Uchovávanie dát

Zdroj: RUSKO, Miroslav - HALÁSZ, Jozef, 2011: *Environmentálne orientované informačné systémy*. - Žilina: Strix, Edícia EV-64, Prvé vydanie, ISBN 978-80-89281-76-3, 220 s.

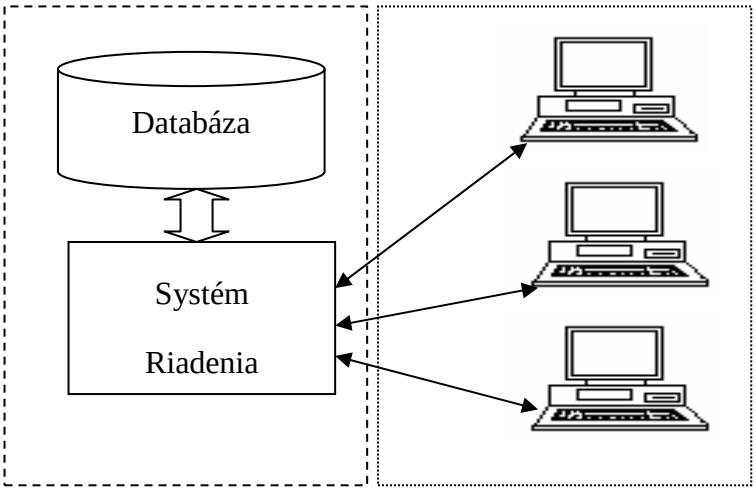
Základná charakteristika databázových systémov

Databázový systém (***DBS***) je organizačno-technický systém, umožňujúci ukladať veľký objem informácií v systematizovanej forme a umožňujúci výstup a použitie ľubovoľnej informácie, či už vo forme jednotlivých údajov alebo v spojeniach s inými údajmi. DBS je tvorený súborom navzájom spriahnutých dát (báza dát) a programami, umožňujúcimi jednotný prístup a manipuláciu s týmito dátami. Tieto programy tvoria systém riadenia databázy (***SRDB***). Základnou črtou tejto definície je oddelenie programového ovládania od databázy (DB), tzn. nezávislosť dát a programov.

Pre databázu existuje množstvo definícií, v ktorých sa zdôrazňujú rôzne aspekty:

- *Databáza* je súbor vzájomne prepojených dát usporiadaných podľa dátových schém, ktorý slúži jednej alebo viacerým aplikáciám. Dáta sú uložené tak, aby mohli byť využívané viacerými programami a tieto programy neboli závislé na konkrétnej štruktúre dát a ich fyzickom uložení.
- *Databáza* je súbor formalizovaných, strojom spracovateľných obsahov, ktorý slúži ako základňa a východisko pre ukladanie, spracovávanie a využívanie informácií.
- *Databáza* je dátová štruktúra pre príjem a uschovanie dát, ktoré sa na požiadanie poskytujú viacerým nezávislým používateľom (ISO).
- *Databázu* tvorí množina navzájom prepojených dát, slúžiacich mnohým aplikáciám, uložených v pamäti spôsobom vylučujúcim nežiaducu redundanciu.
- *Databáza* je skupina dát, vzťahujúcich sa k danej téme a usporiadaná tak, aby sa dala využívať používateľmi (ISO).

Systém riadenia bázy dát (***SRBD***) je programové vybavenie, ktoré riadi databázové operácie súvisiace s uchovávaním, správou a zabezpečením dát.



Obr. 1 Principiálna štruktúra databázového systému

Báza dát musí obsahovať také údaje o objekte záujmu a jeho okolí (používa sa tiež všeobecnejší pojem - realita), aby v nej boli zachytené všetky skutočnosti potrebné pre používateľov databázového systému. Dáta pritom musia odrážať tieto skutočnosti verne a v časovom súlade s vývojom reality. Pre zobrazenie reality pri navrhovaní databázových systémov sa používajú dátové modely. Dátový model vytvára množina pojmov, ktoré uľahčujú špecifikáciu štruktúry dát a umožňujú vytvárať operácie pre manipuláciu s dátami.

Základné požiadavky, ktoré databázový systém musí zabezpečiť:

- oddelenie definície dát a príkazov na manipuláciu s nimi
- nezávislosť dát,
- minimalizáciu redundandancie dát,
- ochranu proti nekonzistencii dát,
- zdieľanie dát,
- bezpečný prístup k dátam,
- integritu dát.

Oddelenie definície dát od programov, ktoré manipulujú s týmito dátami je základnou myšlienkou pri tvorbe databázových systémov. Databázový prístup rozlišuje medzi popisom dát a manipuláciou s údajmi. Všetky objekty musia byť popísané skôr, ako budú vložené, modifikované alebo nakoniec zrušené.

Nezávislosť dát je dôležitá z hľadiska flexibility zmien. Používateľ by nemal vôbec zbadať, že sa niečo zmenilo v konceptuálnom dátovom modeli, alebo v spôsobe uloženia, resp. prístupu k dátam.

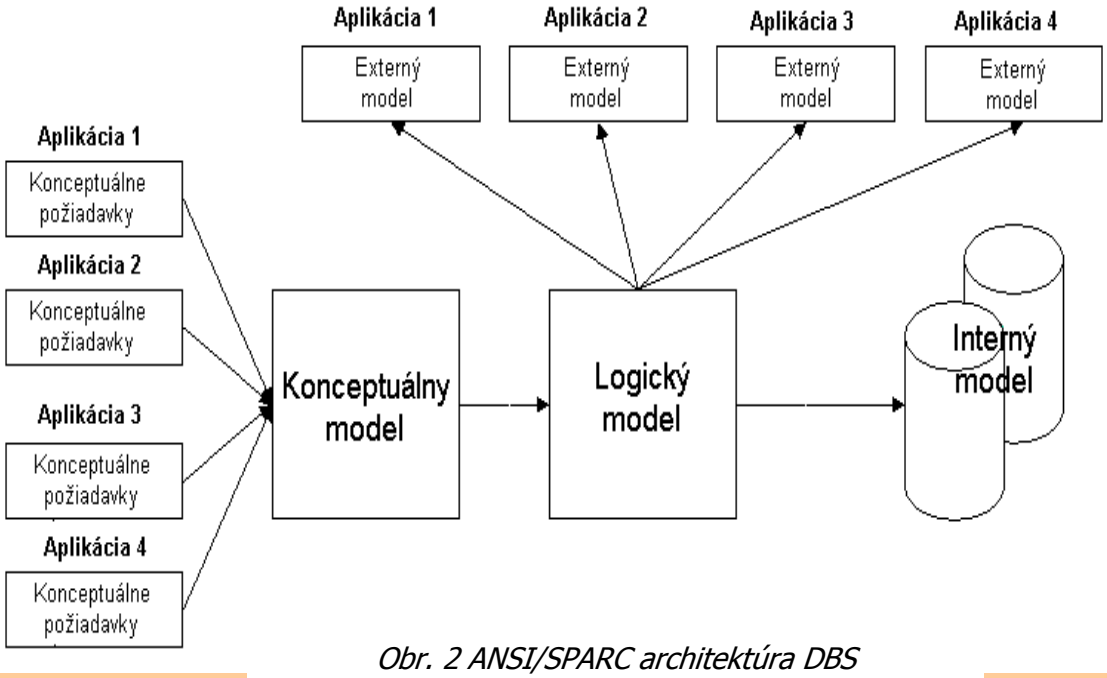
V praxi sa stretávame často s viacnásobným výskytom tých istých údajov umiestnených v rôznych súboroch (tzv. redundandné údaje). Udržiavanie takýchto systémov je náročné na čas i na súvisiace náklady, hlavne v prípade zmien týchto dát je nutné aplikovať zmeny vo všetkých výskytoch.

Dáta uložené v databáze musia vyhovovať definovaným podmienkam, ktoré by sme mohli súhrne nazvať integritné obmedzenia (napr. vek by mal byť v rozsahu 1-130, apod.).Okrem toho, medzi dátovými objektmi môžu existovať vzťahy, ktoré musia byť zabezpečené, hlavne pri vstupe dát, opravách dát, alebo pri ich rušení.

Architektúra DBS

V štandardizovanom databázovom modely (ANSI/SPARC architektúra) sa rozlišujú tri úrovne (pohľady):

- ***Konceptuálna štruktúra.*** Prezentáciou celého informačného obsahu databázy je konceptuálna štruktúra alebo konceptuálny model, ktorý je nezávislý na konkrétnom technickom vybavení, ale zohľadňuje rôzne externé pohľady používateľov. Konceptuálna úroveň je realizovaná vhodnou konceptuálnou schémou. Hlavným nástrojom tvorby konceptuálnej schémy sú entitnorelačné diagramy.
- ***Interná (fyzická) štruktúra.*** Pri tvorbe internej štruktúry databázy sa rieši problém fyzického ukladania dát v pamäti, adresácie, vytvárania blokov, smerníkov apod. Sú to problémy úzko spojené s konkrétnym počítačovým a operačným systémom. Koncový používateľ 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.



Obr. 2 ANSI/SPARC architektúra DBS

- ***Externá (logická) štruktúra.*** Zohľadnením požiadaviek používateľov (aplikácií) sa vytvára externá štruktúra databázy, ktorá sa opisuje použitím špecializovaných jazykov. Ich pomocou sa definujú dátové štruktúry, vzťahy medzi nimi a spôsob manipulácie. Ide o kombináciu dvoch jazykov - tzv. Data Definition Language (***DDL***) a Data Manipulation Language (***DML***), pomocou ktorých používateľ deklaruje svoje dátové objekty a vzťahy medzi nimi a pomocou ktorých rieši svoje úlohy s týmito dátami. Pretože používateľov je spravidla viac a obvykle chcú pracovať s dátami z rôzneho pohľadu a s inými

požiadavkami na ne, môže existovať viacej používateľských externých pohľadov[1].

