# 2. Databáze

## 2.1 Relační databáze

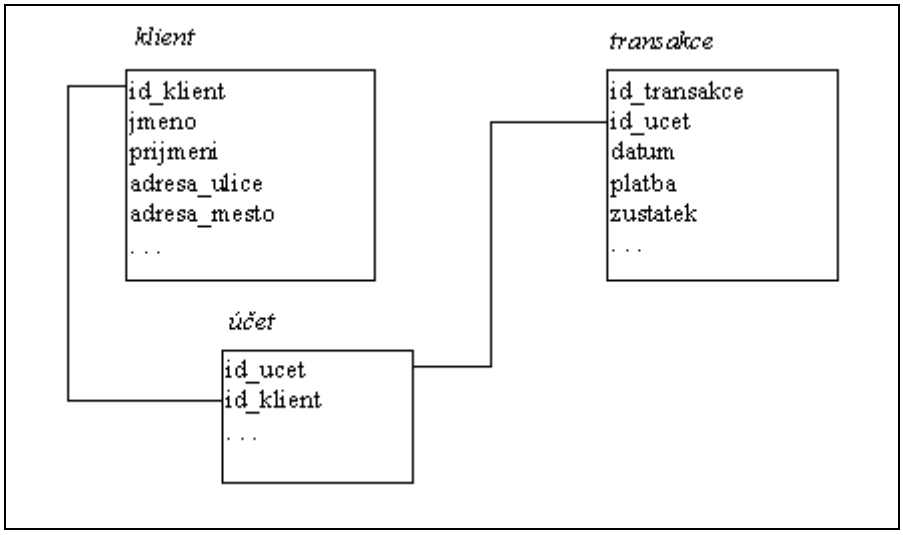
V prehistorii databází byla data ukládána v jednom velkém „plochém“ souboru (tzv. flat file) ke kterému se přistupovalo indexsekvenčními metodami (ISAM). Soubor byl indexován na základě předpokládaných způsobů dotazování. Nevýhodou bylo, že se informace v záznamech opakovaly (viz Obr. 1), další nevýhodou bylo předurčení typu dotazů (daných dopředu zvoleným způsobem indexování).

| datum | jmeno | prijmeni | adresa_ulice | adresa_mesto | cislo_uctu | platba | zustatek |
|---|---|---|---|---|---|---|---|
| 980103 | Jan | Novak | Dlouha 5 | Praha 1 | 9945371 | 100.00 | 100.00 |
| 980105 | Jan | Novak | Dlouha 5 | Praha 1 | 9945371 | 1500.00 | 1600.00 |
| 980106 | Jan | Novak | Dlouha 5 | Praha 1 | 9945371 | -1550.00 | 50.00 |
| 980106 | Karel | Nemec | Podolska 4 | Praha 2 | 24867134 | 3000.00 | 6000.00 |
| 980107 | Karel | Nemec | Podolska 4 | Praha 2 | 24867134 | -4000.00 | 2000.00 |
| 980108 | Jan | Novak | Dlouha 5 | Praha 1 | 9945371 | -150.00 | -100.00 |
| 980111 | Karel | Nemec | Podolska 4 | Praha 2 | 24867134 | 5000.00 | 7000.00 |
| | | | . . . | | | | |

*Obr. 1 Plochý soubor s daty*

Velkým krokem kupředu bylo zavedení relačních databází. Jeden velký datový soubor byl rozdělen do řady relací (tabulek). Relační databáze je tedy tvořena:

- množinou relací - relace je reprezentována dvourozměrnou tabulkou (řádky odpovídají záznamům, sloupce atributům, jednotlivé záznamy jsou jednoznačně identifikovány pomocí primárního klíče),
- operacemi selekce, projekce a spojení pro manipulaci s tabulkami; selekce slouží k výběru záznamů (řádků tabulky), projekce slouží k výběru atributů (sloupců tabulky) a spojeni slouží k propojování tabulek (spojují se řádky se stejnou hodnotou nějakého atributu - obvykle klíče).




*Obr. 2 Relační databáze*

Pro kladení dotazů nabízejí relační databáze dva způsoby:

- QBE (query by example)
- SQL (structured query language)

Oba tyto způsoby navrhla v 70. letech firma IBM. QBE nabízí uživateli relativně jednoduchý, intuitivní způsob kladení dotazů. V předpřipraveném formuláři uživatel vyplní (vybere) co ho zajímá, zadá tedy jakousi „masku“ které budou odpovídat nalezené záznamy. Je tedy tento způsob vhodnější pro méně zkušené uživatele. SQL je naopak určen uživatelům zkušeným. Jedná se vlastně o jednoduchý programovací jazyk pro definování dat a pro manipulaci s nimi (Obr. 3). SQL je daleko mocnější a flexibilnější nástroj než dotazování (vyhledávání) pomocí indexů. Klade však zvýšené nároky na uživatele. Uživatel musí znát syntaxi jazyka, navíc musí znát i detailní strukturu databáze (názvy souborů a polí). Tento způsob dotazování tedy příliš nepřirostl k srdcím manažerům a analytikům. Pokud se nechtěli učit SQL, byli nuceni připravit dotaz v přirozeném jazyce, zajít za programátorem, který jejich dotaz převedl do jazyka SQL a provedl jej. V případě, že obdržené výsledky nepřinesly požadovanou odpověď, museli dotaz přeformuloval a znovu se vydal do výpočetního centra...atd.

```
SELECT    klient.jmeno, klient.prijmeni, klient.adresa_ulice, klient.
          adresa_mesto, ucet.cislo_uctu, transkace.zustatek
FROM      klient, ucet, transakce
WHERE  klient.id_klient  = ucet.id_klient;
    AND  transakce.id_ucet = ucet.id_ucet;
    AND  transakce.zustatek < 100;
GROUP BY      klient.adresa_mesto
```

