agend) a udržiavať ich, aj keď v iných aplikáciách sa už vyskytovali,
- nekompatibilita údajov - v rôznych agendách sa rovnaké údaje spracovávali rôzne a výsledné informácie z rôznych agend mohli byť preto odlišné, aj keď šlo o rovnaké výstupy.

Aby sa odstránili tieto a ďalšie problémy a dosiahla sa čo najväčšia nezávislosť dát od programov moderné informačné systémy používajú tzv. ***databázové spracovanie dát***.

Jednotlivé **údaje sú uložené v *báze dát* (*databáza*),** ktorá predstavuje kolekciu vzájomne súvisiacich údajov. Databáza má spravidla jednu internú organizáciu údajov pre všetky oblasti a spôsoby ich využitia. **Databáza reprezentuje určité vybrané aspekty reálneho sveta**, ktoré majú pre používateľa systému informačný význam. **Údaje** uložené v databáze **sa môžu meniť** (aj pomerne často). **Údaje vyskytujúce sa v databáze v určitom časovom momente nazývame *databázovou inštanciou* (alebo *výskytom, stavom*).** Implementačné programy obsahuje zovšeobecňujúci a integrujúci programový systém - ***systém riadenia bázy dát*** - *SRBD*. SRBD je ***súborom programov, ktoré používateľovi umožňujú definovanie a konštruovanie databáz, ale aj manipuláciu s údajmi v nich uloženými***.

Na **údaje uložené v databáze** možno hľadieť ako na určitú ***populáciu* informácií**. Pojem populácia v tomto význame označuje **ľubovoľnú skupinu (resp. triedu) objektov (entít), ktorú môžeme definovať**. Ak vytvárame databázu, populácia je to, o čom chceme mať prehľad; to, čo sa stáva základom databázy. Napríklad populáciu môžu tvoriť zamestnanci firmy, pre ktorú navrhujeme informačný systém, výrobky evidované v jej sklade, knihy evidované v knižnici a pod. Uveďme si ako príklad zoznam pracovníkov firmy Žilinská bicyklová fabrika, spol. s r.o. (ŽBF). Populáciou databázy ŽBF (Obr. 1) sú pracovníci firmy; každý *záznam* (veta) v databáze obsahuje informácie o jednom členovi tejto populácie. Jednotlivé *položky* (*polia* - *fields*) v každom zázname zachytávajú dôležité detaily a príslušnom členovi.



Obr. 1 Prvky databázy

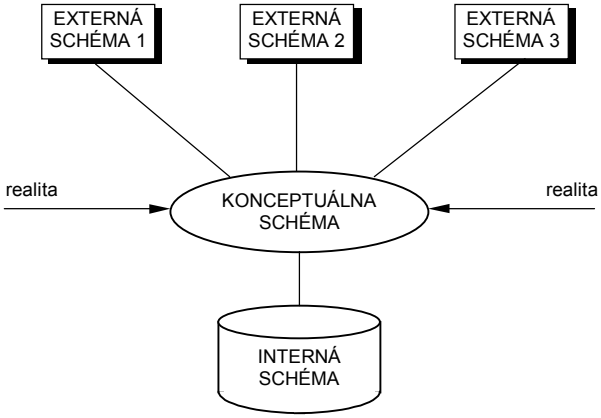
V súčasnosti **dostupné systémy riadenia bázy dát možno rozdeliť** na **niekoľko základných typov**, ktoré sú **charakterizované odlišným spôsobom organizácie bázy dát**.

Každý **spôsob usporiadania bázy dát možno popísať dátovým modelom**, ktorý **popisuje vzťahy medzi jednotlivými položkami databázy** (napr. vzťah každej položky ako je rodné číslo alebo plat ku konkrétnemu zamestnancovi, ktorého popisuje celý záznam, pričom tento záznam môže mať vzťah k iným entitám v databáze - napr. k nadriadenému pracovníkovi) **a spôsob, akým sú údaje sprístupňované užívateľovi a programátorovi**. V informačných systémoch sa v súčasnosti stretávame s využívaním týchto **základných dátových modelov**: **hierarchický** dátový model, **sieťový** dátový model a **relačný** dátový model. Okrem toho sa objavuje nový prístup k návrhu SRBD - tzv. **objektovo orientované** SRBD.

Väčšina **dátových modelov nepopisuje spôsob, akým sú dáta ukladané na disku**. Tieto detaily riešia programátori, ktorí navrhujú SRBD. V niektorých prípadoch však môže použitý model nepriamo obmedzovať spôsob uloženia dát, aby SRBD mohol spĺňať všetky požiadavky kladené na daný model.

2.1 Architektúra databázy

Základným východiskom pre aplikáciu databázových systémov je tzv. **trojúrovňová architektúra** definovaná v roku 1972 americkým národným výborom pre štandardy (ANSI/SPARC Group on Data Base Management Systems - American National Standards Committee on Computers and Information Processing / Standard Planning and Requirements Committee). Táto architektúra zavádza tri úrovne dátových štruktúr (Obr. 2):



Obr. 2 Princíp trojúrovňovej architektúry

1. ***Externá úroveň*** **- predstavuje potreby a požiadavky užívateľa.** Je realizovaná pomocou **externej schémy**. Tá sa vyjadruje prostredníctvom jazyka, ktorý umožní užívateľovi formulovať jeho logické požiadavky. Ide o kombináciu dvoch jazykov - tzv. *Data Definition Language* (DDL) a *Data Manipulation Language* (DML), pomocou ktorých **užívateľ deklaruje svoje dátové objekty a vzťahy medzi nimi a** pomocou ktorých **rieši svoje úlohy s týmito dátami**. Pretože užívateľov je spravidla viac a obvykle chcú pracovať s dátami z rôzneho pohľadu a s inými požiadavkami na ne, **môže existovať viacej užívateľských externých pohľadov**. Pretože nie všetkých užívateľov zaujíma všetko, jeden užívateľský pohľad bude podmnožinou (subschémou) celej komplexnej štruktúry. **Externá schéma závisí predovšetkým od požiadaviek užívateľov**. Zmena v užívateľských pohľadoch by sa nemala prejaviť v zmene koncepčného modelu ale len programovou zmenou v mapovacom procese medzi externými pohľadmi (subschémami) a koncepčnou schémou.
2. ***Konceptuálna úroveň*** **- reprezentuje celý informačný obsah databázy**, **ktorý by mal byť nezávislý ako na fyzickom riešení, tak aj na okamžitých potrebách užívateľa**. Konceptuálna úroveň je realizovaná vhodnou konceptuálnou schémou. Konceptuálnu schému možno tvoriť alebo tzv. ***podnikovým prístupom*****, ktorý je nezávislý na jednotlivých užívateľských pohľadoch** a mal by **čo najvernejšie odrážať** danú **realitu systému**, alebo tzv. ***integračným prístupom*****, kedy je zjednotením rôznych užívateľských pohľadov na dáta**. Hlavným nástrojom tvorby konceptuálnej schémy sú **entitno-relačné diagramy**.

3. ***Interná úroveň*** **- zaoberá sa problematikou fyzickej pamäťovej štruktúry uloženia dát** - pamäťovými blokmi, stránkami, adresami, smerníkmi. **Využíva vlastnosti konkrétneho operačného systému a konkrétnej počítačovej štruktúry**. Užívateľ s touto úrovňou prakticky neprichádza do styku, zaoberá sa ňou len administrátor databázy. Ak z nejakých dôvodov dôjde k zmene internej schémy (napr. zmenou počítačového systému alebo prechodom na iný operačný systém) nesmie sa to prejaviť v konceptuálnej úrovni. Prejaví sa to len v zmene funkcie transformačného mapovacieho procesu medzi konceptuálnou a internou schémou.

