Analytické dolování znalostí v reálném čase

Petr Voborník
UHK – FIM – dIZS
petr.vobornik@uhk.cz

Abstrakt

Naměřená či zjištěná data o nejrůznějších aspektech našich životů i jevech okolního světa neustále narůstají a plní databáze různorodých společností a agentur. Ne vždy jsou však tato data mimo svůj primární účel jejich majitelům co platná. Ovšem z většiny z nich lze doslova vydolovat informace a znalosti, které mohou významným podílem pomoci při rozhodování managerů, plánování dalších strategií, ba dokonce i při rozvoji společnosti jako celku. Proces tohoto dolování však není nikterak triviální a získání užitečné relevantní znalosti z velké databáze lze přirovnat k pověstné hledání jehly v kupce sena.

V tomto článku jsou představeny technologie umožňující toto dolování znalostí v reálném čase. Z obecně známých přístupů OLAP a klasického dolování je představena také jejich kombinace OLAM, softwarová řešení z ní vycházející a v závěru pak teorie řešící problematiku plnění datových skladů.

Klíčová slova

OLAM, OLAP, datový sklad, data mining, dolování v datech, vícerozměrná datová kostka, DBMiner, Informetrie, datové překladiště, trigger.

Úvod

V posledních letech stále více narůstá objem dat, která jsou v průměru ročně vyprodukována na jednoho člověka. Pomineme-li data typu digitální fotografie, video a další nestrukturovaná binární data, která se do tohoto průměru také počítají a jejichž datový objem je nemalý, stále ještě velmi podstatnou část těchto globálně produkovaných dat zabírají klasická databázová data. Jejich producenty jsou především velké obchodní firmy, shromažďující například data z obchodních řetězců, doručovatelské firmy, webové vyhledávače, astronomické observatoře, lékařská zařízení atd.

Příbytek těchto dat má neustále rostoucí tendenci. Jejich primární využití (v obchodě třeba vytištění účtenky a sledování stavu zásob, u doručovatelů určení trasy pro obchodní balík, nebo v lékařství výpis stavu naměřené teploty za pobyt pacienta v nemocnici) ale nemusí být tím jediným k čemu lze tato data využít.

Vhodnými metodami analýzy dat se dá například vysledovat, že pokud si zákazník v obchodě koupí zboží X je velmi pravděpodobné, že si koupí i zboží Y, tedy různou prodejnost určitého druhu zboží v reakci na všelijaké další okolnosti. V medicíně jsou pak takovéto analýzy ještě přínosnější. Lze například zjistit že při určité kombinaci léků vzrůstá u některé skupiny pacientů úmrtnost či naopak dochází k jejich rychlejšímu uzdravení.

Ne vždy je však snadné tyto znalosti z onoho obrovského množství dat získat. Obecný princip totiž spočívá ve stanovení nějaké hypotézy a jejím následném ově-

ření, tedy buď zamítnutí či nezamítnutí na určité hladině pravděpodobnosti. Pokud ale do hry vstupuje velké množství proměnných a vztahy mezi nimi nejsou z logiky věci přímo patrné, není vůbec snadné vytvořit dobrou hypotézu, či neopomenout některou, jež by mohla přinést zajímavé výsledky.

Dolování v datech tedy vyžaduje nejen ověřování hypotéz statistickými metodami, ale zároveň i generování těchto hypotéz dle určitých specifikací zadaných uživatelem – analytikem.

Vzhledem k tomu, že může být vyžadováno testování nejrůznějších hypotéz a domněnek, je pro efektivní práci analytika nezbytné, aby nástroje tuto analýzu provádějící měly co nejrychlejší odezvu, tedy aby zadané úkoly zpracovávali v reálném čase (online). Za tímto účelem existuje celá řada různých teorií i v praxi realizovaných řešení.

Jednak je třeba zabývat se uložením analyzovaných dat ve formě, která i přes jejich obrovskému množství dovoluje provádět nad nimi různé výpočetní operace bez nutnosti požadavku na vyšší výpočetní čas. Takováto úložiště s daty připravenými pro analýzu se nazývají datové sklady a vícerozměrné databáze.