*Obr. 3 SQL dotaz*

## 2.2 EIS

EIS (Executive Information Systems) byl první pokus, jak přiblížit dotazování do databáze manažerům. Zavádění EIS bylo spojeno se zaváděním osobních počítačů v dané firmě; počítače přestaly být doménou programátorů, objevily se na stolech „prostých“ uživatelů. Základním požadavkem se tedy stalo snadné ovládání. Uživatelsky přátelský interface exekutivních informačních systémů „prosté“ uživatele odstínil od syntaxe SQL a od nutnosti znát strukturu databáze, se kterou chtěli pracovat. Analýzu tedy mohl analytik provádět sám, z počítače, který měl na svém pracovním stole. Dotaz, který si uživatel vybral v menu, byl pak převeden do SQL a proveden standardním způsobem. Nevýhodou tohoto přístupu bylo, že uživatel měl k dispozici pouze určitý soubor předpřipravených dotazů. Chtěl-li se zeptat na něco, co tvůrce daného EIS nepředpokládal, byl opět nucen připravit si dotaz v přirozeném jazyce, zajít za programátorem, který dotaz převedl do jazyka SQL...

EIS tedy byly sice uživatelsky přátelské, ale málo flexibilní nástroje pro analýzu dat v databázích.

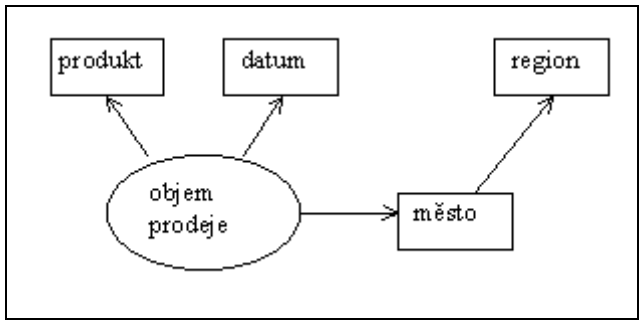
## 2.3 OLAP

OLAP (On-Line Analytical Processing) konečně nabídl uživatelům obojí; flexibilitu (a rychlost) i příjemné, intuitivní ovládání. OLAP umožňuje analytikům firmy snadno získávat odpovědi na otázky typu „Co má největší vliv na růst tržeb v severních Čechách?“ ve velice srozumitelné podobě. Typické pro OLAP jsou totiž možnosti vizualizace. Grafické rozhraní umožňuje uživateli nahlížet na data jak v numerické podobě tak v podobě nejrůznějších grafů.

Podle E.F.Codda, který v 80.letech přišel s touto koncepcí, je pro OLAP technologii typické [Thomsen, 2002]:

- multidimenzionální koncept uložení a manipulace s daty,
- intuitivní manipulace s daty,
- práce s daty z heterogenních datových zdrojů - provádějí se konverze dat,
- použití analytických metod - statistické přehledy, what-if analýzy,
- Client/Server architektura,
- podpora multiuživatelského pohledu,
- ukládání výsledků OLAP mimo zdrojová data,
- dynamická manipulace s řídkými maticemi,
- zpracování chybějících hodnot,
- neomezený počet dimenzí a agregačních úrovní.

Základem OLAP je pohled na data jako na mnohorozměrnou tabulku nazývanou *datová krychle* (data cube). Obr. 4 ukazuje strukturu jednoduché databáze. Záznamy v databázi obsahují údaje o prodeji různých produktů v různých dnech v různých městech (Tab. 1). Tuto databázi můžeme převést na datovou krychli tak, že jednotlivé sledované atributy budou tvořit dimenze krychle, buňky krychle pak odpovídají jednotlivým záznamům v databázi. Tento způsob uložení umožňuje různé pohledy na data (natáčení krychle, provádění řezů) ale plýtvá se při něm místem. Řada buněk v krychli je prázdných (Tab. 2) [1].




*Obr. 4 Struktura databáze*

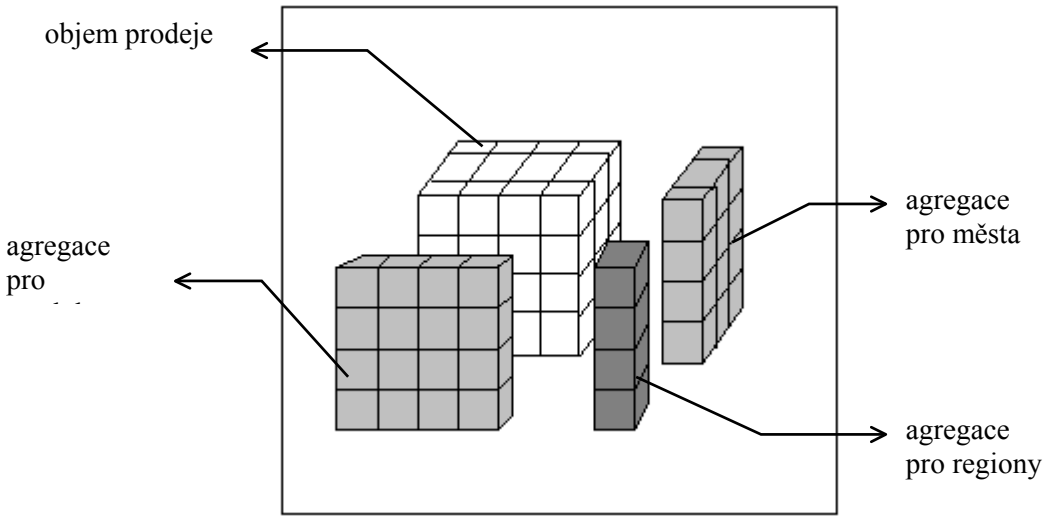
[1] Matematika má pro takovou strukturu označení řídká matice.