2.2 Postup pri výstavbe databázy

Proces výstavby databázy treba chápať ako dynamický a cyklický proces, ktorý je integrálnou súčasťou výstavby celého informačného systému. Vo všeobecnosti sa postupuje v nasledujúcich krokoch :

1. **Analýza požiadaviek užívateľov** - zameriava sa na potencionálnych užívateľov, ktorí k tomu boli vytypovaní ako výsledok všeobecnej analýzy informačných potrieb. Výsledkom analýzy požiadaviek je definovanie externej schémy (resp. externých schém).
2. **Návrh konceptuálnej schémy databázy** - navrhnutá schéma by mala byť nielen syntézou požiadaviek zákazníkov, ale byť aj obrazom reálneho sveta. Každý model predstavuje určitú abstrakciu reálneho sveta, ktorá vedie k vytvoreniu účelového modelu.
3. **Návrh logickej štruktúry databázy** - vychádza z konceptuálnej schémy, mal by však akceptovať externé pohľady budúcich užívateľov. Tu sa berú do úvahy predovšetkým charakter úloh a užívateľské prostredie. Pri návrhu logickej štruktúry je potrebné použiť vhodný dátový model.
4. **Návrh fyzického interného modelu databázy** - spravidla je určený vlastnosťami vybraného (prípadne už zakúpeného) štandardného SRBD. Bežný užívateľ nemá žiadnu a správca databázy len obmedzenú možnosť rozhodovať o vnútornom usporiadaní záznamov na médiách a technikách prístupu k nim. Tu nás zaujímajú predovšetkým nároky na pamäť a celková efektívnosť práce.
5. **Implementácia databázy** - zahŕňa jej inštaláciu (t.j. oživenie príslušného SRBD) na konkrétnom počítačovom systéme. To úloha predovšetkým pre správcu databázy, ktorý musí konfigurovať databázové prostredie, definovať potrebné pamäťové priestory a určiť príslušné oprávnenia. Musí tiež preveriť základnú funkčnosť databázy.
6. **Prvotné naplnenie bázy dát** - predstavuje problémové miesto pri výstavbe databázy, pretože obvykle ide o veľký objem údajov, ktoré treba do databázy zadať v relatívne krátkom čase. Túto úlohu možno plniť viacerými spôsobmi :
   - ***naplňaním prázdnej databázy*** pri postupnom odlaďovaní jednotlivých funkcií celého informačného systému; ide o pomerne priaznivú situáciu, kedy sa pri ladení funkcií systému postupne zadávajú údaje do bázy dát,
   - ***konverziou existujúcich dát*** už raz zozbieraných ale spracovávaných inými prostriedkami nástrojmi, ktoré poskytuje samotný SRBD alebo konverznými programami špeciálne k tomuto účelu vytvorenými,
   - ***špeciálnym zberom dát*** v realite v prípade, keď dáta ešte neboli zozbierané. Takýto zber dát je obvykle pomerne pracný a trvá dlhšie časové obdobie, čo spôsobuje problémy s nekonzistenciou dát (niektoré dáta zachytávajú stav tento mesiac, ďalšie až nasledujúci) - tento problém sa môže vyriešiť tzv. „zmrazením“ dát k určitému dátumu, pričom aktualizácia dát sa uskutoční po spustení databázy.

   Špecifickým **problémom** tejto **etapy je *zabezpečenie správnosti vkladaných dát***. Je nevyhnutné, aby vkladané dáta boli bezchybné, pretože po rozbehu databázy by chyby mohli spôsobiť ne-

funkčnosť databázy. Preto vstupné dáta treba podrobovať programovým kontrolám, ktoré by mali odhaliť nielen formálne ale aj logické chyby.

7. **Overenie funkcie databázy** - v tomto kroku sa postupuje prototypovým prístupom, t.j. najskôr sa overí základné funkčné jadro databázy pre informačné zabezpečenie niektorých užívateľov alebo len niektorých ich potrieb a potom sa postupne rozširuje spektrum funkcií. Súčasne s tým prebieha školenie užívateľov v práci s informačným systémom. Hlavným kritériom správnosti funkcie databázy musí byť stupeň uspokojovania informačných potrieb užívateľov v rámci budovaného informačného systému a nie prevádzka databázového systému ako takého.
8. **Prevádzkovanie informačného systému a jeho ďalší rozvoj** - v tejto fáze je potrebné sledovať chod databázy a odhaľovať chyby, nedostatky a úzke miesta a odstraňovať ich. S rastom vyspelosti užívateľov a rastom ich informačných potrieb a požiadaviek vzniká potreba rozširovať obsah databázy a jej funkcií.

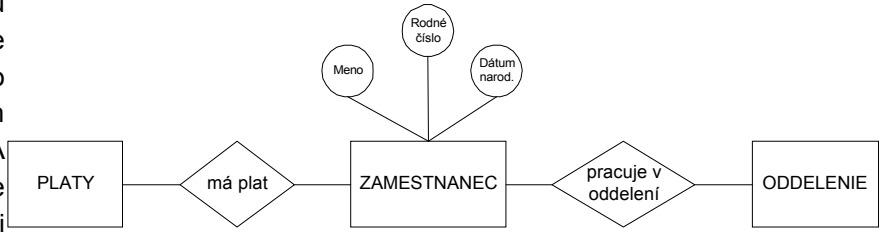
2.3 Opis bázy dát - dátové modely

Databáza v podstate **vždy obsahuje údaje o nejakom objekte**, ktorý je predmetom nášho záujmu, a jeho okolí - tzv. realite, pričom sa vždy snažíme zachytiť všetky podstatné skutočnosti tak, aby presne a časovo aktuálne odrážali vlastnosti a deje v našom okolí. Pretože nie je možné ani nevyhnutné postihnúť okolitý svet v celej zložitosti, **vyberieme si len určitý výsek reality a pre tento vytvárame abstraktný model**. **Z vytvorené abstraktného modelu je možné štandardným postupom navrhnúť konceptuálnu organizáciu dát**. Jedným z možných spôsobov modelovania reality je vytvorenie entitno-relačného modelu.

2.3.1 Entitno-relačný model

Základné modelovacie prostriedky, pomocou ktorých modelujeme realitu sú ***typ objektu*** **a** ***vzťah***. **Model reality sa skladá v každom okamihu z množín objektov a zo vzťahov (relácií) medzi týmito množinami**. **Množine objektov jedného typu v danom časovom okamihu** hovoríme ***populácia typu***. **Pre každý vzťah je v každom okamihu daná relácia**, ktorú nazývame ***populácia vzťahu***. Model reálneho sveta je teda v každom časovom okamihu mnohodruhovou relačnou štruktúrou $s=(m_1,\ldots, m_n, r_1,\ldots, r_u)$, kde m_i sú populácie typov M_i (i=1, …,n) a r_i sú populácie vzťahov R_j (j=1, …, u).

Napríklad populácia typu objektu ZAMESTNANEC je množina zamestnancov ako súčasť modelu reality v istom čase. Populácia vzťahu MÁ PLAT je relácia, ktorá je pre tento čas daná nad množinami objektov typu ZAMESTNANEC a PLATY.



Obr. 3 Grafické znázornenie entitno-relačného modelu

V grafickom znázornení modelu sú použité obdĺžniky pre typy objektov, kosoštvorce pre vzťahy, spojnice vyznačujú obory definície vzťahu a môžu im byť v rámci vzťahu pridelené nové mená, rozdielne od mien typov objektov.