Analytické nástroje pro zpracování ověření nejrůznějších hypotéz v reálném čase jsou pak označovány zkratkou OLAP. Hledání znalostí v datech, o nichž v počátku ani nevíme, že by v nich mohly být skryty, se pak nazývá dolování v datech (data mining).

V tomto článku se po úvodním přiblížení teoretických základů zmíněných přístupů zaměříme na teorii pokročilejších technik, která výše tyto přístupy propojuje (OLAM). Dále si představíme dvě existující funkční softwarová řešení na těchto technikách založené. V závěru si pak přiblížíme teorii, která by mohla aspirovat na jedno z možných řešení určitého nesnadného úkolu, na kterém celá teorie dolování znalostí stojí.

Základy

Analytické dolování znalostí v reálném čase (Online Analytical Mining – OLAM) je založeno na propojení a rozšíření mechanizmů OLAP a dolovaní v datech (data mining). Dříve než se tedy budeme věnovat sjednocující teorii, tak si nejprve názorně vysvětlíme principy obou těchto mechanizmů.

Analytické zpracování údajů v reálném čase (OLAP)

Analytické zpracování v reálném čase (Online Analytical Processing - OLAP) – zahrnuje struktury údajů a analytické služby, které slouží pro ke zpracování údajů uložených v datovém skladu do podoby pro koncového uživatele, tedy managery a analytiky v reálném čase. [5]

Termín OLAP zavedl *Dr. E. F. Codd* a definoval také dvanáctero původních pravidel OLAP:

1. Vícerozměrný konceptuální pohled

OLAP by měl poskytovat uživateli vícerozměrný model odpovídající jeho podnikatelským potřebám tak, aby tento model mohl využívat pro analýzu shromážděných údajů.

2. Transparentnost

Technologie systému OLAP, podřízená databáze, architektura a možná heterogennost vstupů datových zdrojů by měly být pro uživatele transparentní, aby mohl naplno využívat svoji produktivitu a odbornost při použití front-end prostředí a nástrojů.

3. Dostupnost

Systém OLAP by měl přistupovat jen k těm údajům, které jsou potřebné pro analýzu. Systém by měl být navíc schopen

přistupovat ke všem údajům potřebným pro analýzu nezávisle na tom, z kterého heterogenního podnikového zdroje tyto údaje pocházejí, jak často jsou obnovovány a podobně.

4. Konzistentní vykonávání

I když počet záznamů a tedy i velikost databáze časem roste, uživatel by neměl pocítit žádné podstatné snížení výkonu.

5. Architektura klient-server

Systém OLAP musí odpovídat principu architektury klient-server s přihlédnutím k maximální ceně a výkonu, flexibilně a interoperabilitě.

6. Generická rozměrnost

Každý rozměr údajů musí být ekvivalentní ve struktuře i operačních schopnostech.

7. Dynamické ošetření řídkých matic

Systém OLAP by měl být schopen adaptovat své fyzické schéma na konkrétní analytický model, který optimalizuje ošetření řídkých matic, přičemž dosáhne a udrží požadovanou úroveň výkonu.

8. Podpora pro více uživatelů

Systém OLAP musí být schopen podporovat pracovní skupinu uživatelů pracujících současně na konkrétním modelu.

9. Neomezené křížové operace mezi rozměry

Systém OLAP musí dokázat rozeznat hierarchie rozměrů a automaticky vykonat asociované kumulované kalkulace v rámci rozměrů i mezi nimi.

10. Intuitivní manipulace s údaji

Pravidlo definuje konsolidované přeorientování cest na detailní úroveň a zpět (drill-down[1] a roll-up[2]). Uživatelské rozhraní by mělo umožňovat všechny manipulace snadno a intuitivně pomocí myši způsobem drag and drop.

11. Flexibilní vykazování

Musí existovat schopnost uspořádat řádky sloupce buňky způsobem, který umožní analýzu a intuitivní vizuální prezentaci analytických sestav.