| datum | produkt | město | množství |
|---|---|---|---|
| 10.1. | šrouby | Praha | 241 |
| 10.1. | matky | Praha | 61 |
| 10.1. | šrouby | Brno | 17 |
| 10.1. | podložky | Brno | 42 |
| 10.2. | šrouby | Praha | 92 |
| 10.2. | podložky | Praha | 27 |
| 10.2. | šrouby | Kladno | 35 |

*Tab. 1 Záznamy v databázi PRODEJ*

| | Praha | | | Brno | | | Kladno | | |
|---|---|---|---|---|---|---|---|---|---|
| | šrouby | matky | podložky | šrouby | matky | podložky | šrouby | matky | podložky |
| 10.1. | 241 | 61 | | 17 | | 42 | | | |
| 10.2. | 92 | | 27 | | | | 35 | | |

*Tab. 2 Řídká matice*




*Obr. 5 Multidimenzionální model*

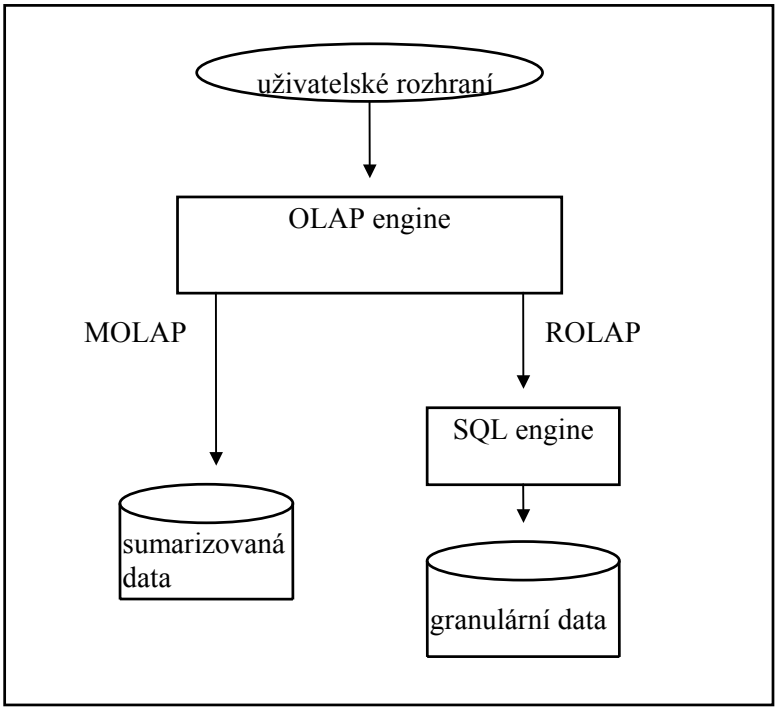
Datová krychle obsahuje jak data z operačních databází tak dílčí souhrny (viz Obr. 5). Právě tyto souhrny umožňují rychlou odezvu na ad-hoc (nepředpřipravené) dotazy uživatele a flexibilitu systému. Práce s krychlí spočívá v různém natáčení (pivot), provádění řezů (slice), výběru určitých částí (dice) a zobrazování různých agregovaných hodnot. Velmi často lze hodnoty atributů sdružovat

do hierarchií (v databázi na Obr. 4 jsou např. města zařazena do regionů). Úrovní v hierarchii bývá obvykle více; (např. *město -> okres -> region -> země*). Tyto hierarchie se využívají při práci s krychlí při operacích roll-up a drill-down. Při roll-up se přechází na hierarchicky vyšší, obecnější úroveň - zobrazované údaje mají podobu souhrnů, při drill-down se přechází na podrobnější pohled na data; někdy se mluví o různých úrovních granularity pohledu na data.

Základní multidimenzionální model má podobu n-rozměrné krychle. To je však podoba v logickém smyslu. Z hlediska implementačního se nabízí několik možností jak tuto strukturu uložit v počítači. Hlavní důvod pro odlišné fyzické implementace logického modelu je především velká řídkost dat a jejich nestejnoměrné rozmístění. Pro implementaci se nabízejí v zásadě dva přístupy:

- hyperkrychle (hypercube) - jedna velká krychle, která obsahuje nástroje pro práci s řídkými daty; výhodou je jednoduchá struktura a srozumitelnost pro uživatele
- multikrychle (multicube) - více navzájem propojených menších krychlí obsahujících jen několik dimenzí; výhodou je efektivní uložení dat.