Rozdelenie databázových systémov

DBS sa delia podľa rozličných hľadísk:

A. Podľa spôsobu použitia:

- ***analytické databázy*** (známe aj pod názvom OLAP- On Line Analytical Processing), ktoré by sa dali charakterizovať ako statické, určené najmä na sprístupňovanie archívnych dát. Sú používané pri analytických prácach, pri určovaní dlhodobých trendov apod. Príkladom môžu byť katalógy knižníc, ktoré sa síce priebežne doplňujú, ale prevládajúcou operáciou je vyhľadávanie.
- ***operatívne databázy*** (známe aj pod názvom ***OLTP*** - On Line Transaction Processing) sa vyznačujú dynamicky sa meniacim obsahom, rôznorodosťou transakcií. Typickým príkladom môžu byť obchodné aplikácie, skladové hospodárstvo apod.

B. Podľa usporiadania spracovateľských komponentov systému hovoríme o architektúre:

- *host-terminal*, kde spracovanie dát vykonáva skoro výlučne hostiteľský počítač (host). Terminály slúžia len na komunikáciu s používateľmi, príp. majú veľmi obmedzené možnosti lokálneho spracovania dát. Táto architektúra sa vyvíjala pre alfanumerické terminály. Má malú flexibilitu vytvorených aplikácií a vyžaduje veľké zriaďovacie náklady. Výhodou je centrálne zabezpečenie zdrojov a schopnosť uloženia obrovského množstva dát. Umožňuje súčasnú prácu mnohých používateľov.
- *file-server* (klient-klient), ktorej charakteristickým rysom je oddelené umiestnenie dát a spracovávajúcich programov. Dáta sú uložené na súborovom serveri, ale spracovanie vykonávajú pripojené používateľské stanice. Záporným rysom tejto architektúry sú veľké nároky kladené na komunikačnú sieť po ktorej sa prenášajú veľké súbory dát, aby mohli byť následne spracované lokálnymi stanicami.
- *klient-server*, ktorá je založená na myšlienke rozdelenia kompetencií medzi programovými procesmi spracovania dát. Proces server (back-end) obstaráva všetky databázové operácie súvisiace s uchovávaním, správou a zabezpečením uložených dát. Proces klient (front-end) komunikuje s používateľom, prijíma jeho požiadavky, odovzdáva ich vo vhodnej forme databázovému serveru a prezentuje výsledky v žiadanej podobe. Výhodou tejto architektúry je pomerne ľahká rozšíriteľnosť a možnosť zvýšenia výkonnosti výmenou databázového stroja.

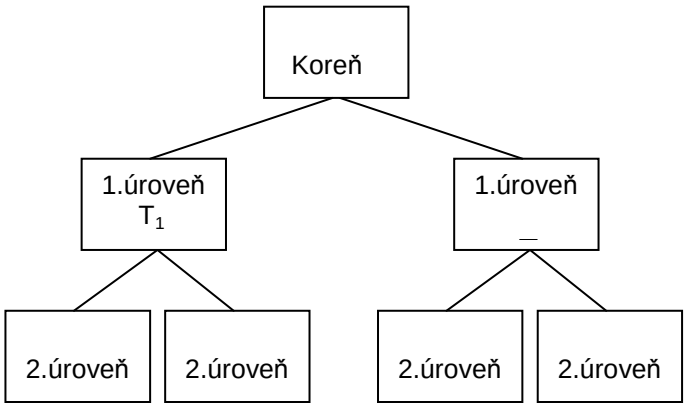
C. Podľa logického dátového modelu:

- Hierarchické DBS sú založené na stromových štruktúrach, v ktorých sa môžu vyskytovať iba vzťahy typu 1:1 alebo 1:M. Strom je konečná množina T, ktorá sa skladá z jedného alebo viacerých uzlov takých, že:

[1] HALÁSZ, J. – KRÁLIKOVÁ, R., 2002. *Environmentálne informačné systémy*. Košice : Vienala

- *má jeden špeciálny uzol, ktorý sa nazýva koreň stromu,*
- *ostatné uzly okrem koreňa obsahujú m≥0 rôznych navzájom sa prekrývajúcich množín uzlov T1, T2,...,Tm, z ktorých každá tvorí strom,*
- *stromy T1, T2,...,Tm sa nazývajú podstromy daného koreňa.*

Hierarchická štruktúra má niekoľko nevýhodných vlastnosti, ktoré sa prejavujú pri aktualizácii bázy dát, vo zvýšenej redundancii, ale aj pri vykonávaní bežných transakcií. Predpokladom používania hierarchickej databázy je podrobná znalosť štruktúry, pretože k údajom na nižších úrovniach vedie cesta len cez príslušné uzly stromu. Pri aktualizácii hrozí, že odstránením niektorých údajov odrežeme celé sústavy vetví. V jednotlivých podstromoch sa budú opakovať tie isté údaje. Táto redundancia vyplýva v danej štruktúry a nedá sa odstrániť.



Obr. 3 Hierarchická stromová štruktúra

- Sieťové DBS. V sieťovej štruktúre sa dajú realizovať okrem vzťahov 1:1 a 1:M, vzťahy N:M, čiže každý prvok dát môže byť zviazaný s ľubovoľným iným prvkom. Pri vyhľadávaní dát sú možné rôzne cesty, je možné odstrániť duplicity vlastné hierarchickému usporiadaniu.
- Relačné DBS. Na relačných modeloch je založená väčšina dnešných DBS, preto sa budeme venovať ich problematike osobitne.

Z uvedených typov sa považujú za tradičné:

- hierarchické,
- sieťové
- a relačné DBS.

V podstate boli všetky tieto systémy vyvinuté pre oblasť obchodu a financií. Spoločnými vlastnosťami týchto systémov sú:

- Uniformita: veľký počet rovnako štruktúrovaných dátových položiek rovnakých rozmerov.
- Orientácia na záznamy: základné dátové položky pozostávajú zo záznamov rovnakej dĺžky.
- Malé dátové položky: všetky záznamy sú pomerne krátke.
- Atomické polia: jednotlivé polia záznamov majú pevnú a pomerne malú dĺžku.
- Krátke transakcie: zlomky sekundy. Počas vykonanie transakcie nie je interakcia s obsluhou.

- Statické konceptuálne schémy: štruktúra databázy sa mení len výnimočne. Sú umožnené len jednoduché úpravy, napr. vytváranie a rušenie relácií, pridávanie a odoberanie atribútov.

Medzi novšie aplikácie patria:

- DBS pre CAD, CAM, CAE, CIM.
- DBS pre CASE.
- Multimediálne DBS, ktoré obsahujú priestorové údaje, audio a video dáta.
- Kancelárske systémy.
- Hypertextové databázy.

Spoločnými vlastnosťami týchto systémov sú:

- *Komplexné objekty*: štruktúra dátových objektov sa prispôsobuje objektom reálneho sveta, atribúty a relácie sú mnohokrát zložité, hierarchicky členené.
- *Behaviorálne dáta*: podrobné opisy pravidiel chovania systému. V reálnych podmienkach môže ten istý príkaz vyvolať rôzne reakcie objektov v závislosti na stave systému. Rieši sa to uložením potrebných procedúr (metód) spolu s údajmi.
- *Metadáta*: komplexnosť dátových modelov vyžaduje opis štruktúry použitím metadát a metamodelov.
- *Dlhotrvajúce transakcie*: v aplikáciách CAD a CASE sa vyžaduje interakcia s operátorom v priebehu vykonania transakcie. Môžu vznikať kolízne situácie, ktorých riešenie vyžaduje používanie vnorených transakcií a paralelné procedúry.

Relačné databázové systémy

Najrozšírenejšie sú dnes relačné databázové systémy. Báza dát v relačnom modeli je konečná množina v čase premenných konečných relácií. Zmena v čase spočíva v možnosti pridania alebo vynechania jedného alebo viacerých riadkov v niektorých reláciách alebo v zmene niektorých hodnôt už existujúcich riadkov niektorých relácií.

Základná myšlienka riešenia relačných databázových systémov je jednoduchá, založená na zoskupovaní dát do tabuliek. V roku 1982 E. F. Codd definoval tzv. minimalistické požiadavky pre relačný databázový systém (RDBS):

- všetky dáta sú uložené v relačných tabuľkách,
- neexistujú viacnásobné prístupové cesty,
- je k dispozícii databázový jazyk umožňujúci selekciu, projekciu a spojenie.

Pre praktické použitie môžeme základné vlastnosti relačnej databázy zhrnúť takto:

- Každá tabuľka RDBS má svoj jednoznačný názov.
- Každá tabuľka obsahuje len riadky (záznamy) rovnakého typu, pričom každý riadok odpovedá jednému výskytu entity daného typu.
- Každý stĺpec tabuľky má svoj názov - meno atribútu a obsahuje jeho hodnoty.
- Nezáleží na poradí pomenovaných stĺpcov v tabuľke.
- Každý riadok je jednoznačne identifikovateľný tzv. primárnym kľúčom.
- Nezáleží na poradí riadkov v tabuľke.

V správne navrhnutých DBS je veľmi dôležitá integrita entít, ktorá zaručuje ich jednoznačnú identifikovateľnosť. Primárny kľúč preto nikdy nemôže mať hodnotu NULL, čiže jeho hodnota musí byť vždy definovateľná. Rovnako dôležitá je aj referenčná integrita zaručujúca možnosť vzájomného spájania hodnôt z ľubovoľných tabuliek RDBS. Tu nám pomáhajú tzv. cudzie (nevlastné) kľúče, umožňujúce spájať výskyty jednotlivých typov entít (riadky tabuliek), ktoré spolu logicky súvisia. Vždy sa jedná o primárny kľúč alebo jeho časť z inej tabuľky.