12. Neomezené rozměry a úrovně agregace

V závislosti na požadavcích podnikání může mít analytický model více rozměrů, přičemž každý z nich může mít vícenásobné hierarchie. Systém OLAP by neměl zavádět (např. z technických důvodů) žádné umělé omezení počtu rozměrů nebo úrovní agregace. [1] [5]

Později bylo těchto 12 pravidel rozšířeno ještě o dalších 6, tedy celkem na 18.

Vícerozměrná databáze

Klasické transakční databáze nejsou pro přímou analýzu OLAP příliš vhodné. Ve stejném čase k nim totiž přistupuje velké množství uživatelů, kteří nejen data čtou, ale i upravují. Složité analytické dotazy přímo nad touto databází jsou pak těmito přístupy značně zpomalovány a zároveň zase i brzdí ony je. Mimoto jsou data v těchto databázích organizována primárně pro jejich vkládání, čtení a úpravu na úrovni jednotlivých datových jednotek (řádků tabulky) a vzhledem k požadavku eliminace chyb jsou jednotlivé tabulky normalizovány tak, aby byly vyloučeny veškeré redundance. Souhrnné analytické dotazy poté vyžadují výpočty napříč mnoha tabulkami zahrnujíce obrovská množství jejich ne vždy indexovaných řádků, což u velkých databází znamená nejen delší odezvu na dotaz, ale i vyšší zatížení serveru.

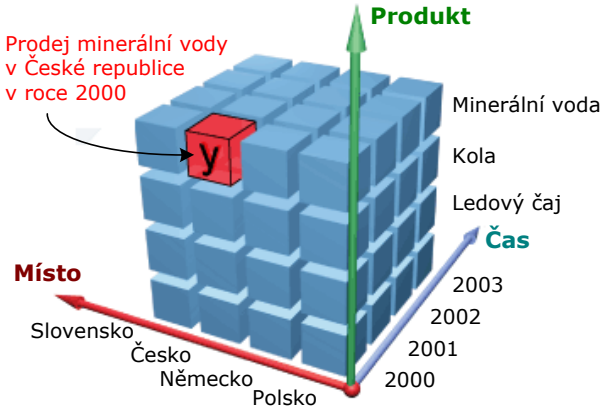
Řešením těchto problémů může být vytvoření zvláštní databáze na bázi vícerozměrných datových struktur. Tyto pak slouží jako zdroj pro získávání sumarizovaných a agregovaných údajů, čili informací. Do

[1] Drill-down – rozšíření o jeden rozměr (zavrtávání se hlouběji, detailněji), např. přidání evidence nového státu či rozložení jejich sum pro jednotlivé města
[2] Roll-up – redukce rozměrů o jeden (globálnější pohled na data), např. zrušení evidence jednoho státu, či pohled na sumy za celé kontinenty

těchto databází se ukládají již pouze „vyčištěné" údaje. Tabulky v databázi nejsou normalizované, přičemž rozlišujeme tabulky faktů a rozměrů (dimenzí). Tabulky faktů pak většinou obsahují velké množství redundantních dat.

K hlavním výhodám vícerozměrných databází patří rychlý a komplexní přístup k velkým objemům údajů, přístup k vícerozměrným datovým strukturám, možnost komplexních analýz a účinné možnosti pro modelování a prognózy. Nevýhodou je však vyšší nárok na kapacitu úložiště, problémy při změnách rozměrů atd.

Datový model vícerozměrné databáze je možné představit si jako vícerozměrnou kostku. Ta je v podstatě ekvivalentem tabulky v relační databázi. Hlavním rozdílem je však fakt, že prostor pro celou kostku je už dopředu rozvržen a rezervován. Jednotlivé záznamy se pak nacházejí v průsečících rozměrů. Na Obr. 1 je vyobrazen příklad trojrozměrné datové kostky s ukazatelem na její jeden agregovaný datový údaj – suma prodejů určitého produktu v určité zemi ve zvoleném roce.



Obr. 1 – Příklad třírozměrné datové kostky [3]

Společně s rostoucím počtem rozměrů kostky pak ovšem vrůstá i exponent požadavku na úložnou kapacitu. Ne ve všech segmentech kostky jsou přitom uloženy hodnoty. Díky tomu lze efektivně použít kompresy datového souboru a redukovat tak zabíranou diskovou kapacitu.