Uložení dat v krychli ale nepřináší jen výhody. Za rychlost přístupu k datům platíme zvýšenými nároky (a tedy i cenou) na datový server. Tyto nároky někdy vedou k tomu, že se místo „čistého“ OLAP řešení založeného na datové krychli (někdy nazývaného MOLAP - multidimenzionální OLAP) použije tzv. ROLAP - relační OLAP založený na klasické relační databázi. V tomto druhém případě se OLAP dotazy převádějí do klasických SQL dotazů (viz Obr. 6).




*Obr. 6 MOLAP vs. ROLAP*

MOLAP se hodí pro středně velké, statické aplikace. Příkladem může být analýza historických dat o prodeji nějakého produktu. Vzhledem k tomu, že výpočty souhrnů vyžadují jistý čas, není MOLAP vhodný pro dynamické aplikace, kde jsou požadovány informace z průběžně aktualizovaných dat.

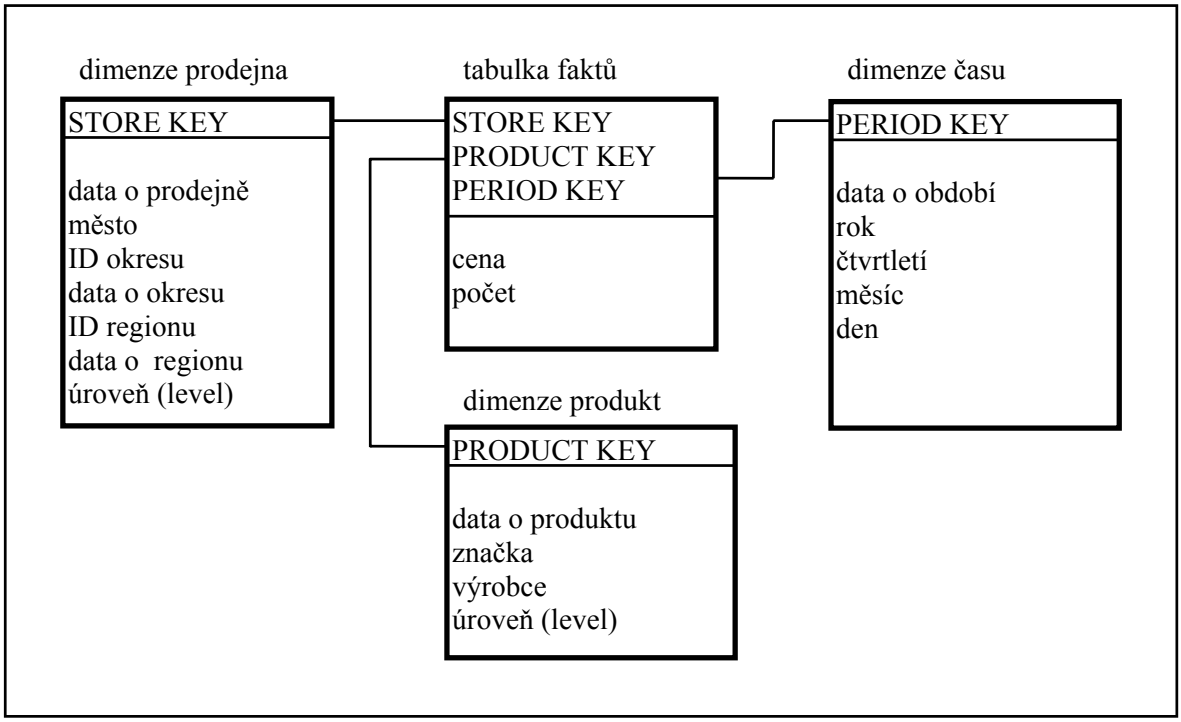
ROLAP je vhodný pro rozsáhlé aplikace hojně využívající transakční data. Výhodou je schopnost zpracovávat rozsáhlá data za použití existujících databázových technologií. Nepoužívají se ale příliš

pro obchodní nebo finanční aplikace. Podobně jako u MOLAP, jsou i zde různé možnosti fyzické implementace systému:

- schéma hvězdy (star schema),
- schéma sněhové vločky (snowflake schema).