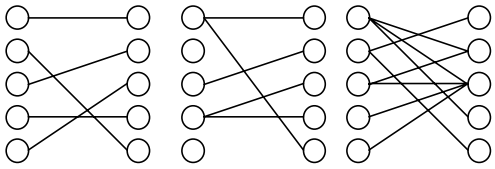
Pre ďalšie spresnenie modelu reality je vhodné rozlišovať medzi dvomi triedami typov objektov. ***Objekty prvej triedy*** **sú abstrakciou takých vecí a skutočností reality, o ktorých chceme v báze dát registrovať dáta tvoriace predmet nášho záujmu** - sú to teda **rozlíšiteľné a identifikovateľné objekty sveta objektov**. Takéto objekty nazývame ***entity***. **Množina podobných entít** (z určitej subjektívnej úrovne popisu pomocou reprezentovateľných hodnôt ich vlastností) sa nazýva ***entitná množina***. Entitné množiny označujeme *E, E_1, E_2*. **Subjektívna úroveň popisu** znamená **výber niektorých vlastností entít**, ktoré sa nazývajú ***atribúty***. Atribút *A* je funkcia definovaná na entitnej množine *E* priraďujúca každej entite $e \in E$ najviac jednu hodnotu z množiny hodnôt V_A**. Vždy musí existovať atribút (resp. množina atribútov), ktoré rozlišujú jednotlivé entity z *E* navzájom**. **Táto množina atribútov, ak je minimálna, sa nazýva *kľúč entitnej množiny*. Ak je atribútov niekoľko, je jeden z nich projektantom aplikácie vybraný ako *primárny***. Druhá trieda objektov charakterizuje, opisuje entity a tieto typy objektov nazývame *typy hodnôt*. V predošlom príklade je entitou ZAMESTNANEC, ostatné typy objektov sú typy hodnôt.

Vzťahy entít sú chápané ako *n*-tice (e_1, e_2, …, e_n) z kartézskeho súčinu entitných množín E_1, E_2, …, E_n, kde $n \geq 2$, pričom E_i nemusia byť navzájom nutne rôzne.

Tak ako entity možno zoskupovať do množín objektov istých vlastnostní podľa daných spoločných množín atribútov, možno zoskupovať vzťahy entít podľa toho, či spolu logicky rovnako súvisia.

Zodpovedajúce množiny vzťahov sa nazývajú vzťahové množiny a označujú sa R, R_1, … Opäť možno hovoriť o *type vzťahu R*. Ako entitné množiny tak aj vzťahové množiny môžu byť definičným oborom nejakých atribútov. Vzťah R je *atribútom* definovaným na type entity E, ak z oborov jeho definície je práve jeden typ E a ostatné sú nejaké typy hodnôt. *Jednoduchý atribút* má iba dva obory definície, ostatné atribúty sa nazývajú *zložené*. Hovoríme, že atribút R je *funkčným atribútom*, ak v každom stave pre každú entitu e z populácie entít typu E existuje práve jedna k-tica hodnôt $<n_1, \ldots, n_k>$ taká, že $<e, n_1, \ldots, n_k>$ patrí do populácie R v tomto stave, pričom k+1 je stupeň vzťahu R.

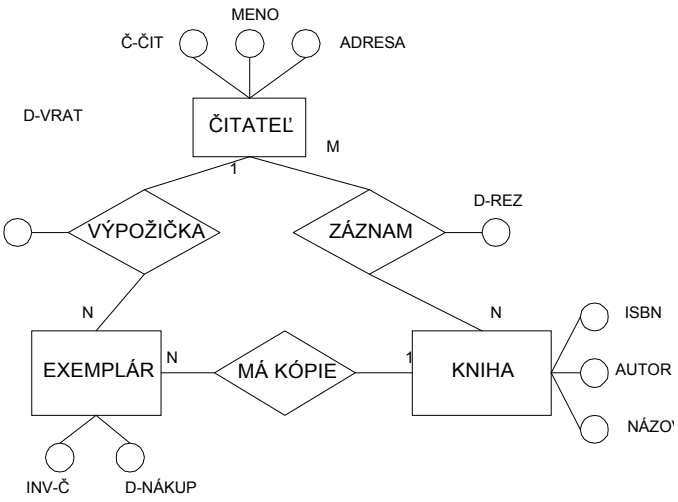
Posledným základným prvkom entitno-relačného (E-R) modelu sú tzv. ***charakteristiky vzťahov*** v R. **Pre binárne vzťahové množiny *R*** (sú definované nad dvomi entitnými množinami) sa zapisujú **1 : 1, 1 : *N*, *N* : 1, alebo *N* : *M***. Znaky udávajú počet entít jedného typu, ku ktorým môže byť entita iného typu vztiahnutá vo dvojiciach z R nad E_1 a E_2. Binárny entitný vzťah medzi dvomi typmi entít E_1 a E_2 posudzujeme vzhľadom na obidve relácie $R(E_1, E_2)$ a $R(E_2, E_1)$. Zaujímame sa predovšetkým o ich charakter - či ide o funkcie (t.j. každej entite z E_1 priraďuje práve jednu entitu z E_2 - totálna funkcia, resp. maximálne jednu entitu z E_2 - čiastočná funkcia). Ak sú obidve relácie $R(E_1, E_2)$ a $R(E_2, E_1)$ funkcie, označujeme vzťah medzi E_1 a E_2 ako vzťah typu 1 : 1 (napr. ZAMESTNANEC a RODNECISLO). Ak pre $e_1 \square E_1$ môže existovať viac ako jedna $e_2 \square E_2$ taká, že $(e_1, e_2) \square R$ (v takomto prípade je len jedna z relácií $R(E_1, E_2)$ a $R(E_2, E_1)$ funkciou), potom je charakteristika typu 1 : N (napr. RIADITEL a ZAMESTNANEC). Ak ani jedna z uvedených relácií nie je funkcia, t.j. pre $e_2 \square E_2$ môže existovať viac ako jedna $e_1 \square E_1$ taká, že $(e_1, e_2) \square R$ ide o vzťah M : N (napr. AUTOR a KNIHA).



Obr. 4 Typy entitných vzťahov

Príklad:

Uvažujme model knižnice, kde exemplár každej knihy je daný svojim názvom (NÁZOV), menom autora (AUTOR), inventárnym číslom (INV-Č) a vlastná kniha medzinárodným identifikačným kódom ISBN. Čitatelia sú identifikovaní číslom (Č-ČIT), menom (MENO) a adresou (ADRESA). Vypožičiavajú si jednotlivé exemplár s dátumom vrátenia (D-VRAT), môžu si tiež knihy rezervovať s dátumom rezervácie do D-REZ, prípadne s určenou prioritou. Príklad odpovedajúceho entitno-relačného diagramu je na obrázku (Obr. 5).



Obr. 5 Entitno-relačný diagram

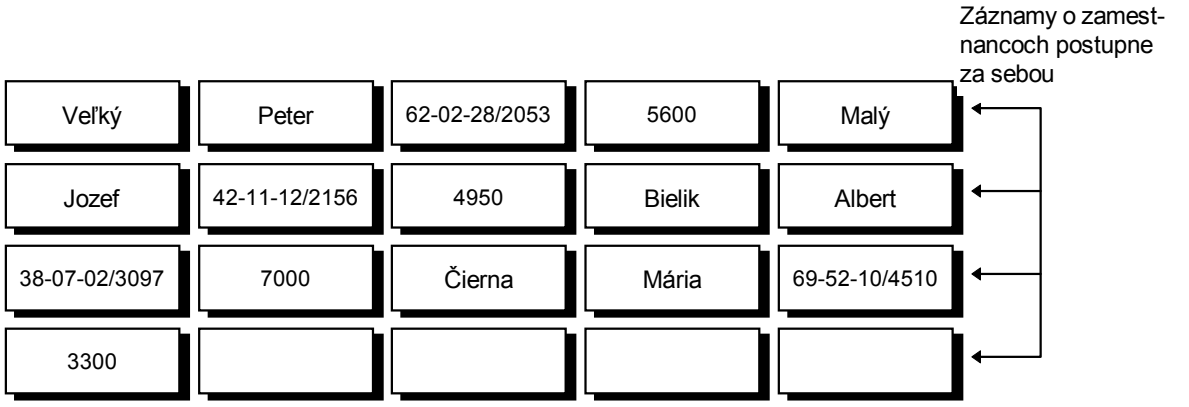
Postup pri vytváraní entitno-relačného modelu možno zhrnúť do nasledujúcich krokov:

1. Určia a pomenujú sa typy zobrazovaných objektov z reality a vzťahy medzi nimi, ktoré nás budú zaujímať.
2. Rozhodne sa o rozdelení typov objektov na typy entít a typy hodnôt.
3. Definičným oborom entitných vzťahov a atribútom sa podľa potreby priradia nové mená.
4. Stanovia sa identifikátory entít.