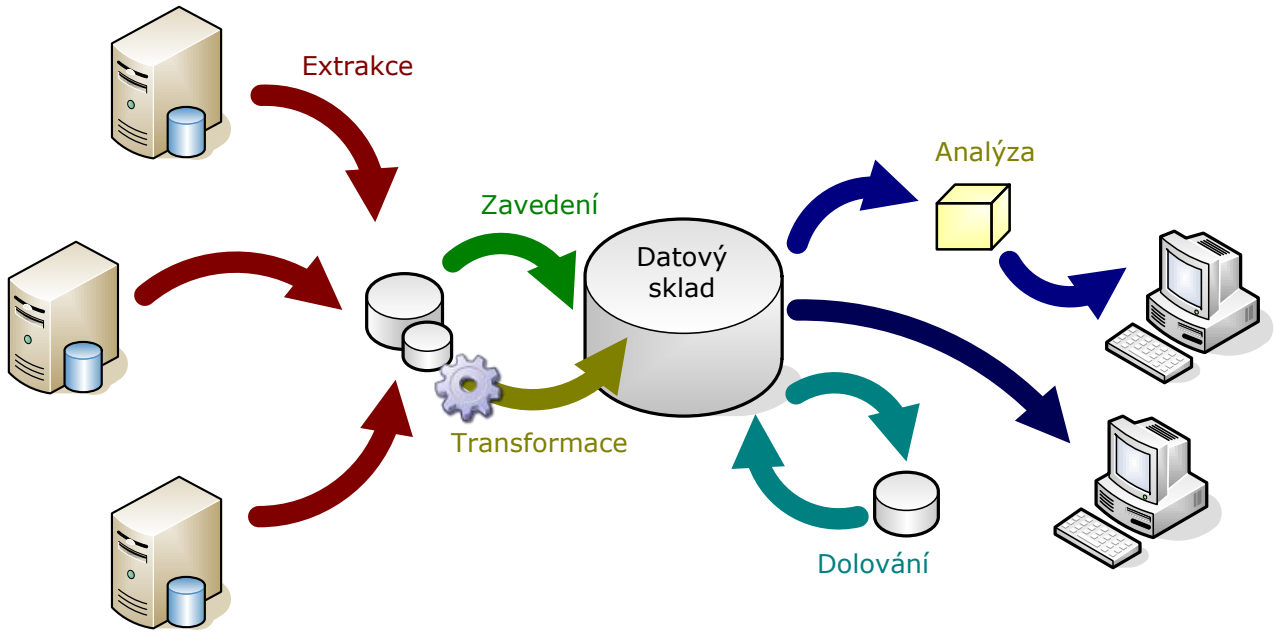
Jednotlivé rozměry zároveň mohou být i hierarchicky rozloženy na více podrobnější „podrozměry". Například kromě sum (maxima, minima, průměru, směrodatné odchylky...) za jednotlivé roky pro každý produkt a místo mohou být uloženy také tyto sumy za jednotlivá čtvrtletí, měsíce, týdny i dny. Stejně tak v případě místa lze ve vícerozměrné kostce schraňovat sumy za jednotlivé regiony, okresy, města a obchody. U produktů pak je možné rozlišovat jejich nejrůznější kategorie, podkategorie, výrobce, příchuti atd.. Takto postupně dělený model se nazývá schématem sněhové vločky a umožňuje zavrtávání se (drill-down) do podrobností příslušeného rozměru či naopak pohled na globálnější data (drill-up). [5] [6]

Úložiště vícerozměrných údajů

Jak již bylo řečeno, úložiště vícerozměrných údajů v drtivé většině případů není součástí hlavní transakční databáze, do níž se data ukládají. Je tedy uloženo zvlášť, mnohdy i fyzicky na jiném serveru. Tyto databáze se nazývají datovým skladem. Nad nimi jsou pak prováděny samotné OLAP analýzy (viz Obr. 2).