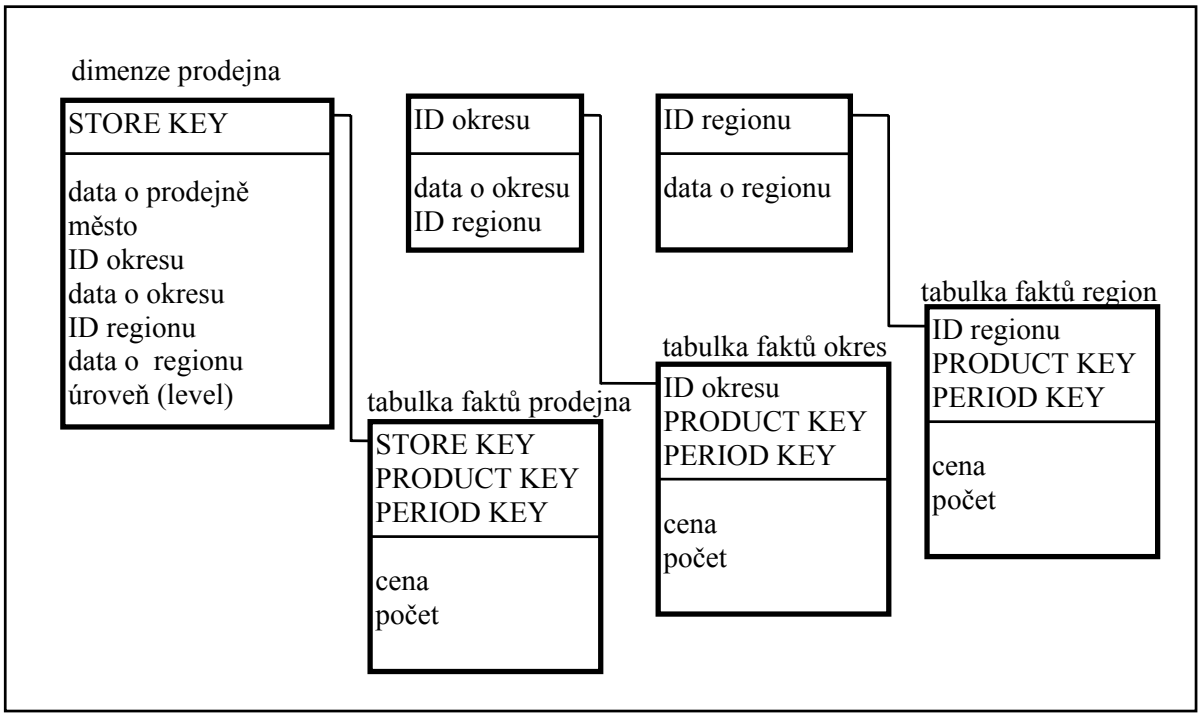
Rozdíl mezi oběma přístupy ilustruje opět příklad evidence prodeje nějakých výrobků. Naše databáze má opět tři dimenze: *Prodejna*, *Produkt* a *Čas*. Dimenze prodejen je tvořena hierarchií *obchod -> okres -> region*, dimenze produktu je tvořena hierarchií *výrobek -> značka -> výrobce*, a časová dimenze je tvořena hierarchií *datum -> měsíc -> čtvrtletí -> rok* [dbms, 2000].

Schéma hvězdy (viz Obr. 7) vychází z jedné centrální tabulky (tzv. *tabulky faktů* - fact table), která obsahuje složený primární klíč (jeden segment klíče pro každou dimenzi) a detailní data (objem prodeje daného výrobku v daném krámu za dané období); může rovněž obsahovat data agregovaná. Pro každou dimenzi existuje jedna tabulka, která obsahuje údaje na různé úrovni příslušné hierarchie (tzv. *tabulka dimenzí*). Úroveň v hierarchii (level) se rovněž zaznamenává jako další indikátor do tabulky dimenzí; je totiž třeba mít přehled o tom, jaké úrovně granularity se konkrétní záznam týká. Indikátor úrovně je tedy nutný při dotazování do tabulky, která obsahuje současně data detailní i agregovaná. Výhodou hvězdy je její srozumitelnost, snadné definování hierarchií, jednoduchá metadata a rychlost přístupu k datům. Nevýhodou jsou problémy s velkými tabulkami dimenzí a předpoklad, že pracujeme pouze se statickými daty, která nejsou aktualizována on-line.




*Obr. 7 Hvězda*

Alternativou ke schématu hvězdy je schéma sněhové vločky. Zde se pracuje s normalizovanými tabulkami dimenzí tak, že každá tabulka nějaké dimenze ukazuje na odpovídající agregovanou tabulku faktů. Tabulky dimenzí obsahují jediný primární klíč pro danou úroveň dimenze spolu s odkazem na nejbližšího rodiče v hierarchii dimenzí. Obr. 8 ukazuje takto upravenou dimenzi prodejen. Při použití sněhové vločky odpadá nutnost používat indikátor úrovně v hierarchii; v každé tabulce jsou údaje jen z jedné úrovně. Výhody této koncepce se projeví ve chvílích, kdy dotazy do databáze se týkají agregovaných hodnot. Nevýhodou je složitá údržba a nárůst počtu tabulek v databázi.




*Obr. 8 Sněhová vločka*

Snadnost použití OLAP v sobě skrývá nebezpečí chybné interpretace dat způsobené nesprávným zobecněním závěrů při přechodu mezi jednotlivými úrovněmi granularity. Na dílčí souhrny uvedené v procentech (relativní četnosti) totiž nelze použít běžné aritmetické operace, což si ne každý uživatel asi uvědomí. Na tento paradox relativních četností upozorňuje (v souvislosti s pravděpodobnostními modely závislostí mezi daty) S. Lauritzen [Lauritzen, 1996]. Ve svém příkladu použil data o rozsudcích vynesených v případě vražd na Floridě v letech 1973-79 (viz Tab. 3).

| | oběť běloch | | oběť černoch | |
|---|---|---|---|---|
| | pachatel běloch | pachatel černoch | pachatel běloch | pachatel černoch |
| rozsudek smrt | 72 | 48 | 0 | 11 |
| jiný rozsudek | 2074 | 238 | 111 | 2309 |

*Tab. 3 Vraždy na Floridě podrobně*

Provedeme-li roll-up (přechod na méně podrobnou úroveň) pro barvu oběti, můžeme dospět k závěru (z Tab. 4), že bylo popraveno relativně více bílých pachatelů ( 72/(72+2185) = 3.2% ) než černých pachatelů ( 59/(59+2547) = 2.3% ). Při pohledu do původní tabulky tab. 3 ale zjistíme, že jak v případě, že obětí byl běloch, tak v případě, že obětí byl černoch, bylo popraveno relativně více černých pachatelů. Ve skupině bílých obětí to bylo $\frac{48}{48+238} = 16.8\%$ černých pachatelů oproti