5. Pri entitných vzťahoch sa určí ich typ (1 : 1, 1 : *N*, *N* : *M*), pričom v prípade vzťahu typu *N* : *M* sa vzťah upraví na dva vzťahy typu 1 : *N* (resp. *N* : 1).
6. Pre názorné zachytenie modelu sa zostrojí jeho schematické grafické zobrazenie.

2.3.2 Systém pre správu súborov

Prvé informačné systémy, ktoré pracovali na báze agendového spracovania používali predchodcu dátového modelu - tzv. **systém pre správu súborov** (***File Management System - FMS***). Je to jednoduchý dátový model, ktorý **popisuje spôsob uloženia dát na disku**. V modeli FMS sú **všetky položky ukladané na disk za sebou** (sekvenčne) **do jedného** veľkého **súboru**. Ak v ňom chceme nájsť niektorú položku (slovo), aplikácia musí prehľadať celý súbor od začiatku po jednotlivých položkách, až narazí na hľadané údaje.



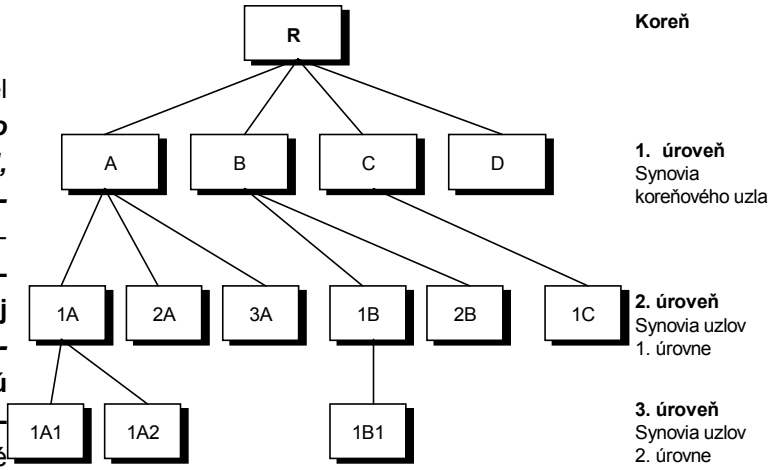
Obr. 6 Model FMS - systém pre správu súborov

Pri štúdiu príkladu tohto modelu (Obr. 6) možno nájsť viacero **nevýhod** tohto typu modelu:

- ***Medzi jednotlivými položkami nie je žiadny vzájomný vzťah okrem postupnosti ich uloženia***. Programátor (a niekedy aj užívateľ) musí preto presne vedieť ako sú dáta v súbore uložené, pretože inak s nimi nemôže manipulovať. Potreba znalosti ako operačný systém príslušného počítača fyzicky ukladá dáta na disku komplikuje prístup k dátam a veľmi sťažuje (až znemožňuje) prenos databázy a aplikácie na inú platformu. Naviac tieto detaily musí rešpektovať každá aplikácia čo zväčšuje priestor pre vznik chýb.
- FMS vytvára ***problémy s integritou dát***, pretože hodnoty všetkých položiek pred uložením na disk musia kontrolovať jednotlivé aplikácie. Do tej istej databázy môže pristupovať viac aplikácií a každá z nich môže pre jednotlivé položky pripúšťať čiastočne odlišné hodnoty. Zjednotenie prípustných hodnôt jednotlivých položiek pre rôzne aplikácie závisí len na dohode ich tvorcov..
- ***Neexistuje spôsob, ako rýchlo nájsť určitý záznam***; každé vyhľadávanie musí postupovať od začiatku a preverovať každý záznam, prípadne si SRBD môže pamätať ukazovateľ (logický alebo fyzický indikátor) na poslednú vyhľadanú položku, takže pri hľadaní ďalšej položky rovnakého typu nie je nevyhnutné začínať opäť od začiatku súboru.
- Najväčšou nevýhodou je ***komplikovaná zmena štruktúry databázy***. Ak by sme napríklad do záznamu o zamestnancovi chceli pridať novú informáciu (napr. počet rokov praxe), musel by SRBD prečítať každý záznam, zapísať ho do pomocného pracovného súboru a za koniec každého záznamu pridať novú informáciu. Po prečítaní celého súboru a zapísaní nového by sa musel pôvodný súbor zrušiť a pomocný pracovný súbor premenovať. Obdobný postup by sa musel urobiť aj v prípade, že by programátor chcel niektoré položky rozšíriť.

Réžia potrebná na uskutočnenie zmeny štruktúry je mimoriadne veľká a je sprevádzaná značným rizikom vzniku chýb, ktoré môžu celú databázu znehodnotiť.

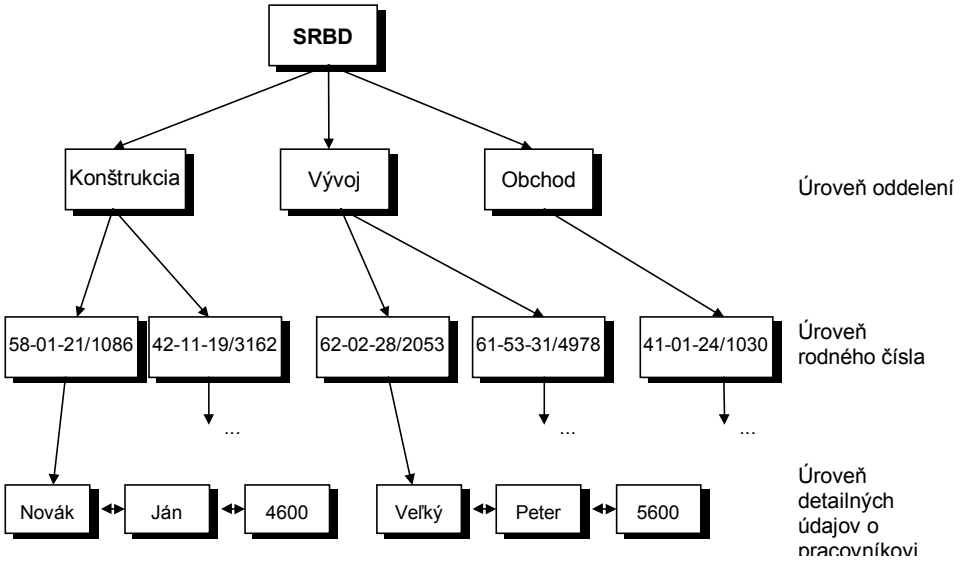
2.3.3 Hierarchický model



Obr. 7 Model hierarchického databázového systému

Hierarchický dátový model (HDM) ***predstavuje dáta ako množinu relácií 1:1 alebo 1:N, pričom vždy ide o vzťah nadriadenosti a podriadenosti*** (vlastník - člen). V tomto modeli sú **dáta organizované na základe stromovej štruktúry vychádzajúcej z *koreňa***. Jednotlivé **dátové štruktúry sú umiestnené na rôznych úrovniach** ležiacich pozdĺž ***vetiev***, ktoré vychádzajú z koreňa. Dátovým štruktúram na jednotlivých úrovniach sa hovorí ***uzly***, pokiaľ z uzla nevychádza ďalšia vetva, nazýva sa ***list***. Stromová štruktúra umožňuje definovať „*rodičovské*“ a „*súrodenecké*“ vzťahy medzi rôznymi prvkami v databáze. Je zrejmé, že v takomto modeli je podstatne jednoduchšie definovať vzťahy typu 1:*N* v porovnaní s FMS modelom. Rovnako aj vyhľadávanie dát je jednoduchšie a rýchlejšie. Ak chce užívateľ nájsť informáciu v položke 1B1 (Obr. 7), nemusí SRBD prehľadávať celý súbor, ale stačí, keď požiadavku rozloží na jednotlivé zložky. Stačí prehľadať len vetvu B a z položky 1B potom pokračovať na 1B1.