Problém referenčnej integrity má aj svoju sémantickú stránku. Nemôžeme napr. jednoducho vymazať z určitej tabuľky záznam, ktorý je významovo dôležitý v iných tabuľkách.

Schéma relačnej bázy dát je konečná množina relačných schém: $R_1(A_1:D_1)$, $R_2(A_2:D_2)$,..., $R_n(A_n:D_n)$, spolu s množinou integritných obmedzení F.

Podobne, ako je konkrétna relácia (tabuľka) inštanciou určitej relačnej schémy, je aj relačná báza dát aktuálnou inštanciou svojho typu, ktorým je schéma relačnej bázy dát.

Pri práci s relačnou bázou dát sa používajú množinové a relačné operácie.

Množinové operácie sú:

- *Zjednotenie* dvoch relácií $A \cup B$ je množina všetkých riadkov, ktoré patria do *A* alebo *B*.
- *Prienik* dvoch relácií $A \cap B$ je množina všetkých riadkov, ktoré patria do *A* a zároveň aj do *B*.
- *Rozdiel* dvoch relácií $A - B$ je množina všetkých riadkov, ktoré patria do *A* a nepatria do *B*.

Kartézsky súčin dvoch relácií $A \times B$ je množina všetkých riadkov m, takých pre ktoré platí: $A \times B = \{m = a \,.\, b \mid a \in A \cap b \in B\}$.

Medzi relačné operácie patria:

- *Projekcia* $\Pi_s(E)$ (kde s je zoznam atribútov) je operácia na vytvorenie „vertikálnej" podmnožiny z relácie. Vykonáme ju výberom špecifikovaných atribútov a odstránením nepožadovaných. Po vykonaní projekcie je potrebné odstrániť opakujúce sa riadky v novej relácii.
- *Selekcia* $\sigma_p(E)$ (kde p je predikát) je operácia na vytvorenie „horizontálnej" podmnožiny z relácie. Slúži pre vyselektovanie riadkov relácie na základe hodnoty atribútov. Výsledná relácia je množina riadkov, ktorých atribúty spĺňajú stanovenú podmienku.
- *Spojenie* relácie *A* na atribúte *X* s reláciou *B* na atribúte *Y* t.j. $A\,[\,X=Y\,]\,B$.

Konzistencia a integrita dát

Konzistencia databázy v podstate znamená, že uložené údaje odpovedajú skutočnosti. V teórii RDBS sa stretávame aj s užšie ponímanou definíciou konzistencie, ktorá vyžaduje, aby databáza bola v súlade so zadanými integritnými obmedzeniami (***IO***). Je zrejmé, že z prvého významu plynie druhý, ale nie naopak. Pod integritou (celistvosťou) bázy dát rozumieme jej pohotovú fyzickú disponibilitu, pričom báza dát je v konzistentnom stave.

IO sú logické výroky, ktoré majú platiť o údajoch v databáze. Mali by byť odvodené z konceptuálnych modelov, ktoré opisujú objekty reálneho svata. Z logického hľadiska je IO napr. formula predikátového kalkulu rádu.

Jednoduché IO v personálnej báze dát môžu byť napríklad:

Vek > 2
1000 < Mzda < 50000
Stav = "Slobodný" $\vee$ Vek > 15

Základným typom integritného obmedzenia je referenčná integrita, ktorá popisuje vzťah medzi údajmi obsiahnutými v dvoch reláciách. Jedna tabuľka je pri referenčnej integrite vždy hlavná a jedna vedľajšia. Hlavná tabuľka väčšinou odpovedá typu entity, vedľajšia tabuľka typu vzťahu. Atribút, ktorého sa referenčná integrita týka vo vedľajšej tabuľke sa nazýva cudzí kľúč (foreign key). Ide o atribút (skupinu atribútov), ktorého hodnota v n-tici relácie je buď prázdna, alebo musí byť obsiahnutá ako hodnota primárneho kľúča hlavnej relácie. Existencia prázdnych hodnôt (NULL) vychádza z praktických potrieb. Nie vždy sú dáta v n-ticiach relácií známe, či definované.

Integrita bázy dát sa môže narušiť rôznymi operáciami pri aktualizácii, pri transakciách. Transakcia je postupnosť operácií nad objektmi bázy dát, ktorá realizuje jednu ucelenú operáciu z hľadiska používateľa bázy dát. Ochranu integrity bázy dát pri vykonávaní transakcií zabezpečuje systém riadenia DBS.

Normalizácia

Normalizácia vychádza z požiadavky efektívneho ukladania dát, minimalizuje nadbytočnosť pri zachovaní integrity a konzistencie bázy dát. Základnou myšlienkou procesu normalizácie je postupný rozklad pôvodných tabuliek do viacerých menších tabuliek tak, aby sa znížili nároky na pamäť bez straty informácie. Takáto dekompozícia musí umožniť aj spätnú rekonštrukciu pôvodných tabuliek.

Nech R(A) je schéma relácie $\Re$.

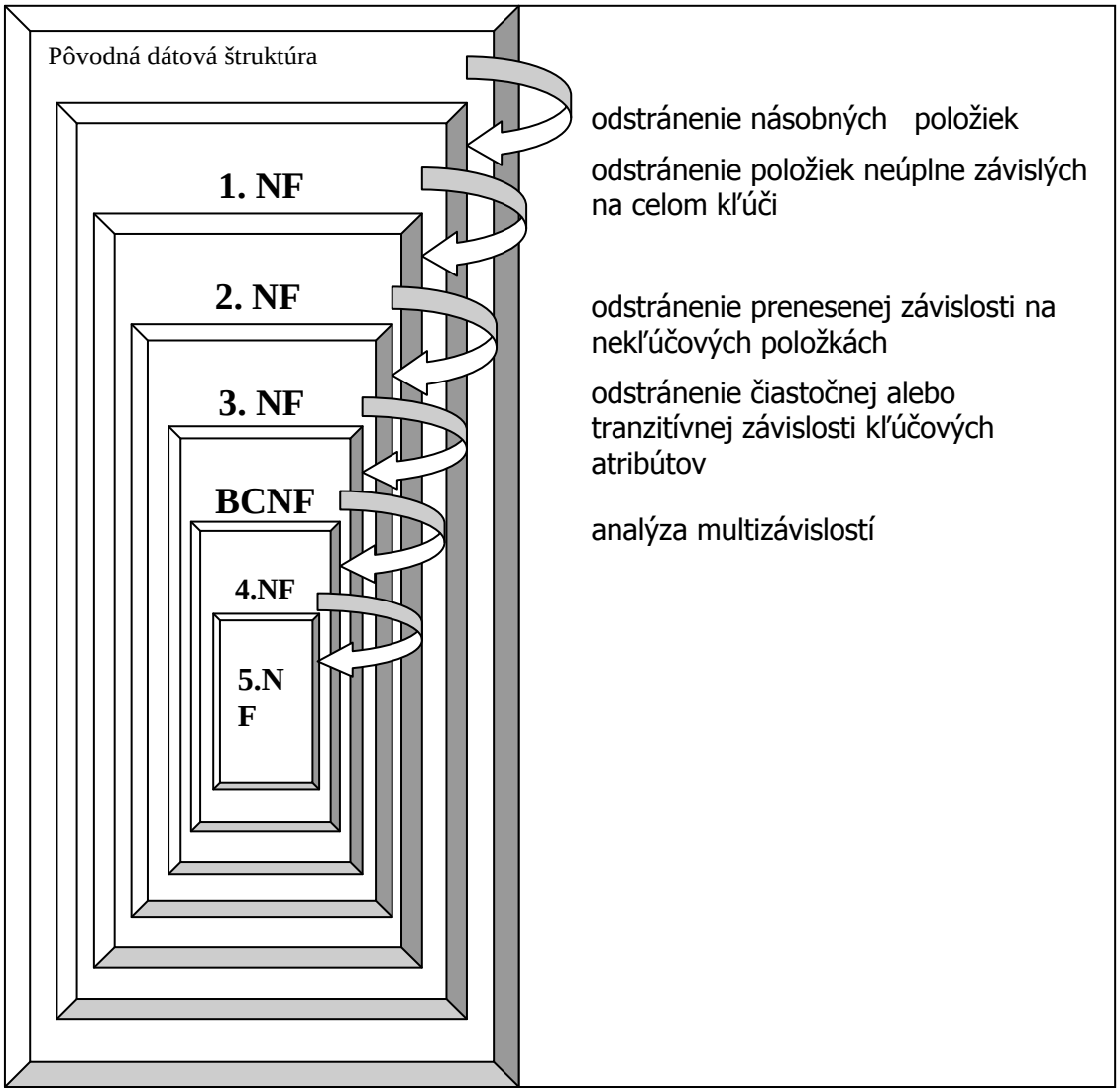
Množina relačných schém $\{R_1(A_1),R_2(A_2), \ldots ,R_n(A_n)\}$ je dekompozíciou schémy R(A), ak $A = A_1 \cup A_2 \cup \ldots \cup A_n$.

Vlastnú dekompozíciu vykonávame po krokoch, v ktorých transformujeme tabuľky RDBS do tzv. normálnych foriem. Jednotlivé normálne formy kladú na štruktúru a obsah tabuliek stále prísnejšie kritéria, súvisiace hlavne s nadbytočnosťou a funkčnou závislosťou atribútov.
Codd pôvodne definoval tri normálne formy. Neskôr k ním pribudla modifikácia tretej normálnej formy, tzv. Boyce-Coddova normálna forma. Pre špecifické prípady sa dá použiť štvrtá a piata normálna forma.