$\frac{72}{72+2074} = 3.4\%$ bílých pachatelů a ve skupině černých obětí to bylo $\frac{11}{11+2309} = 0.5\%$ černých pachatelů oproti žádnému bílému pachateli.

| | pachatel běloch | pachatel černoch |
|---|---|---|
| rozsudek smrt | 72 | 59 |
| jiný rozsudek | 2185 | 2547 |

*Tab. 4 Vraždy na Floridě bez vztahu k barvě oběti*

Nutno ještě podotknouti, že OLAP není totéž co data mining. Podobně jako je ve statistice rozdíl mezi konfirmační analýzou dat (kdy se pokoušíme vyvrátit resp. potvrdit dříve formulovanou hypotézu) a explorační analýzou dat (kdy hledáme nějaké zajímavé souvislosti), je rozdíl i mezi OLAP (kdy získáváme z dat sumární charakteristiky na zvolené podrobnosti pohledu) a data mining (kdy hledáme v datech nějaké zajímavé souvislosti). OLAP tedy přináší odpovědi na konkrétní, přesně specifikované otázky, ale sám o sobě nic „neobjevuje“.

## 2.4 Datové sklady a datové trhy

Zatímco OLAP představuje nástroj pro analýzu (a vizualizaci) dat o firmě, datový sklad představuje místo, kde jsou analyzovaná data uložena. Podle W. H. Inmona, který v 80. letech zformuloval koncept datového skladu, je datový sklad

- subjektově orientovaný,
- integrovaný,
- časově proměnný,
- leč stálý

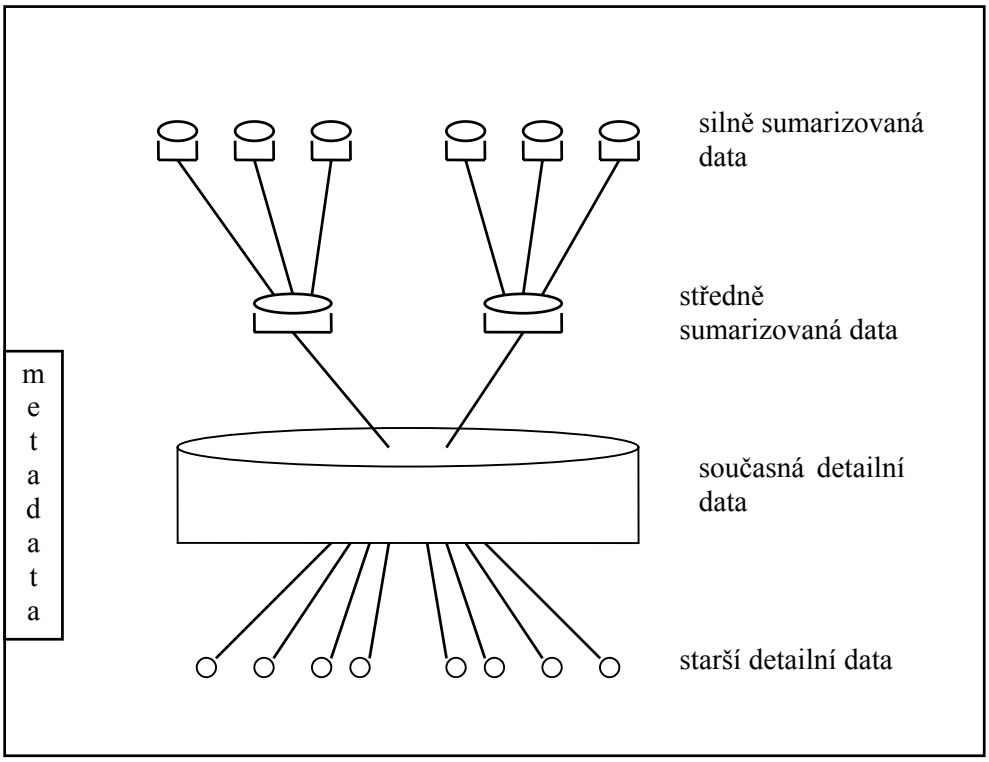
soubor dat, který slouží pro podporu rozhodování [Inmon, 1999].

Prvním charakteristickým rysem datového skladu je, že je *orientován na subjekty* kterými se daná firma zabývá (zákazník, dodavatel, produkt a aktivita). To je výrazný rozdíl od tzv. *produkčních databází,* které se zabývají operacemi a transakcemi jako např. půjčky, faktury, vklady, výběry a pod. Datový sklad neuchovává data, která nejsou vhodná pro podporu rozhodování na manažerské úrovni, produkční databáze uchovávají data potřebná pro operativní řízení bez ohledu na to, zda budou využitelná při strategickém rozhodování.

Vzhledem k tomu, že do datového skladu vstupují data z různých produkčních databází firmy, je důležitá *integrace a sjednocení dat*. Toto integrování zahrnuje sjednocení názvů stejných ukazatelů, sjednocení měřítek (různý způsob měření týchž veličin - např. finanční částky v tisících nebo v jednotkách, v různých měnách apod.), sjednocení kódování (např. pohlaví kódované v jedné databázi hodnotami „M“ a „Z“, v jiné databázi „0“, „1“) apod.