V hierarchickom dátovom modeli je vždy **práve jeden *koreňový uzol***, ktorého „vlastníkom“ je obvykle systém alebo SRBD. Ukazovatele z koreňa vedú nadol k uzlom 1.úrovne, kde začína vlastná databáza. Každý uzol 1.úrovne môže mať jeden alebo viac synovských uzlov 2.úrovne. V uvažovanom príklade Žilinskej bicyklovej fabriky uvažujeme ako uzly 1.úrovne jednotlivé oddelenia firmy, uzly 2.úrovne reprezentujú zamestnancov jednotlivých oddelení (identifikovaných jednoznačným údajom - v našom príklade rodným číslom). Synovské uzly identifikátora zamestnanca obsahujú položky s informáciami o jednotlivých zamestnancoch. Štruktúra uzlov 3.úrovne sa v príklade (Obr. 8) odlišuje od všeobecnej schémy hierarchického dátového modelu (Obr. 7), aby znázornila súrodenecké vzťahy medzi jednotlivými položkami. Prepojenie položiek 3.úrovne pozostáva z reťazca ukazovateľov smerujúcich od jedného listu k ďalšiemu (napr. od priezviska ku krstnému meno a od neho k platu). Takýmto spôsobom môžu byť prepojené uzly na ľubovoľnej úrovni.



Obr. 8 Ukážka databázy zamestnancov v hierarchickom dátovom modeli

V tomto modeli môže každý synovský uzol obsahovať ukazovatele na ľubovoľný počet súrodencov, ale len jediný ukazovateľ na rodičovský uzol, takže vzťah 1:*N* je len jednosmerný.

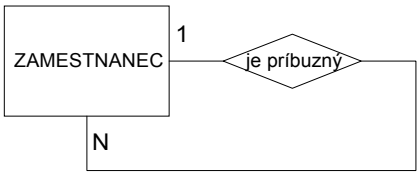
Na fyzickej štruktúre dát na disku v HDM nezáleží; SRBD ukladá obvykle ako zreťazený zoznam položiek s ukazovateľmi vedúcimi od otca k synom a od súrodenca k súrodencovi, pričom posledný list má nulový (koncový) ukazovateľ. Pridanie položiek na ktorejkoľvek úrovni je jednoduché - SRBD musí len koncový ukazovateľ zmeniť na ďalší súrodenecký uzol.

Hierarchický dátový model má však viacero **nevýhod** :

- Počiatočná **štruktúra databázy je daná pevne** a musí byť **definovaná pri vytváraní databázy**. Rodičovské vzťahy sa nedajú meniť bez úplného prepracovania celej štruktúry. Ak do štruktúry potrebujeme pridať ďalšiu úroveň, je nevyhnutné vytvoriť úplne novú štruktúru a potom skopírovať dáta z pôvodnej databázy do príslušných miest v novej databáze.
- Nevýhodou je „**strnulosť" HDM** - t.j. je obtiažne meniť definíciu úrovní. Napr. zmenu identifikátora zamestnanca - namiesto identifikácie rodným číslom, chceme zamestnancov identifikovať osobným číslom. Programátor musí opäť spracovať novú štruktúru s úrovňou „osobné číslo", ktorá nahradzuje úroveň „rodné číslo" a rodné čísla presunúť na úroveň detailných údajov o zamestnancovi.
- Najvýraznejším nedostatkom modelu je, že **neposkytuje jednoduchú metódu pre definovanie *krížových vzťahov*** (vzťah typu *M*:*N* - mnoho k mnohým). Príkladom takýchto vzťahov je situácia, kedy zamestnanec je nadriadený iným zamestnancom. Aby sme mohli definovať, komu sú pracovníci podriadení, bolo by treba zriadiť ďalšiu úroveň medzi názvom oddelenia a rodnými číslami, kde by boli vedúci. Pretože však každý vedúci je súčasne zamestnancom určitého oddelenia, vznikla by nelogická situácia, kedy by existoval rodičovský vzťah medzi vedúcim a ním samotným. Toto je možné riešiť aj tak, že záznam každého zamestnanca by obsahoval položku identifikujúcu jeho vedúceho - táto metóda obsahuje však veľa duplicitných údajov a je teda neefektívna.

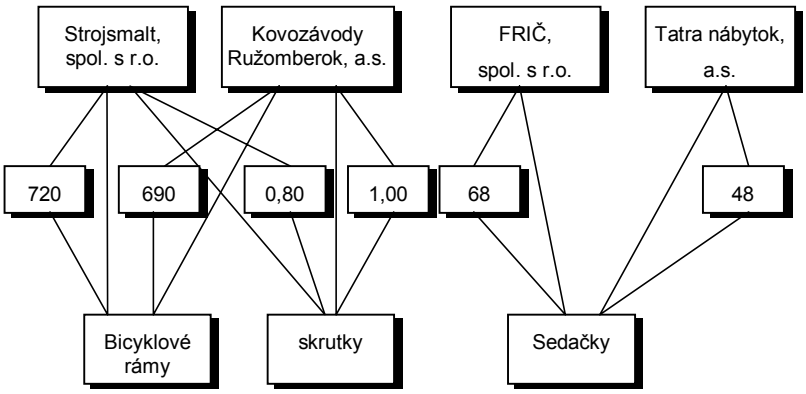
2.3.4 Sieťový model

Sieťový dátový model (*Network Database Model - NDM*) **predstavuje dáta ako množinu záznamov (entít) a párových vzťahov medzi nimi, ktoré môžu mať vzťah typu 1:1, 1:*N* alebo *N*:*M***. **Atribúty záznamov** v sieťových modeloch **môžu byť *jednoduché, opakujúce sa, zložené* alebo *zložené opakujúce sa***. Naviac, na rozdiel od predchádzajúceho hierarchického dátového modelu, **sieťový model môže obsahovať aj *cykly* a *slučky*.** Príkladom opakujúceho sa atribútu môže byť napríklad AUTOR v prípade knihy s viacerými autormi, ako zložený atribút možno rozumieť napríklad atribút ADRESA, pokiaľ v nej rozlišujeme jednotlivé časti (ULICA, PSČ, …). Cyklus môže vzniknúť, ak chceme mať viac prístupov k rovnakej entite. Príkladom slučky môže byť situácia, keď máme entitu ZAMESTNANEC a v podniku sú zamestnaní aj jeho príbuzní. Pri zavedení vzťahu „je príbuzný vznikne štruktúra zobrazená na obrázku (Obr. 9).



Obr. 9 Príklad slučky

Sieťový databázový systém je založený na ukazovateľoch (lineárnych alebo cyklických), ktoré vyjadrujú vzťahy medzi jednotlivými položkami. Ukážme si to na príklade skladovej databázy Žilinskej bicyklovej fabriky. Na obrázku (Obr. 10) je pomocou lineárneho tvaru NDS znázornená skladová evidencia, kde sú vyjadrené rôzne vzťahy medzi tovarom a jeho dodávateľmi, pričom sú registrovaní nielen dodávatelia jednotlivých tovarov, ale aj cena, za ktorú sú jednotlivé tovary dodávané.