Nadbytočnosť

Nadbytočné (redundandné) údaje spôsobujú narastanie veľkosti databázy a navyše spôsobujú rôzne typy anomálií. Pozrieme sa bližšie, čo vlastne redundancia znamená. Začneme tým, že atribúty rozdelíme do troch skupín:

- Atribúty slúžiace výlučne pre identifikačné účely, kľúčové atribúty.
- Atribúty nekľúčové, ktoré v danej tabuľke majú len informatívny obsah.
- Atribúty použiteľné pre oba účely, tzv. kandidáty kľúčov



Obr. 4 Schematický postup normalizácie RDBS

Funkčné závislosti

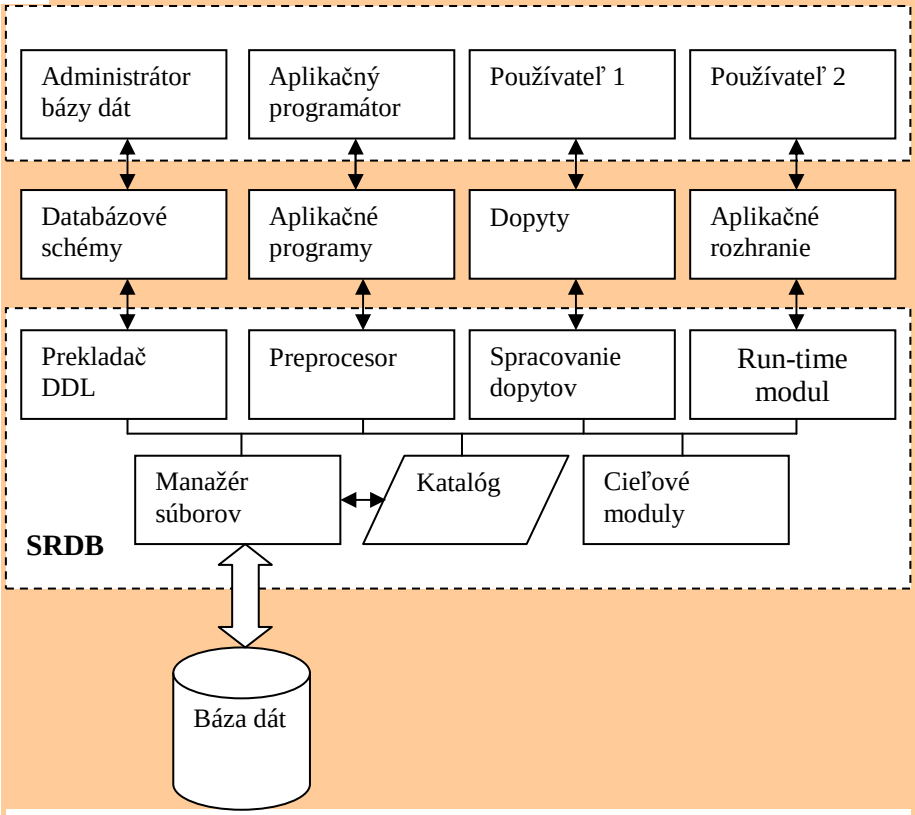
Medzi atribútmi každej relačnej tabuľky existujú väzby, ktoré budeme nazývať funkčnými závislosťami. Využívajú sa najmä pre identifikáciu konkrétnych entít.

Na základe hodnôt týchto atribútov vieme jednoznačne určiť ďalšie údaje autora alebo vydavateľa.

Systém riadenia relačnej bázy dát

Systém riadenia bázy dát (***SRDB***), anglicky Database Management System (***DBMS***), predstavuje súbor programov, pomocou ktorých sa vykonáva riadenie nad danými, jednotne organizovanými bázami dát. Tieto programy umožňujú najmä:

- vytváranie bázy dát, definovanie štruktúry bázy dát,
- aktualizáciu bázy dát, manipulácia s údajmi,
- cielený výber z bázy dát, tvorba dopytov.
- SRDB znižuje závislosť dát od aplikačného prostredia, organizuje spoločné využívanie tých istých dát rôznymi programami (používateľmi), zaručuje celistvosť, integritu a konzistenciu bázy dát a taktiež utajenie a ochranu pred zneužitím, čo je mimoriadne dôležité najmä v sieťovom prostredí.



Obr. 5 Základné komponenty systému riadenia bázy dát

Používateľ komunikuje s DBS cez používateľské rozhranie, ktoré poskytuje rôzne formy prístupu k jednotlivým programom SRDB a ich prostredníctvom k údajom v báze dát. S bázou dát bezprostredne spolupracuje súborový (dátový) manažér. Tento program eviduje fyzické umiestnenie dát v operačnej pamäti počítača alebo na diskovej pamäti. Je zodpovedný za efektívne ukladanie dát a zohľadnenie špecifických vlastností pamäťových jednotiek. Umožní ďalším častiam SRDB pracovať už len na úrovni logických adries. Je napojený na katalóg dát, ktorý obsahuje tak údaje o štruktúre dát (metadáta), ako aj informácie o ich aktuálnom umiestnení.

Súčasné SRDB ponúkajú správcom a používateľom DBS

- jazyky pre definovanie dát ***DDL*** (Data Definition Language),
- jazyky pre manipuláciu s dátami ***DML*** (Data Manipulation Language),

- pre komfortné vyhľadávanie dát slúžia dopytovacie jazyky (Query Language). Väčšinou sa jedná o neprocedurálne jazyky, pomocou ktorých používateľ formuluje len svoje požiadavky, ale neurčuje spôsob vykonania.

Ďalšie moduly SRDB slúžia na generovanie výstupných zostáv, formulárov a pohľadov. Okrem toho SRDB by mal mať nástroje, ktoré umožnia monitorovať výkonnosť databázy a nastavovať parametre, prípadne meniť spôsob uloženia dát a prístupových metód k dátam z hľadiska efektívnosti spracovania dát, alebo ich ochrany.

SRDB môžu byť jedno- a viacužívateľské (single/multiuser). Vo viacužívateľských SRDB je nutné zabezpečiť individuálny súbežný prístup pre viacerých abonentov, spolu s adekvátnou ochranou dát.

Najrozšírenejšími SRDB sú: Informix, INGRES, PROGRESS, ORACLE, DB2, dBASE, FoxPro, FoxBase, Paradox, Access.

Databázové jazyky

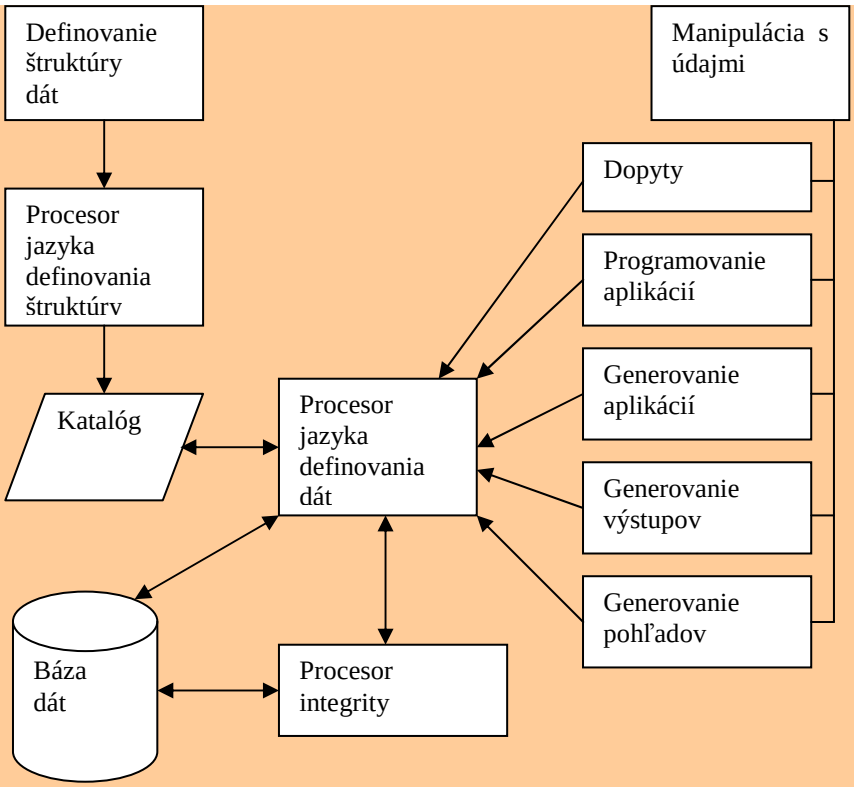
Základnými modulmi SRDB sú prekladače definičného, manipulačného a dopytovacieho jazyka. Prekladač definičného jazyka vytvára na základe deklarácií dátových typov vhodnú reprezentáciu dát v pamäti. Pomocou príkazov tohto jazyka sa vytvára katalóg dát alebo sa upravujú jeho časti.

Pri preklade príkazov manipulačného jazyka sa často používa tzv. ***hostiteľský jazyk*** (niektorý z vyšších programovacích jazykov, napr. C++). V tomto jazyku sa uplatňujú predspracované programové moduly.

Jazyk SQL

Komplexným riešením je vytvorenie združeného prekladača pre uvedené jazyky, ktorý by umožnil aj spracovanie dopytov a generovanie výstupov. Do tejto podoby sa dostali v posledných rokoch prekladače jazyka SQL.

SQL (Structured Query Language) je neprocedurálny jazyk, pôvodne vyvinutý firmou IBM pre prácu s relačnými bázami dát, najmä v počítačových sieťach. Z dopytovacieho jazyka sa po viacerých rozšíreniach vytvoril univerzálny databázový jazyk, ktorý ako jediný jazyk tohto druhu je štandardizovaný. Dnes sa nachádza v každom komerčnom SRDB prekladač SQL v rôznych modifikáciách.



Obr. 6 Štruktúra relačného jazyka

Napríklad databázový systém Informix používa pre vývoj aplikácií jazyk ***4GL*** (4th Generation Language), do ktorého je plne integrovaný jazyk SQL.