Všechna data v datovém skladu představují „časový snímek“ dat z produkčních databází sejmutý v určitém okamžiku. Datový sklad je aktualizován off-line v určitých časových intervalech (měsíčně, čtvrtletně, ročně - podle povahy dat a předpokládaných analýz) a je rovněž analyzován odděleně od produkčních databází. Výhodou je, že nešetrný zásah do datového skladu (dotaz, který vede k zacyklení) neovlivní operativní řízení firmy. Rovněž odezva na dotaz položený do datového skladu je rychlejší, než by byla odezva do produkční databáze. Produkční databáze je totiž plně vytížena zaznamenáváním transakcí a analytikovi by odpovídala jen okrajově. Nevýhodou je, že data v datovém skladu postupně stárnou. *Časovou proměnností* se tedy myslí v první řadě toto zafixování dat z produkčních databází. Druhý časový aspekt spočívá v tom, že časové údaje jsou v datovém skladu explicitně přítomny jako jedna z důležitých informací.

Dotazy, které do datového skladu směřují uživatelé-analytici, nezpůsobují změnu zde uložených dat. Je tedy datový sklad v tomto smyslu *stálý*.

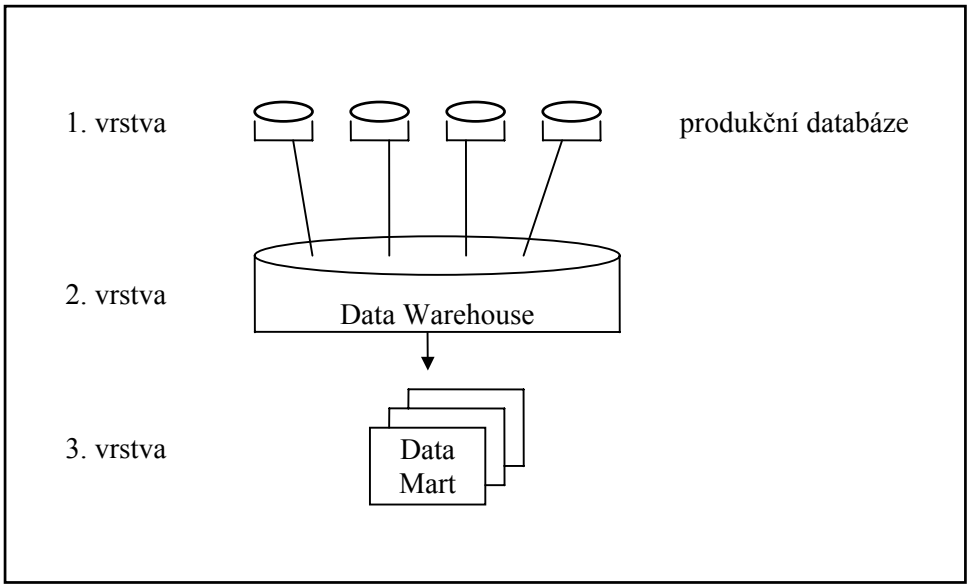



*Obr. 9 Struktura datového skladu*

Struktura datového skladu je uvedena na (Obr. 9). Datový sklad obsahuje obvykle operační data uložená v daném okamžiku, starší (historická) operační data, souhrny na různých úrovních abstrakce a tzv. *metadata*, zachycující informace o datech. Vezměme jako příklad datový sklad obsahující údaje o prodeji nějakých výrobků. Pak současná detailní data mohou být údaje o prodeji v roce 1998, starší detailní data mohou být údaje o prodeji v letech 1990 až 1997, středně sumarizovaná data mohou být souhrnný prodej po skupinách výrobků za jednotlivé týdny nebo souhrnný prodej jednotlivých výrobků po okresech za jednotlivé týdny, silně sumarizovaná data mohou být souhrnný celkový měsíční prodej nebo souhrnný týdenní prodej za jednotlivé kraje, metadata mohou být údaje o tom, které okresy tvoří jednotlivé kraje, které výrobky vytvářejí jednotlivé skupiny, jakým způsobem byly kategorizovány numerické veličiny (např. jak byla rozdělena cena výrobku na cenu nízkou, střední a vysokou) apod.

Vytvoření datového skladu zahrnuje úkoly jako načtení dat, konverzi dat, čištění a transformaci. Data uložená v datovém skladu představují jakýsi neutrální datový prostor, který není vytvářen s myšlenkou konkrétních analýz. Proto se doporučuje vytvářet v návaznosti na datový sklad řadu

specializovanějších *datových tržišť* (data mart) kam se z datového skladu přesunou data relevantní pro určitý typ analýz (pro určité oddělení firmy). Mluví se pak o třívrstvé (three tiered) architektuře datového skladu [Symons, 2000].




*Obr. 10 Třívrstvá (three tiered) architektura datového skladu*

## 2.5 Dotazovací jazyky pro dobývání znalostí z databází

Klasický jazyk SQL, obdobně jako OLAP, umožňuje najít v databázích jen to, co hledáme. V tomto smyslu se tedy ani v jednom případě nejedná o dobývání znalostí (kdy přesně nevíme, co chceme). V posledních letech se ale objevilo několik rozšíření SQL směrem k řešení deskriptivních úloh KDD ([Boulicaut, 1999]).