Obr. 10 Príklad sieťového databázového modelu

Referent ŽBF môže zistiť dodávateľa bicyklových rámov alebo skrutiek tak, že požiada SRBD o prehľadanie množiny výrobkov a sleduje ukazovatele späť k ich výrobcom. Pokiaľ by chcel zistiť, čo všetko sa nakupuje od Kovozávodov Ružomberok, a.s., môže prehľadať množinu dodávateľov a sledovať ukazovatele k druhom tovarov, ktoré predávajú. Sieťový model umožňuje sledovať aj zložitejšie vzťahy - napr. sledovať cenu jednotlivých komponentov a pod.

Flexibilita NDS modelu pri vytváraní vzťahov typu N:M **je** jeho výraznou **výhodou**. **Vzájomné vzťahy medzi rôznymi množinami však môžu byť veľmi komplikované a obtiažne sa mapujú**. (Predstavme si databázu podľa Obr. 10 so stovkami dodávateľov, tisíckami druhov tovarov a cenami, ktoré závisia od objednaného množstva). To môže predstavovať pomerne veľkú komplikáciu, pretože väčšina databáz typu NDS vyžaduje pre sledovanie rôznych ciest, aby programátor zostavil príslušný program. Obdobne ako hierarchické databázy môžu byť aj sieťové databázy veľmi rýchle, zvlášť ak sa použijú tzv. indexové ukazovatele smerujúce priamo k prvej položke v prehľadávanej množine.

Nevýhody sieťového modelu sú obdobné ako u hierarchického modelu - ide hlavne o určitú **zviazanosť počiatočným návrhom štruktúry**. Keď už je databáza vytvorená, **každá zmena na inú štruktúru dát vyžaduje od programátora vytvorenie úplne novej dátovej štruktúry**. Sieťový model zjednodušuje pridávanie nových položiek do databázy a zmeny existujúcich položiek - programátor musí len definovať novú množinu a zmeniť potrebné ukazovatele, aby sa nová množina správne začlenila do zostávajúcich dát.

2.3.5 Relačný model

Relačný dátový model (*Relational Database model - RDM*) **organizuje údaje do** tzv. **usporiadaných n-tíc.** Relačnú databázu užívateľ vníma ako **sústavu v čase sa meniacich normalizovaných tabuliek s usporiadanými stĺpcami**. V RDM sa **každá položka stáva stĺpcom tabuľky** a **každý záznam jej riadkom**. Rôzne množiny z komplexného sieťového modelu sa stávajú rôznymi tabuľkami, ktoré identifikujú jednotlivé položky pomocou názvov stĺpcov v prvom riadku. Ako príklad si uveďme opäť skladovú databázu Žilinskej bicyklovej fabriky, spol. s r.o. usporiadanú tentoraz do relačného tvaru (Obr. 11).

DODÁVATELIA

D#	Firma	Pracovník
01	Strojsmalt spol. s r.o.	Peter Veľký
02	Kovozávody Ružomberok a.s.	Ignác Malý
03	Fric spol. s r.o.	Michal Martinec
04	Tatra nábytok a.s.	Vladimír Kováč

TOVAR

T#	Názov_tovaru	D#
1005	Bicyklové rámy	01
1281	Skrutky	02
1281	Skrutky	01
5210	Sedačky	04
1005	Bicyklové rámy	02
5210	Sedačky	03

CENA

T#	D#	Cena / ks	Min_mn.
1005	01	720,00	10
1005	02	690,00	40
1281	01	0,80	150
1281	01	1,00	10
5210	03	68,00	10
5210	04	48,00	40

Obr. 11 Príklad relačného databázového modelu

V každej tabuľke má jeden alebo viac stĺpcov rovnaký názov ako v inej tabuľke. Tieto spoločné stĺpce vytvárajú relácie medzi jednotlivými tabuľkami. Samotné názvy stĺpcov v jednotlivých tabuľkách nemusia byť nevyhnutne zhodné ako v našom príklade, stačí, aby dáta v spoločných stĺpcoch mali rovnaký typ a doménu.

Ak chceme nájsť určitý tovar od určitého dodávateľa, SRBD vyhľadá v tabuľke TOVAR názov tovaru, v stĺpci D# nájde číslo dodávateľa a vztiahne ho k obdobnému údaju v stĺpci D# v tabuľke DODÁVATELIA, kde nájde názov firmy a meno kontaktného pracovníka. Pre zistenie ceny tovaru sa využije relácia medzi stĺpcami T# a D# v tabuľke TOVAR a rovnomennými stĺpcami v tabuľke CENY. Jednotlivé relácie môžu byť aj kombinované: užívateľ môže požiadať SRBD o zobrazenie dodávateľa, názvu tovaru, ceny a minimálneho odberu, pre ktorý táto cena platí.

Na to, aby bolo možné dátové tabuľky efektívne ukladať do pamäti a manipulovať s nimi, **musia** tieto tabuľky **spĺňať** nasledujúce **vlastnosti**:

1. Každá tabuľka má v RDM svoj jednoznačný názov, ktorý ju v databáze identifikuje.
2. Každá tabuľka obsahuje len riadky (záznamy) rovnakého typu.
3. Každý stĺpec tabuľky má svoj názov - meno atribútu, ktorým je identifikovaný v RDM. Ak sa v rôznych tabuľkách vyskytujú rovnaké atribúty, môžu byť pomenované rôzne, ale praktickejšie je pomenovať ich rovnako, pretože ide o rovnaké veličiny. Pri rovnakom pomenovaní ich však musíme odlišovať príponou - t.j. kvalifikovať ich príslušnosťou k určitej tabuľke.
4. Každý stĺpec obsahuje hodnoty rovnakého atribútu a tieto hodnoty musia byť z domény skalárnych hodnôt rovnakého typu - t.j. napr. len samé číslice, len textové reťazce, len logické hodnoty a pod.
5. Každý riadok tabuľky zodpovedá jednému výskytu entity daného typu.
6. Každý riadok je jednoznačne identifikovateľný. Pre identifikáciu sa používa zvláštny atribút - tzv. primárny kľúč.
7. Na poradí stĺpcov a riadkov v tabuľke nezáleží.
8. Všetky hodnoty v danom riadku sú jednoznačne a úplne závislé na primárnom kľúči.
9. Každá bunka tabuľky (ak je obsadená) musí obsahovať len jednoduchú hodnotu príslušnej domény (t.j. nie opakujúcu sa skupinu hodnôt). Kľúčové hodnoty musia byť vždy obsadené - t.j. nesmú obsahovať NULL (Hodnota NULL indikuje, že príslušná hodnota nie je známa).

Celý RDM je tvorený sústavou tabuliek, ktoré majú uvedené vlastnosti. Na to, aby sme mali zaručenú integritu RDM, musí byť starostlivo vyriešená problematika kľúčov. Sledujeme dvojakú integritu RDM:

- ***integritu entít***, ktorá zaručuje, že každá entita bude v RDM jednoznačne identifikovateľná. To je dané tým, že žiadna z veličín tvoriacich primárny kľúč nebude mať hodnotu NULL.
- ***referenčnú integritu***, ktorá zaručuje, že sústava tabuliek bude navzájom prepojená s možnosťou spájať vzájomne hodnoty z navzájom súvisiacich tabuliek RDM.