Databáza v SQL je súbor tabuliek a pohľadov. Môže byť definovaná jednou alebo viacerými schémami.

Príkazy SQL môžeme rozdeliť do niekoľkých skupín:

- príkazy pre definíciu dát - ***DDL*** (Data Definition Language),
- príkazy pre manipuláciu s dátami - ***DML*** (Data Manipulation Language),
- príkazy pre zabezpečenie dát - ***DCL*** (Data Control Language),

Základ DDL tvoria príkazy:

- CREATE TABLE vytvorenie tabuľky,
- CREATE INDEX vytvorenie indexu,
- CREATE SPACE vytvorenie priestoru pre uloženie tabuliek,
- CREATE VIEW vytvorenie výrezu z tabuľky,
- ALTER SPACE modifikácia oblasti,
- ALTER TABLE modifikácia tabuľky.

Tabuľky sa vytvárajú príkazom CREATE TABLE, v rámci ktorého sa špecifikuje názov tabuľky, jej atribúty a ich typy, primárne kľúče a ohraničenia. SQL dovoľuje iba definíciu atomárnych typov.

Preddefinované typy sú:

- NUMERIC(n,m)
- SERIAL

- CHAR(n)
- SMALLINT
- DECIMAL(m,n)
- FLOAT
- DATE
- INTEGER
- MONEY(m,n)
- SMALLFLOAT
- INTERVAL
- DATETIME
- VARCHAR
- TEXT
- BYTE

Základ DML tvoria príkazy:

- DELETE výmaz riadku,
- DROP TABLE zrušenie tabuľky,
- INSERT vloženie riadku,
- SELECT výber dát z databázy,
- UPDATE pridanie položky do tabuľky.

Pre tvorbu dopytov v SQL slúži príkaz SELECT. Tento názov je zavádzajúci, pretože na základe príkazu SELECT sa môže vykonať aj kombinácia relačných operácií selekcie $\sigma_p(E)$ a projekcie $\Pi_s(E)$.

Príkaz má tvar:
SELECT [DISTINCT | ALL] {* | meno_atr1[,meno_atr2]...\} - čo sa má vyberať,
FROM meno_tab1[,meno_tab2 ...] - z ktorej tabuľky (tabuliek) sa má vyberať,
WHERE podmienka - podľa akých podmienok,
[ORDER BY meno_atr1[,meno_atr2] ...] - podľa čoho sa má vybraná skupina usporiadať,

Alebo

[GROUP BY] - ako zoskupiť vybranú skupinu,
[HAVING] - podobný ako WHERE pre riadky vybranej skupiny.

```
MAIN
   CREATE DATABASE demodb # Vytvorenie databázy demodb
   CREATE TABLE zamestnanec # Vytvorenie tabuľky zamestnanec
    (
       idz          SERIAL       NOT NULL,
       idf             INTEGER      NOT NULL,
       meno       CHAR(15),
       priezvisko CHAR(20)     NOT NULL,
       datum       DATE,
       plat          DECIMAL(5,0),
       vyska       DECIMAL(3,2),
       vaha        DECIMAL(4,1)
    )
   CREATE UNIQUE INDEX idz_ix ON zamestnanec (idz)
   CREATE       INDEX odkidf_ix    ON zamestnanec (idf)
   CREATE       INDEX mezam_ix  ON zamestnanec (meno)
   CREATE       INDEX priezam_ix ON zamestnanec (priezvisko)
   CREATE TABLE deti # Vytvoreni tabulky deti
    (
       idd           SERIAL    NOT NULL,
       idz           INTEGER NOT NULL,
       meno        CHAR(15),
priezvisko CHAR(20)  NOT NULL,
datum       DATE
     );
   CREATE UNIQUE INDEX idd_ix ON deti (idd)
   CREATE       INDEX odkidz_ix   ON deti (idz)
   CREATE       INDEX medeti_ix   ON deti (meno)
   CREATE       INDEX priedeti_ix  ON deti (priezvisko)
   CREATE TABLE stat # Vytvorenie tabulky stat
    (
skratka  CHAR(3)   NOT NULL,
nazov    CHAR(30) NOT NULL
     );
   CREATE UNIQUE INDEX skratka_ix ON stat (skratka)

CREATE TABLE firma # Vytvorenie tabulky firma
    (
       idf      SERIAL   NOT NULL,
       ico      CHAR(8),
       nazov    CHAR(30) NOT NULL,
       ulica    CHAR(30),
       psc      CHAR(8),
       mesto    CHAR(20),
       stat     CHAR(3),
       telefon  CHAR(17)
    );
   CREATE UNIQUE INDEX idf_ix     ON firma (idf)
   CREATE INDEX ico_ix     ON firma (ico)
   CREATE INDEX nazov_ix   ON firma (nazov)
   CREATE INDEX mesto_ix   ON firma (mesto)
   CREATE INDEX odkstat_ix ON firma (stat)
 END MAIN
```

Obr.7 Príklad vytvorenia databázy v SQL

Podmienkami výberu sú operátory pre rovnosť, nerovnosti, intervaly hodnôt, indikácia obsahu.

Operátory sa dajú spájať logickými operátormi negácie, súčtu a súčinu. Dopyty môžu byť zahniezdené pomocou zátvoriek.

Na potlačenie viacnásobného výpisu totožných dát slúži klauzula DISTINCT.

Príkazy pre zabezpečení dát (riadenia prístupu):

- GRANT – prideľovanie prístupových práv k danej databáze,
- REVOKE – odoberanie už pridelených prístupových práv.

```
SELECT  idf, ico, nazov, ulica, psc, mesto, stat, telefon FROM firma
END

SELECT firma.*, zamestnanec.*, deti.meno menoDietata, deti.idz,
                    deti.priezvisko priezviskoDietata
 FROM firma, zamestnanec, deti
 WHERE zamestnanec.idf = firma.idf AND deti.idz = zamestnanec.idz;
END

SELECT * FROM firma WHERE stat = $zvolenyStat ORDER BY nazov
END

SELECT firma.*, stat.nazov nazovStatu, skratka FROM firma, stat
WHERE firma.stat = stat.skratka ORDER BY stat
END

SELECT meno, priezvisko, datum, vyska, MONTH(datum) mesiac
FROM zamestnanec
 WHERE YEAR(datum) \> 1950  ORDER BY mesiac, vyska
END

SELECT zamestnanec.idf, MONTH(datum) mesiac, firma.idf idFirmy, nazov
 FROM zamestnanec, firma
 WHERE zamestnanec.idf = firma.idf  ORDER BY idFirmy, mesiac
 END
```

Obr. 8 Príklady použitia príkazu SELECT

Jazyk QBE

Pre väčšinu používateľov DBS, by bolo používanie jazyka SQL zbytočnou záťažou, preto sa hľadalo jednoduchšie, názornejšie riešenie.

Vzhľadom na to, že používanie grafického užívateľského rozhranie je dnes bežné vo väčšine aplikácií, vhodným nástrojom tvorby dopytov je tzv. jazyk QBE (Query By Example).

QBE využíva tabuľkovú formu relácie. Dopyty vytvára používateľ na obrazovke zápisom do záhlavia tabuliek. Vytvára tak vzory, šablóny dopytov (odtiaľ názov). QBE tvorí súčasť o.i. systémov Paradox a Access.

Nástroje ochrany dát v SRDB

Ochrana transakcií

Transakcie sú dôležitým konceptom v databázových systémoch. Transakcia je postupnosť operácií nad objektmi bázy dát, ktorá realizuje jednu ucelenú operáciu z hľadiska používateľa bázy dát. Transakcia sa začína začatím vykonávania svojho prvého príkazu alebo špeciálnym príkazom, napr. BEGIN TRANSACTION. Po úspešnom ukončení transakcie sa stávajú zmeny, ktoré transakcia uskutočnila v báze dát, platnými (bod potvrdenia) a databázový systém je pripravený na realizáciu ďalších transakcií. Posledný príkaz transakcie môže byť napr. END TRANSACTION. Neúspešne môže transakcia skončiť napr. kvôli poruche technického zariadenia, chybe programového vybavenia, prípadne ak sa zistili okolnosti, kvôli ktorým transakcia nemôže pokračovať. Úspešne ukončené transakcie nazývame potvrdenými transakciami, neúspešné sú tzv. zničené transakcie.

Realizáciou transakcie sa dostáva báza dát z konzistentného stavu znova do konzistentného stavu. V priebehu realizácie transakcie sa môže báza dát nachádzať aj v nekonzistentných stavoch.

Z hľadiska svojho vplyvu na bázu dát sú transakcie nedeliteľné (atomárne). Buď sa transakcia ukončí úspešne a všetky zmeny, ktoré spôsobila, budú platné, alebo sa skončila neúspešne a v báze dát nesmie zostať žiadna zmena ňou spôsobená.

Transakciou sa môžu meniť hodnoty niektorých objektov bázy dát. Počas vykonávania transakcie nesmú byť hodnoty, ktoré transakcia mení, prístupné pre iné súbežne bežiace transakcie.

SRDB zabezpečujú prevádzku databázového systému tak, aby opísaný transakčný model pracoval za všetkých okolnosti, a to i v prípade porúch. Dosiahne sa tak integrita bázy dát aj v podmienkach paralelnej práce viacerých transakcii. V prípade poruchy je potom potrebné zariadiť, aby sa obnovilo čo najviac transakcii potvrdených pred poruchou, v báze dát nezostali následky po žiadnej zničenej transakcii a po žiadnej transakcii, ktorá nebola do momentu poruchy potvrdená. Na zabezpečenie týchto funkcii sa používajú v SRDB rôzne techniky zneprístupnenia (zamykania) objektov a iné metódy ochrany integrity bázy dát.