Jedním takovým systémem je MINE RULE umožňující klást dotazy na asociační pravidla [2]. Příklad dotazu do databáze uvedené v Tab. 1 je na Obr. 11. Syntaxe systému umožňuje provádět výběr záznamů jako v běžném SQL (příkazy FROM, WHERE, GROUP BY, CLUSTER BY), na vybrané záznamy se pak aplikuje algoritmus pro hledání asociací (příkazy SELECT, EXTRACTING RULES). V uvedeném příkladu dotazu tedy pracujeme jen s produkty, které byly zakoupeny ve stejný den ve stejném městě. V takto zvolených záznamech hledáme pravidla tvaru

IF $\text{produkt}_1$ & $\text{produkt}_2$ & ... $\text{produkt}_n$ THEN produkt (SUPPORT, CONFIDENCE)

kde

- SUPPORT (*podpora*) je podíl počtu záznamů, ve kterých současně platí předpoklad (BODY) i závěr (HEAD) pravidla a celkového počtu záznamů vybraných na základě podmínky WHERE
- CONFIDENCE (*spolehlivost*) je podíl počtu záznamů, ve kterých současně platí předpoklad (BODY) i závěr (HEAD) pravidla a počtu záznamů ve kterých platí pouze předpoklad.

[2] O asociačních pravidlech pojednává podrobněji samostatná kapitola. Zde se spokojíme s intuitivním chápáním pravidla jako implikace „jestliže platí předpoklad, platí i závěr“ doplněné o kvantitativní charakteristiky odvozené z počtu záznamů v databázi splňujících předpoklad resp. závěr pravidla.

Dotazem z Obr. 11 do Tab. 1 tedy budou nalezena pravidla:

IF podložky THEN šrouby (0.67, 1.00)
IF šrouby THEN podložky (0.67, 0.67)

```
MINE RULE Priklad AS
SELECT DISTINCT 1..n produkt AS BODY, 1..1 produkt AS HEAD,
        SUPPORT, CONFIDENCE
FROM Prodej
WHERE BODY.město = HEAD.město
        AND BODY.datum = HEAD.datum
EXTRACTING RULES WITH SUPPORT: 0.1, CONFIDENCE: 0.5
```

*Obr. 11 Dotaz v MINE RULE*

V dotazovacím jazyku MSQL [Imielinski, Virmani, 1999] je od sebe oddělena fáze generování pravidel a fáze dotazování na výsledky analýzy. V příkladu na Obr. 12 hledáme v databázi zaměstnanců pravidla, která by nám umožnila na základě věku a pohlaví odvodit, jaké má dotyčný zaměstnanec auto. Kromě hledání asociací nabízí MSQL i paralelní pohled na data a pravidla. Příklad na Obr. 13 ukazuje, jak je možné nalézt všechny záznamy z databáze *Emp,* které odporují pravidlům tvaru *Age* => *Salary* s alespoň 30% spolehlivostí.

```
Emp(Id,Age,Sex,Salary,Position,Car)

GetRules (Emp)
into R
where support > 0.1 and confidence > 0.9

SelectRules (R)
where body has {Age=*), (Sex=*)}
        and head is {(Car=*)}
```

*Obr. 12 Dotaz v MSQL - hledání pravidel*

```
Select *
from Emp
where violates all  (GetRules (Emp)
                    where body is {(Age=*)}
                            and head is {(Salary=*)}
                            and confidence > 0.3)
```

*Obr. 13 Dotaz v MSQL - hledání výjimek*

## Literatrura:


[Berson, Smith, 1997] Berson,A. – Smith,S.J.: Data Warehousing, Data Mining, and OLAP. McGraw Hill, 1997 ISBN: 00-7006-272-2

[Boulicaut, 1999] Boulicaut,J.-F.: Query languages for Knowledge Discovery in Databases. Tutorial PKDD'99

[dbms, 2000] Internet. http://warehouse.chime-net.org/software/genappsw/dbms.

[Dhar, Stein, 1997] Dhar,V. - Stein,R.: Seven methods for transforming corporate data into usiness Intelligence. Prentice Hall. 1997. ISBN 0-13-282006-4.

[Humphries a kol, 2002] Humphries,M. - Hawkins,M.W. – Dy,M.C.: Data warehousing návrh a implementace. Computer Press, 2002, ISBN 80-7226-560-1

[Imielinski, Virmani, 1999] Imielinski,T. - Virmani,A.: A query language for database mining. *Data Mining and Knowledge Discovery*, Vol.3 No.4, 1999, 373-408.

[Info Discovery, 2000] Information Discovery: OLAP and Data Mining. Bridging the gap. White Paper. *DM Review*. Internet. http://www.datawarehouse.com.

[Inmon 1999] Inmon,W.H.: What is a Data Warehouse. Tech Topic. Internet. http://www.cait.wustl.edu/papers/prism/vol11_no1.

[Inmon 2002] Inmon,W.H.: Building the Data Warehouse (3. vydání). John Wiley & Sons, 2002. ISBN: 04-7108-130-2

[Lauritzen, 1996] Lauritzen,S.: Graphical Models. Oxford Statistical Science Series, Clarendin Press, Oxford, 1996.

[Pokorný, Halaška, 1998] Pokorný,J. - Halaška,I: Databázové systémy. ČVUT, 1998.

[ShowCase, 1999] ShowCase Corp.: Accelerating the deployment of a business intelligence system. White Paper.

[Symons, 2000] Symons,V.: Three tiered Data Warehouse structure. White Paper. *DM Review*. Internet. http://www.datawarehouse.com.

[Thomsen, 2002] Thomsen,E.: OLAP Solutions, Building Multidimensional Information Systems. (2. vydání) 2002, ISBN 0-471-40030-4.

[TM1, 1999] TM1 White Paper. Internet. http://www.csm.uwe.ac.uk/~jharney/tm1wppr.ht