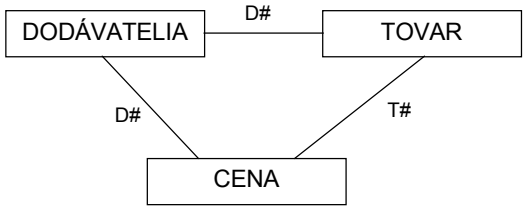
Ako už bolo uvedené skôr, **primárny kľúč slúži na jednoznačnú identifikáciu riadku v každej tabuľke**. Primárny kľúč **môže byť tvorený jedným alebo viacerými atribútmi** - hovoríme potom o ***jednoduchom*** alebo ***zloženom kľúči.*** Primárny kľúč musí spĺňať nasledujúce požiadavky:

- ***jedinečnosť***, t.j. v tabuľke sa nesmú vyskytnúť dve a viac rovnakých hodnôt kľúča;
- ***minimalizáciu***, t.j. v prípade, že kľúč je vytvorený zložením viacerých hodnôt, potom žiadna z týchto hodnôt nemôže byť vypustená bez porušenia zásady jedinečnosti kľúča.

Primárny kľúč **môže byť** v prípade potreby **tvorený aj všetkými hodnotami v riadku**.

Pri rozhodovaní o voľbe primárneho kľúča zvažujeme niekedy rôzne atribúty, ktoré by mali spĺňať funkciu kľúčových hodnôt. Hovoríme potom a tzv. možných kľúčoch (candidate keys). To býva v situácii, keď existuje viac identifikačných hľadísk - napr. zamestnanca môžeme identifikovať jeho osobným číslom (spravidla prideleným zamestnávateľom) alebo jeho rodným číslom. Keď sa pre jeden z možných kľúčov rozhodneme, druhý zostáva ako alternatívny kľúč.

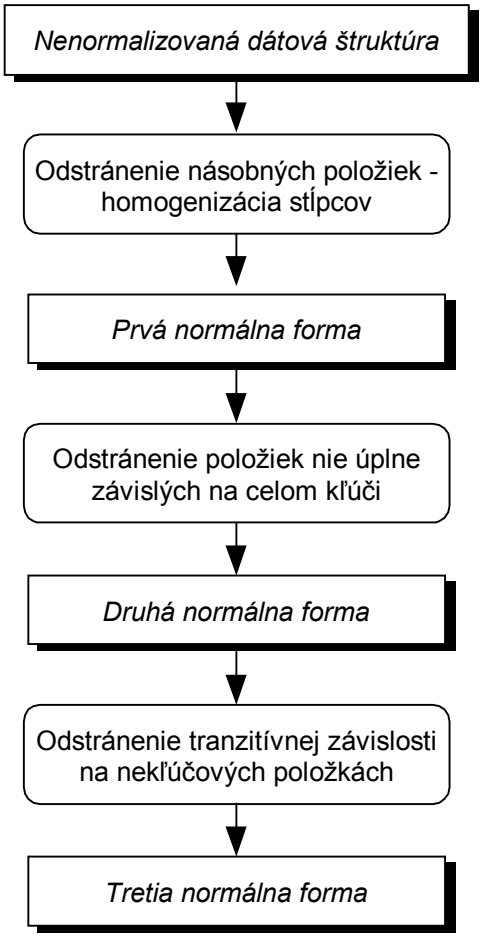
Ak chceme zabezpečiť skôr uvádzanú referenčnú integritu, musíme **umožniť spájať výskyty jednotlivých riadkov entít (riadky tabuliek), ktoré k sebe logicky patria.** Toto umožňujú tzv. ***cudzie kľúče*** (foreign keys). Ide vždy o primárne kľúče alebo ich časť z inej tabuľky. Pre uvažovaný príklad sú cudzie kľúče uvedené na spojniciach jednotlivých tabuliek (Obr. 12). **Problém referenčnej integrity spočíva nielen v prepojení všetkých tabuliek, ale aj v tom, aby toto prepojenie bolo sémanticky správne**. To znamená, že predpokladáme, že v tabuľke TOVAR sa neobjaví riadok s uvedením tovaru od dodávateľa, ktorý nemá záznam v tabuľke DODÁVATELIA. Ak by sa taký tovar skutočne objavil, SRBD musí takýto stav rozoznať a vyžiadať si riešenie. Obdobné problémy treba riešiť napríklad pri vyraďovaní niektorého tovaru z tabuľky TOVAR. V takom prípade sa musia zrušiť aj všetky odpovedajúce záznamy z tabuľky CENA.



Obr. 12 Relácie medzi tabuľkami

Závažným rozhodnutím, pred ktoré je postavený projektant databázy **je definovanie tabuliek a následne relácií medzi nimi**. **Proces dekompozície dát do podmnožín pre jednotlivé tabuľky sa nazýva *normalizácia***. Normalizácia **vychádza z požiadavky na efektívne ukladanie dát do pamäti**, t.j. **minimalizáciu nadbytočnosti** (redundancie) **pri zachovaní integrity a konzistencie databázy a pri zachovaní požadovaného informačného obsahu**. Pri **tom sa musí dbať o zaručenie efektívnej práce s RDM**, predovšetkým **jednoznačnosť a správnosť odpovedí na dopyty**. RDM definuje celkom **5 úrovní normalizácie**, pričom každá úroveň redukuje množstvo duplicitných dát. V praxi sa používajú len prvé 3 úrovne normalizácie. **Základnou myšlienkou normalizácie je postupný rozklad pôvodných nenormalizovaných tabuliek do sústavy viacerých menších tabuliek tak, aby sa dosiahlo zmenšenie nárokov na pamäť bez straty informácií pôvodne v tabuľke obsiahnutých**, t.j. s možnosťou kedykoľvek vytvoriť pôvodné tabuľky. Ide nám o tzv. ***bezstratovú dekompozíciu***. Vlastný postup dekompozície sa realizuje v jednotlivých normalizačných krokoch odpovedajúcich jednotlivým normálnym formám (Obr. 13):

- *Prvá normálna forma* (1NF) vyžaduje, aby jednotlivé stĺpce obsahovali vždy len hodnoty z homogénnej dátovej domény bez opakujúcich sa hodnôt (t.j. jedno políčko - jedna hodnota).
- *Druhá normálna forma* (2NF) vyžaduje, aby všetky nekľúčové hodnoty v riadku boli významovo úplne závislé na kľúčových hodnotách daného riadku.
- *Tretia normálna forma* (3NF) rieši problém tzv. tranzitívnej závislosti, kedy niektorá nekľúčová hodnota závisí od inej nekľúčovej hodnoty v danom riadku. Možno povedať, že RDM je znormalizovaný do 3NF, ak každý atribút je buď kľúčovým atribútom alebo je svojou hodnotou jednoznačne závislý na celom kľúči (t.j. všetkých kľúčových hodnotách).
- *Štvrtá* a *piata normálna forma* riešia niektoré špeciálne prípady viachodnotových závislostí. Pre praktické použitie spravidla nie sú potrebné a uspokojíme sa s normalizáciou do 3NF.



Obr. 13 Všeobecný postup normalizácie

TOVAR

T#	Názov_tovaru	D1	D2
1005	Bicyklové rámy	01	02
1281	Skrutky	01	02
5210	Sedačky	03	04

Obr. 14 Tabuľka po ďalšej normalizácii

Ako príklad normalizácie si pozrime tabuľku TOVAR (Obr. 11). Hoci stĺpce T# a „Názov tovaru“ obsahujú určité opakujúce sa (duplicitné) informácie, každý riadok je jedinečný, pretože hodnoty v stĺpci D# sú odlišné. Na túto tabuľku môžeme aplikovať ďalšiu úroveň normalizácie tým, že stĺpec D# rozdelíme na viac stĺpcov (Obr. 14). To ešte viac redukuje priestor potrebný na uloženie informácie o skladovanom tovare, aj keď to o niečo zväčšuje rozsah tabuľky CENY, v ktorej musí byť stĺpec D# nahradený stĺpcami D1 a D2. Tieto zmeny nijako neovplyvnia štruktúru tabuľky DODÁVATELIA.