Základným prostriedkom ochrany transakcií je zaznamenávanie zmien do tzv. žurnálového súboru. Tieto súbory uchovávajú staré a nové hodnoty tých objektov bázy dát, ktoré sa menia operáciami transakcie. Tieto záznamy umožňujú po poruchách:

- zopakovanie transakcie (redo), t. j. zmena stavu bázy dát z východiskového do cieľového stavu tak, ako sa táto zmena vykonala vlastnou transakciou,
- zrušenie transakcie (undo), keď sa odstraňujú z bázy dát dôsledky operácii úplne alebo čiastočne realizovanej transakcie.

Z hľadiska ochrany bázy dát je možné rozdeliť operácie nad objektmi bázy dát na tri skupiny:

- nechránené operácie, ktoré nemusia byť v prípade zničenia transakcie alebo rekonštrukcie hodnoty objektu zopakované alebo zrušené, napr. čítanie, diagnostické správy, zápisy na pracovne súbory,
- chránené operácie, ktoré musia byť v prípade zničenia transakcie alebo rekonštrukcie hodnoty objektu zopakované alebo zrušené,
- operácie na reálnych objektoch, ktorých následky sa nedajú po ich vykonaní technikami zotavenia odstrániť (expedícia výrobku, vyplatenie peňazí).

Jedným zo základných techník pre ochranu transakcií je dvojfázové potvrdzovanie, pri ktorom platí:

- transakcia nesmie meniť hodnoty objektov bázy dát skôr, ako je potvrdená,
- transakcia nemôže byť potvrdená skôr ako sa vytvoria príslušné záznamy v žurnálovom súbore.

Transakcia teda najprv zaznamená všetky zmeny do žurnálového súboru a až keď sa podarí vykonať posledný záznam do žurnálového súboru, je transakcia potvrdená. Potom sa prepíšu nové hodnoty objektov do bázy dát. Ak sa vyskytne porucha pred potvrdením transakcie, nebudú po nej v báze dát žiadne stopy. Ak sa vyskytne porucha po potvrdení transakcie, sú k dispozícii všetky záznamy žurnálového súboru, pomocou ktorých možno priradiť objektom, ktorých sa to týka, správne hodnoty.

Od začatia danej transakcie až po prepísanie posledného jej záznamu zo žurnálového súboru do bázy dát je potrebné zneprístupniť pre iné transakcie tie objekty, ktorých hodnoty sa budú meniť.

Pri dvojfázovom potvrdzovaní sa nové hodnoty priraďujú objektom bázy dát až po skončení transakcie. Toto môže byť nevýhodou pri dlhotrvajúcich transakciách a v prostredí, v ktorom sa predpokladá pri spracovaní viacerých transakcií paralelná, súbežná práca. Z tohto hľadiska je lepší taký postup, pri ktorom sa zmeny objektov vykonávajú priamo v báze dát. Predpokladá to vedenie žurnálového súboru so záznamami o hodnotách objektov po príslušnej operácii a pred príslušnou operáciou. Po priradení novej hodnoty objektu bázy dát ho možno, ak to nenaruší integritu bázy dát, uvoľniť pre iné, paralelne bežiace transakcie.

V prípade zničenia transakcie je možné zreštaurovať konzistentný stav aplikovaním tých záznamov zo žurnálového súboru, ktoré umožňujú zrušenie transakcie. Po poruche, ktorá nastane po potvrdení transakcie, t. j. po poslednom zázname do bázy dát a do žurnálového súboru, je možné využiť záznamy v žurnálovom súbore na zopakovanie transakcie. Pri tomto postupe sme ale pripustili, aby transakcie bežiace paralelne s danou transakciou mohli čítať objekty, ktoré daná transakcia mení. Môže to spôsobiť značné ťažkosti pri rekonštruovaní konzistentného stavu bázy dát. Je zrejmé, že v žurnálovom súbore sa musia zaznamenávať aj informácie o prečítaní hodnoty objektov, ktoré sa menili inou, doteraz nepotvrdenou transakciou, ako aj záznamy o začatí a ukončení každej transakcie.

Pri vykonávaní zrušenia alebo zopakovania môže dôjsť k poruchám, a preto je potrebné mať možnosť spustiť tieto operácie znovu. Systém ochrany transakcií musí preto zaistiť, aby sa pri obnove konzistentného stavu nemenili hodnoty objektov, ktoré majú správnu hodnotu.

Žurnálový súbor môže obsahovať i ďalšie informácie, napr. údaje o čase na umožnenie rekonštrukcie k presnému termínu, údaje o iniciátorovi transakcie (programe, programátorovi).

Ochrana integrity bázy dát je v reálnych aplikáciách vec prvoradého významu. Preto sa súčasné SRDB vybavujú v tomto smere bohatými funkčnými možnosťami, ktorých programové zabezpečenie tvorí často značnú časť SRDB.

Trigger

Trigger je pomenovaný databázový objekt, ktorý je implicitne aktivovaný, ak nastane udalosť podmieňujúca jeho spustenie a po aktivovaní vyvolá dopredu definovanú akciu. Pod udalosťou sa tu rozumie špecifická situácia v databázovom systéme, napr. doplnenie (INSERT) riadku do tabuľky. Akcia je procedúra alebo postupnosť operácií uložená ako súčasť SRDB na databázovom serveri.

Triggery umožňujú nielen sledovať referenčnú integritu databázy, ale ich pomocou sa dajú vykonať pomerne rozsiahle opatrenia pre uplatnenie integritných obmedzení. Zložitejšie integritné pravidlá môžu byť samozrejme aj súčasťou aplikačného programu, ale použitie triggerov ich posunie k definícii databázy.

Pomocou triggerov sa ďalej dá zabezpečiť:

- ochrana dát - trigger môže kontrolovať, či používateľ má právo vykonať danú operáciu.
- generovanie auditných záznamov, tj. súborov zaznamenávajúcich vykonanie určitých operácií na chránených údajoch,

- aktualizácia pohľadov,
- replikácia tabuliek.

Výhodami použitia triggerov sú:

- rýchlejší vývoj aplikácie. Akcie vykonávané triggermi sú vytvorené a uložené spolu s SRDB a sú dostupné pre rôzne aplikácie.
- globálne uplatňovanie aplikačných pravidiel (raz definovaný trigger je použiteľný pre všetky aplikácie, ktorá menia tabuľku),
- jednoduchšia údržba aplikačných pravidiel (stačí zmeniť trigger a nie niekoľko aplikačných programov),
- centrálna aktivácia triggerov vyhovuje architektúr klient/server.

Podľa štandardu SQL3 udalosťami spúšťajúcimi trigger sú aktualizačné operácie INSERT, DELETE a UPDATE (pre vybrané stĺpce alebo na celú tabuľku). Niektoré SRDB túto možnosť spájajú aj s príkazom SELECT. Podmienka spustenia sa vyjadruje štandardnými logickými podmienkami SQL. Trigger môže byť aktivovaný pred (BEFORE) alebo po (AFTER) aktualizačnej operácii. Akcia sa môže vykonať pre každý riadok (FOR EACH ROW), ktorý spĺňuje podmienku, alebo len raz pre celý príkaz triggera (FOR EACH STATEMENT).

```
int create account owner (String name, String address)
{
Database *bank db;
bank db = Database::open("Bank-DB");
Transaction Trans;
Trans.begin();
Ref hAccount iaccount = new(bank db) Account;
Ref hCustomer icust new(bank db) Customer;
cust->name = name;
cust->address = address;
cust->accounts.insert element(account);
account->owner.insert element(cust);
: : : Inicializácia customer id, account number, atď.
Trans.commit();
}
```

Obr. 9 Príklad programu v jazyku ODMG C++ OML

Objektovo orientované databázy

Objektová orientácia DBS sa môže prejaviť dvoma spôsobmi:

- objektový prístup sa uplatní ako nástroj návrhu, ale výsledkom je relačná databáza;
- používa objektovo orientovaný jazyk pre manipuláciu s údajmi v rámci SRDB.

V druhom prípade je možné objektovú koncepciu integrovať do existujúcich jazykov. Napríklad SQL sa dá rozšíriť o komplexné dátové typy a objekty. Systémy, ktoré vznikajú doplnením svojich prostriedkov o objektovo orientované prvky sa nazývajú objektovo-relačné systémy. Taktiež môžeme vychádzať z existujúceho objektovo orientovaného programovacieho jazyka a rozšíriť ho o nástroje pre prácu s databázou. Tieto jazyky sa niekedy nazývajú perzistentné programovacie jazyky. Súvisí to so spô-

sobom spracovávania dát: perzistentné (pretrvávajúce) dáta sú dáta, ktoré sa uchovajú aj po ukončení programu, ktorý ich vytvoril alebo modifikoval. Perzistentný programovací jazyk je programovací jazyk doplnený o možnosť práce s perzistentnými údajmi.

ODMG (Object Database Management Group) pracuje v posledných rokoch na štandardizácii rozšírení jazyka C++ o nástroje na manipuláciu s databázovými objektmi a na definovaní knižnice príslušných tried. Rozšírenia ODMG C++ sa delia do dvoch skupín: jazyk na definovanie objektov (Object Definition Language - C++ ODL) a jazyk na manipuláciu s objektmi (C++ Objekt Manipulation Language - C++ OML).

Databázy priestorových objektov

V konvenčných DBS sa uchovávajú a spracúvajú dáta predstavujúce najmä textové a číselné položky. Súčasné databázové technológie umožňujú aj spracovanie údajov reprezentujúcich priestorové objekty. V dvojrozmernom priestore sa jedná o geografické aplikácie typu GIS, v troj-rozmernom priestore o objekty používané v CAD a CIM. Spoločnou charakteristikou týchto aplikačných oblastí je nutnosť podpory veľkého množstva relatívne jednoduchých geometrických objektov. Tieto objekty sú lokalizované v priestore, majú tvar a identitu.