V správne navrhnutom relačnom SRBD sa informácia o štruktúrach tvoriacich databázu zapisujú do zvláštnej množiny tabuliek, ktoré sa nazývajú *systémové tabuľky* alebo *databázový slovník*. Tieto informácie obsahujú dátové prvky ako názvy tabuliek, názvy stĺpcov v týchto tabuľkách a typy dát ukladaných do jednotlivých stĺpcov. SRBD zaobchádza s týmito systémovými tabuľkami rovnako ako s ostatnými tabuľkami, takže programátor ich môže prehľadávať, aby zistil názvy tabuliek v databáze alebo názvy stĺpcov v každej tabuľke.

Najdôležitejšou výhodou relačného modelu oproti hierarchickému alebo sieťovému je jeho veľká pružnosť (flexibilita). Zmena štruktúry databázy spočíva v jednoduchom pridaní stĺpca do tabuľky, jeho zrušení prípadne vytvorení novej tabuľky alebo zrušení existujúcej, čo minimálne ovplyvní ostatné tabuľky. Vďaka možnosti definovania vzťahov (relácií) medzi jednotlivými stĺpcami rôznych tabuliek je možné jednoducho vytvoriť novú tabuľku ako podmnožinu (projekciu) existujúcich tabuliek. To, že na zmenu štruktúry nie je potrebné nanovo vytvárať celú štruktúru databázy sa pozitívne odrazí na zachovaní integrity dát.

2.3.6 Objektovo orientované SRBD

Tento typ prístupu k výstavbe databázového systému je ovplyvnený technológiou ***objektovo orientovaného prístupu***, ktorá sa čoraz viac presadzuje pri tvorbe programov v súčasnosti. Hlavnou zmenou v myslení, ktorú prináša objektovo orientovaný (OO) prístup, je pohľad na dáta v databáze. Dáta nie sú dané len hodnotami zoskupenými do dátových štruktúr (ako je to napríklad v relačnom dátovom modeli), ale sú vztiahnuté k objektom, ktoré priamo zodpovedajú entitám reálneho sveta. Pojem ***objektu*** v OO prístupe zahŕňa aj chovanie objektu, ktoré je napr. v relačných systémoch dané aplikačnými programami pracujúcimi na reláciami a nie nad objektmi. Tým, že sa funkčnosť aplikácie prenáša priamo k objektom, možno redukovať vlastný kód aplikačných programov až na 20% [6].

Pre objektovo orientované SRBD (OOSRBD) sú charakteristické tieto vlastnosti:

- ***Identita objektov*** – v OOSRBD sa objekty chápu ako abstraktné objekty vyjadrené pomocou tzv. ***identifikátorov objektov***. Identifikátor zostáva stále rovnaký, mení sa len hodnota objektu, ktorá reprezentuje stav objektu. Čo robí objekt objektom, je jeho ***identita***, ktorá je nezávislá na hodnote objektu. Dva objekty môžu byť teda identické (ide vlastne o ten istý objekt) alebo sa môžu rovnať (majú rovnakú hodnotu). Napríklad objekt Novák typu HRÁČ, popísaný atribútmi odrážajúcimi situáciu v športovom klube môže byť ten istý objekt ako Novák typu ZAMESTNANEC, popísaný atribútmi odrážajúcimi situáciu u určitej firmy.

- ***Triedy a typy – Typ*** v OOSRBD sumarizuje spoločnú štruktúru množiny objektov s rovnakými charakteristikami. Okrem štruktúry sú zadané operácie, ktoré možno s objektmi daného typu vykonávať. Pojem ***trieda*** je viac ako typ a je v rôznych OOSRBD chápaný rôzne. Zahŕňa v sebe možnosť produkovať nové objekty, združovať všetky potrebné objekty spolu v tzv. ***kontejneri objektov***. Ku kontejneru môže užívateľ pristupovať pomocou daných operácií. Typ je teda vo všeobecnosti vztiahnutý skôr ku konceptuálnemu pohľadu, trieda je použitá pri implementácii typu.

- ***Zapuzdrenie –*** pojem vychádza z ponímania abstraktných dátových typov. Ide o princíp, že k objektom danej triedy (či ku kontejneru triedy) možno pristupovať len cez dané rozhranie a nie iným spôsobom, t.j. hodnoty atribútov objektu nie sú vo všeobecnosti prístupné z programovacieho jazyka. Užívateľ teda nemá inú možnosť, ako používať ponúkané operácie, čo vedie k vysokej ochrane dát.

  Objekt má rozhranie a implementáciu. Rozhranie je dané špecifikáciou operácií, ktoré môžu byť vykonávané na objekte. Operácia tvorí jedinú viditeľnú časť objektu. Implementácia má časť dátovú a časť procedúr, ktoré slúžia k reprezentácii stavu objektu a implementáciu každej operácie. Operácie sa v OOSRBD obvykle nazývajú ***metódy***, použitie metód sa deje prostredníctvom ***správ***, daných napr. možnosťou volania funkcií alebo príkazov daného jazyka. Namiesto vyvolávania procedúry sa hovorí o ***posielaní správ***. Metódy môžu byť spojené len s jednotlivým objektom, častejšie však s triedou objektov.

  Zapuzdrenie umožňuje meniť programátorovi dátovú časť bez toho, že by sa menili programy pracujúce so zodpovedajúcimi objektmi.

- ***Polymorfizmus*** – schopnosť operácií fungovať na objektoch viac ako jedného typu alebo patriacich do viac ako jednej triedy. Napríklad metóda realizujúca triedenie môže byť implementovaná jednou procedúrou ako pre atribút MENO_ZAMESTNANCA, tak pre atribút ADRESA_ZAMESTNANCA.

  Polymorfizmus sa rozdeľuje na ***univerzálny*** s potencionálne nekonečným oborom typov a ***ad hoc***, kde funkcie pracujú nad nejakou konečnou množinou typov.

- ***Dedenie –*** Objektovo orientovaný prístup umožňuje užívateľovi odvodzovať z existujúcich tried nové triedy. Nová trieda - ***podtrieda*** danej triedy alebo niekoľkých daných tried - ***dedí*** všetky atribúty a metódy existujúcej triedy, ktorá sa nazýva ***nadtrieda*** novej triedy. Užívateľ môže pre podtriedu špecifikovať prídavné atribúty a metódy. Trieda môže mať ľubovoľné množstvo podtried. V niektorých OOSRBD existuje obmedzenie, že trieda smie mať len jednu nadtriedu, v niektorých je prípustné tzv. ***viacnásobné dedenie***.

- ***Rozšíriteľnosť' –*** znamená možnosť definovať nové základné typy a následne nové typy pomocou konštruktorov pre vytváranie typov (externe definované typy). To vedie na požiadavku mať možnosť a vytvoriť efektívnu implementáciu takých typov. Tento pojem sa používa skôr vo funkčnom zmysle, t.j. ide o zahrnutie užívateľom definovaných operácií.

Súčasné trendy v rozvoji databázových systémov možno rozdeliť na dva základné smery. Jeden z nich - označuje sa ako ***revolučný*** **- sa snaží o dôslednú implementáciu objektovo orientovaného prístupu**. Je potrebné uviesť, že **v súčasnosti dostupné OOSRBD, zatiaľ nespĺňajú všetky požiadavky na ne kladené**. Druhý smer vývoja databázových systémov - nazýva sa ***evolučný*** - je **smerom rozširovania relačnej technológie**. Tento smer je **charakterizovaný rozširovaním dátových typov v súvislosti s masívnym rozvojom multimediálnych technológií (integrácia formátovaných dát, textov, zvuku a obrazu), rozvojom aplikácií využívajúcich vlastnosti tzv. hypertextu alebo CAD systémov, ktoré tiež vedú k definovaniu nových spôsobov implementácie dát v databázach**. Súčasťou tohto smeru je aj **dopĺňanie niektorých** (vhodných) **vlastností objektovo orientovaného prístupu** (napríklad doplnenie tzv. objektového rozhrania do relačného systému a pod.).