Dvojrozmerným priestorovým objektom môže byť mapa, ktorú si zjednodušene môžeme predstaviť ako mnohouholník. K objektom typu mapa môžu existovať indexové štruktúry umožňujúce nájsť k danému objektu susedné objekty, napr. priestorovým spojením rieky a miest môžeme určiť mestá na brehu rieky.

Tradičné databázové techniky nie sú pre reprezentáciu takých objektov vhodné. Predstavme si napr. budovu, ako teleso zložené zo štvorstenov. Štvorsteny sú dané pomocou rovín, hrán a bodov. Ak použijeme relačnú databázu, môžeme odpovedajúce dáta uložiť v štyroch reláciách: ŠTVORSTENY, ROVINY, HRANY a BODY. Relácia ŠTVORSTENY obsahuje dvojice (*t*, *f*), kde *t* je identifikátor štvorstenu, *f* je identifikátor jednej z jeho rovín. Relácia ROVINY obsahuje dvojice (*f*, *e*), kde *f* je identifikátor roviny a *e* je jedna z hrán v rovine. Relácia HRANY obsahuje trojice (*e*, *p*, *q*), kde *e* je identifikátor hrany, *p* a *q* sú koncové body hrany. Relácia BODY obsahuje štvorice (*p*, *x*, *y*, *z*), kde *p* je identifikátor bodu a *x*, *y*, *z* sú jeho súradnice. Je zrejmé, že takáto reprezentácia nezohľadňuje geometrické vzťahy potrebné pre priestorové dopyty. Nájsť napr. Odpoveď na dopyt, či daná priamka pretína daný štvorsten vyžaduje viac vstupov do rôznych relácií. Pri takejto manipulácii s údajmi nevyužívame geometrické vlastnosti priestoru.

Postup pri vytváraní bázy dát

Jednotlivé etapy budovania databázy môžeme zhrnúť takto:

- analýza požiadaviek používateľov,
- tvorba konceptuálneho modelu databázy,
- logický model databázy,
- fyzický model databázy,
- implementácia bázy dát.
- prvotné naplnenie bázy dát údajmi.
- overenie funkčnosti databázy.
- prevádzkovanie a ďalší rozvoj databázy.

Základ konceptuálneho modelu databázy spočíva v dôslednom vyhodnotení informačného toku, ktorý prebieha v organizácii, ktorý vzniká na základe sledovania

vzájomných súvislostí medzi jednotlivými oddeleniami, spôsobu využívania uložených údajov a vyhodnotenia požiadaviek na vstupné a výstupné údaje. Ak analytik má všetky informácie, vyhotoví prvý (konceptuálny) model databázy a postupne ho konzultuje so všetkými pracovníkmi organizácie, ktorí budú pracovať s jej údajmi. Pretože užívatelia databázy nemusia byť odborníci v oblasti tvorby databáz, konceptuálny model nepoužíva odborné výrazy. Jeho význam spočíva v tom, že jednoduchým spôsobom mapuje tok informácií v organizácii a znázorňuje ho spôsobom, ktorý je pochopiteľný pre všetkých užívateľov vytváranej databázy s cieľom, čo najviac prispôsobiť databázu jej reálnemu využitiu. Konečným krokom pri tvorbe konceptuálneho modelu databázy je dôsledný návrh skupín navzájom súvisiacich údajov (napr. údaje o pracovníkovi, údaje o oddelení a pod.) a náčrt vzájomných prepojení medzi týmito skupinami. Konceptuálny model obsahu je aj návrh východzích údajov, ktoré má organizácia k dispozícii a ktoré sú nevyhnutné pre tvorbu výstupných zostáv v jednotlivých oddeleniach.

Tvorbu logického modelu databázy môžeme charakterizovať pomocou týchto základných krokov:

- špecifikácia potrebných údajových položiek, ktoré budú včlenené do databázy,
- špecifikácia úrovne detailov jednotlivých údajových položiek, v závislosti na manipulácii s nimi v rámci databázy,
- určenie logických celkov údajových položiek,
- určenie vzájomných prepojení týchto celkov,
- normalizáciou navrhovaných relačných vzťahov medzi logickými skupinami údajov.

Logický model databázy už obsahuje návrh tabuliek, v ktorých budú údaje usporiadávané, určenie typov údajov ukladaných v databáze a grafické znázornenie ich relačných prepojení. Fyzický model databázy tvorí návrh programu, vytvorený na základe predchádzajúcich dvoch návrhov. Tento model databázy tvoria odborníci v oblasti programovania databázových systémov a je priamo závislý na programovom jazyku, v ktorom sa databázový systém vytvára.

Samotné modely reality neberú do úvahy prostredie v akom bude IS prevádzkovaný, riešia len obsahovú stránku problému. Implementácia je súhrn činností, ktoré zabezpečia konkrétne technické a programové prostriedky pre realizáciu abstraktného modelu, pokiaľ možno s najmenším počtom dodatočných obmedzení.

Postup pri implementácii môže byť nasledujúci:

- Výber implementačného prostredia.
- Vytvorenie logickej štruktúry bázy dát.
- Vytvorenie fyzickej štruktúry bázy dát.

Pre úspešnú implementáciu si treba ujasniť:

- spôsob aktualizácie údajov (on-line, dávkovo),
- požiadavky na prístupové stratégie k údajom (podľa kľúča, sekvenčne),
- predpokladanú frekvenciu prístupu,
- požiadavky na ochranu dát,
- požiadavky na dobu odozvy,
- predpokladaný počet záznamov a kapacitu dostupnej pamäti,
- aké percento záznamov sa zmení pri jednej aktualizácii,
- obmedzenia vyplývajúce z parametrov použitého počítačového systému,
- možnosť implementácie po etapách,
- kompetencie v prípade používania distribuovanej databázy.

Multidatabázové systémy

Najstarším prístupom k správe a integrácii viacerých heterogénnych informačných zdrojov je vytváranie tzv. multidatabázových systémov. Multidatabázový systém (***MDBS***) je databázový systém, ktorý zastrešuje existujúce autonómne heterogénne databázy (lokálne databázy) pri zachovaní ich lokálnej autonómie. Na rozdiel od distribuovaných DBS sa nejedná o zjednotenie do jedného celku, napriek tomu používatelia majú dojem, že pracujú s jedinou databázou. Lokálna autonómia znamená, že neexistuje žiadna možnosť modifikácie lokálnych dátových schém, rozmiestnenia dát a SRDB. Pre MDBS sa niekedy používa aj termín „federated databases", čo je možné preložiť ako „združené databázy". Tento termín sa používa ako ekvivalent MDBS, alebo pre MDBS s nižším stupňom integrácie dátových modelov ale vyšším stupňom spolupráce medzi databázami.

MDBS uchováva tzv. globálnu schému, ktorá vzniká integráciou schém lokálnych databáz. V skutočnosti sa integrujú iba časti lokálnych schém, tzv. exportné schémy, ktoré určí administrátor lokálnej databázy. Exportná schéma je následne preložená do spoločnej schémy MDBS.

Spoločné schémy jednotlivých databáz sú po vyriešení konfliktov spojené do integrovanej schémy. Na základe integrovanej schémy administrátor alebo užívateľ zostaví externú schému, ktorá je sadou pohľadov na integrovanú schému. Globálna schéma pozostáva z integrovanej schémy a externej schémy. Dopyty užívateľov smerujú na globálnu schému MDBS, ale všetky dáta sú spravované lokálnymi databázovými systémami.

Komunikácia medzi MDBS a lokálnymi systémami prebieha pomocou brán (gateways). Problémom tohto prístupu je potreba veľkého množstva znalostí o lokálnych schémach a o tom, ako identifikovať a riešiť heterogenity medzi nimi.

Zmeny v lokálnych schémach musia byť premietané do globálnej schémy. Zložitosť globálnej schémy spôsobuje, že aj malá zmena lokálnej schémy môže vyžadovať rozsiahle zmeny v globálnej schéme, čo zapríčiňuje problémy pri jej udžiavaní.

Niektoré riešenia multidatabázových systémov sa snažia vyhnúť týmto problémom tým, že vôbec nepoužívajú globálnu schému. Tieto prístupy poskytujú užívateľovi funkcie rozširujúce štandard SQL, aby mohli vo svojich dopytoch špecifikovať niektoré podmienky súvisiace s vyhodnocovaním dopytov.

Distribuované databázové systémy

V počiatkoch hromadného spracovania dát sa vytvárali centralizované DBS, v ktorých tak súbory dát, ako aj programy na ich spracovanie boli sústredené na jedno miesto (do jediného počítača). Vývoj komunikačných a počítačových sietí umožnil prístup veľkého počtu, aj vzdialených používateľov k bázam dát. S ohľadom na požiadavku vzdialeného prístupu a aktualizácie dochádzalo k distribúcii súborov na jednotlivé počítače a vytváraniu distribuovaných databázových systémov (DDBS).

Pod distribúciou rozumieme rozdelenie výpočtového výkonu a dát do viacerých uzlov počítačovej siete. Distribuovanú databázu môžeme definovať ako jedinú logicky jednotnú databázu, ktorá fyzicky je rozmiestnená na viacerých počítačoch, prepojených dátovou komunikačnou sieťou.

Typické DDBS sú už od začiatku budované s úmyslom teritoriálnej distribúcie súborov pri zachovaní možnosti integrovaného spracovania dát, bez ohľadu na fyzické umiestnenie. Dôvody pre takéto riešenie môžu byť buď spoľahlivostné (dôležité súbory sa v oddelených počítačoch vyskytujú v niekoľkých kópiách) alebo ekonomické

a organizačné (keďže súbory používajú viacerí účastníci, sú umiestnené tak, aby náklady na prístup k údajom boli čo najmenšie).

Výhody distribuovaného spracovania:

- Lokálna transparentnosť, čiže používateľ systému vidí údaje tak, ako keby boli na jeho počítači. Všetky medzistupne sú preňho neviditeľné („priesvitné - transparentné") a nemusí riešiť ani súvisiace problémy.
- Zvýšená spoľahlivosť je daná rozdelením dátových zdrojov, čiže pri poruche niektorých častí DDBS, neprestane fungovať celý systém, spravidla dôjde len k obmedzeniu určitých funkcií.
- Zodpovednejšie spravovanie dát vyplýva z dislokácie častí dát tam, kde vznikajú a kde sú útvary kompetentné ich udržiavať.
- Pružnejšia rozšíriteľnosť konfigurácie je daná možnosťou modulárnej výstavby DDBS. Lokálne systémy sa dajú obnovovať a doplňovať bez odstavenia celku.
- Znížené náklady na komunikáciu a rýchlejšiu odozvu systému môže priniesť premyslené rozdelenie dát, ktorý zredukuje objem dát prenášaných medzi jednotlivými účastníkmi.
- Dá sa docieliť vyšší stupeň paralelizmu spracovania a tým celkove vyšší výkon DDBS.

Nevýhody distribuovaného spracovania:

- Vyššie náklady na programové vybavenie sú dané nutnosťou adekvátneho programového vybavenia na lokálnej a globálnej úrovni.
- Vyššie prevádzkové náklady.
- Ťažšia kontrola a ochrana integrity dát v dôsledku viacúrovňovosti DDBS.

Efektívna činnosť DDBS je podmienená vhodným rozdelením dát. V prípade relačných DBS sa bude jednať o vhodnú distribúciu dát z globálnych tabuliek.

V praxi sa používajú nasledujúce spôsoby distribúcie:

- *Replikácia dát*, pri ktorej vznikajú lokálne bázy dát totožné s centrálnou databázou, čo predpokladá odpovedajúcu pamäťovú kapacitu. Význam replikácie spočíva vo väčšej dostupnosti a bezpečnosti dát. Všetky transakcie sa robia na lokálnych počítačoch. Po prípadnom zničení dát na jednom mieste, sa dajú tieto nahradiť z iných počítačov DDBS. Problémom je udržanie konzistentnosti dát v rôznych lokalitách. Pri aktualizácii sa musia nové hodnoty zapracovať do všetkých lokálnych tabuliek. Replikácia dát sa používa u systémov, v ktorých je dôležité zabezpečenie dát, pričom počet a objem zmien je nízky.
- *Fragmentácia dát* umožňuje používateľom rozdeliť dátové tabuľky horizontálne alebo vertikálne a umiestniť takto vzniklé fragmenty do rôznych uzlov siete. Horizontálna fragmentácia sa označuje ako objektové členenie. Štruktúra tabuliek je rovnaká na všetkých uzlových počítačoch, rozdielne sú objekty (entity), ktorých sa záznamy týkajú. Vertikálnu fragmentáciu označujeme ako funkčné členenie, pri ktorom sú v jednotlivých lokalitách uložené tabuľky s odlišnou štruktúrou. V stĺpcoch týchto tabuliek budú rôzne atribúty entít. Pochopiteľne, kľúčové atribúty sa musia zachovať pre zabezpečenie referenčnej integrity DDBS.

Podľa konkrétnych podmienok sa používa aj kombinácia oboch uvedených postupov.

Dátové sklady

Klasická definícia dátových skladov znie: subjektovo orientovaná, integrovaná, časovo rozlíšená a stála zbierka dát určená na podporu procesu manažérskeho rozhodovania (W. Inmon 1988).

Dátový sklad (Data Warehouse - ***DW***) je akýmsi archívom alebo skladom informácií, ktoré sa získavajú z odlišných a fyzicky oddelených operatívnych systémov organizácie, ako aj z externých dátových zdrojov. Používajú ho riadiaci pracovníci a špecialisti, ako zdroj údajov v rozhodovacom procese.

Údaje v dátovom sklade sú usporiadané prioritne podľa predmetu a až potom podľa aplikácie. Rozdielne dátové a prezentačné formáty sa zosúladia na vopred dohodnutý štandard. Údaje sa priebežne ukladajú a používajú sa na porovnávanie, na určenie trendov a na prognózovanie. Tieto údaje nie sú aktualizované v reálnom čase, ale sú dopĺňané a obnovované z operatívnych systémov v pravidelných intervaloch (napr. denne, týždenne, mesačne), keď prenos údajov neovplyvní výkon informačných systémov.

Dátový sklad je v podstate inteligentný systém ukladania údajov do pamäti, v ktorom môžu pracovať extrémne široké aplikácie na poskytovanie informácií, akými sú vybilancované systémy indikátorov alebo systém riadenia vzťahov k respondentom.

Dátové sklady sú výhodné pre veľké organizácie, ktoré chcú čo najviac vyťažiť z implementácie dátového skladu a pritom používajú viaceré rozdielne operačné systémy a preto majú problémy s prístupom ku koherentným a konzistentným firemným informáciám.

Koncovými používateľmi dátových skladov sú najmä riadiaci pracovníci a špecialisti, ktorí prijímajú rozhodnutia na základe existujúcich informácií.

Typickými problémami, ktoré dátové sklady pomáhajú riešiť sú nasledujúce:

- Získať informácie o celkovej situácii v organizácii ako takej alebo rýchlo získať určité špecifické informácie alebo ich vôbec získať. Dátový sklad rieši tento problém tým, že poskytuje databázu informácií, ktoré organizácia alebo jej oddelenia nutne potrebujú pre svoj pracovný proces.
- Organizácie majú také množstvo údajov v rôznych systémoch a lokalitách, že je obtiažne získať potrebné výstupy. Dátový sklad obsahuje vyfiltrované a sformátované údaje, ktoré umožňujú používateľom rýchly prístup k analýzam a zostavám (report), čo je pre nich dôležitejšie ako strácať čas pri prístupe k samotným údajom alebo čakaním na pracovníkov informačných technológií, aby im vytvorili potrebné zostavy.
- Tvorba zostáv je príliš pomalá a často aj oneskorená. Pri každej potrebe novej zostavy alebo pri jej zmene musia spolupracovať špecialisti na IT. Dátový sklad odstraňuje závislosť pracovníkov organizácie od pracovníkov IT, ktorí pre nich vytvárajú analýzy a zostavy.
- Informácie sú často nekonzistentné a nepresné, a preto neposkytujú aktuálny, celkový pohľad. Údaje v dátovom sklade sú konzistentné, pretože sú štandardizované v procese ich získavania z operatívnych systémov. Používatelia priebežne rozhodujú, ktoré údaje sa ponechajú v dátovom sklade a ako sú prezentované.

Zmyslom vybudovania dátového skladu je vytvoriť v organizácii jednotnú, homogénnu a konzistentnú dátovú základňu, nad ktorou sa potom dajú efektívne používať nástroje a systémy na podporu manažérskeho rozhodovania a nástroje na *dolovanie dát (data mining)*.

Pojem dolovanie dát možno v najširšom ponímaní definovať ako vyhľadávanie vzťahov a vzorov v súbore dát. Táto definícia zahŕňa aj sféru verifikačných nástrojov, ktorými sú dopytovacie

- nástroje,
- multidimenzionálna analýza,
- vizualizácia.

Verifikačný charakter u týchto nástrojov spočíva v iteračnom procese:

- definícia problému,
- stanovenie hypotézy používateľom,
- vyhodnotenie hypotézy pomocou verifikačného nástroja,
- podľa prijatia, či zamietnutia hypotézy návrat na krok 2.

V priebehu uvedeného procesu je používateľ v úlohe „navádzača", kým sa nedopracuje k uspokojivému výsledku. „Pravé" dolovanie dát používa model „objavovania", v ktorom systém automaticky nájde príslušné vnútorné vzťahy a vzory v dátach a na tieto fakty upozorní používateľa.

Metódy dolovania dát využívajú päť základných typov funkcií:

- *Charakterizovanie tried (klasifikácia).*
  Je daná množina záznamov s príslušnými atribútmi, spolu s množinou príznakov určujúcich príslušnosť každého záznamu do klasifikačnej triedy. Klasifikačná funkcia zistí (pomocou príznakov príslušnosti a na základe atribútov) explicitný alebo implicitný popis vlastností príslušných tried. Pri explicitnom popise ide o sadu pravidiel popisujúcich danú triedu, pri implicitnom popise je navrhnutý funkčný predpis priraďujúci ľubovoľný vstupný záznam príslušnej triede.
- *Zhlukovanie.*
  Je daná množina záznamov s príslušnými atribútmi a úlohou je rozdeliť tieto záznamy (na základe atribútov a určitého kritéria) do zmysluplných tried/segmentov. Voľba zhlukovacieho kritéria závisí od príslušného nástroja.
- *Vyhľadávanie asociácií.*
  Je daný súbor položiek a množina záznamov, z ktorých každý obsahuje položky z daného súboru a úlohou asociačnej funkcie je vrátiť príbuznosti, ktoré existujú medzi položkami daného súboru na základe analýzy záznamov. Napríklad pri vyhľadávaní asociácií v nákupnom koši môže byť výsledkom výrok „60% všetkých záznamov, ktoré obsahujú položky A a B, obsahuje aj položku C".
- *Objavovanie sekvenčných vzorov.*
  Na rozdiel od predchádzajúceho je známa identita každého záznamu, ako napr. meno zákazníka. Tým pádom je možné vyhľadať opakujúce sa vzory položiek pre danú skupinu zákazníkov a stanoviť prípadne pravidlá definujúce príčinno-následné vzťahy medzi skupinami záznamov.
- *Predikcie.*
  Kombináciou predchádzajúcich funkcií možno predpovedať smer ďalšieho vývoja.