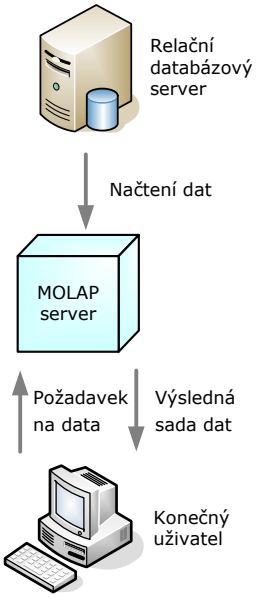
Převod dat z operační databáze do datového skladu lze pak provádět mnoha způsoby. Jednak je možné rozlišovat úplné kopírování či přírůstkové, a dále i čas, kdy k tomuto dochází. Ten je obvykle zvolen periodicky (každou noc, každou hodinu, o víkendu...).

[3] http://www.techxii.com/wp-content/uploads/2006/12/cubes.gif *(upraveno)*



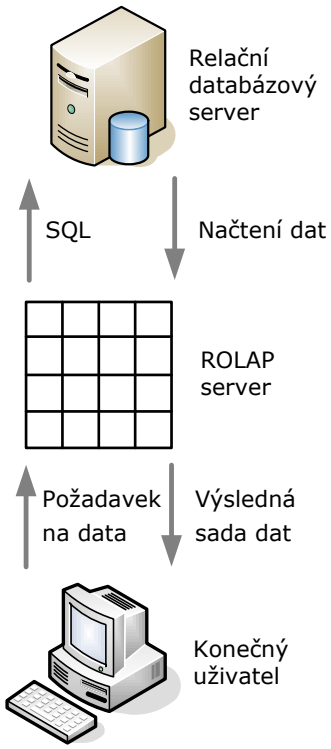
Obr. 2 – Schéma datového skladu [6]

Vícerozměrné analytické zpracování v reálném čase (Multi-dimensional OLAP - *MOLAP*) používá speciální datové struktury a vícerozměrné databázové systémy pro organizaci, navigaci a analýzu dat. Data jsou získávána buď z datového skladu nebo přímo z operačních zdrojů. Hlavní výhodou je maximální výkon vzhledem k dotazům uživatelů, nevýhodou je ale redundance údajů, jelikož ty jsou uloženy jednak v relační databázi a zároveň ve vícerozměrné databázi. [1] [5]



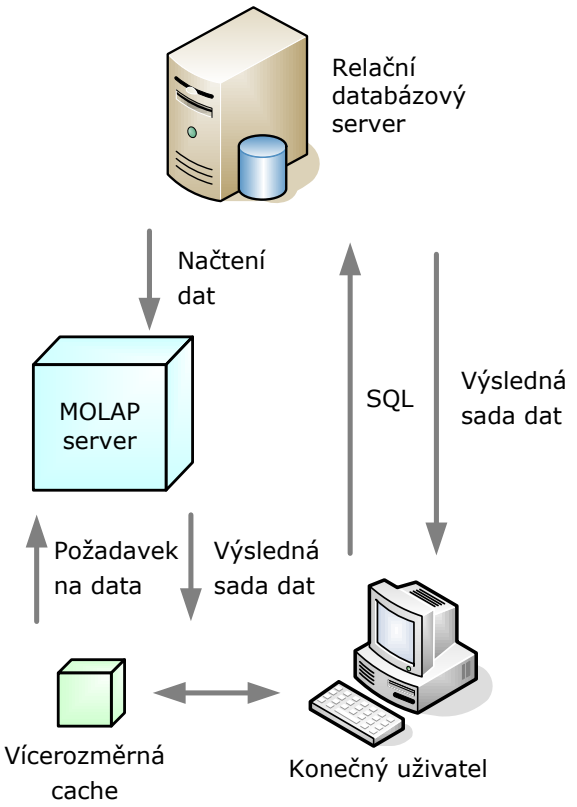
Obr. 3 – Architektura MOLAP [1]

Relační analytické zpracování údajů v reálném čase (Relational OLAP - *ROLAP*) získává údaje z analýzy relačního datového skladu. Tyto údaje se po zpracování překládají uživateli jako vícerozměrný pohled a ukládají se v relační databázi, díky čemuž nevzniká problém s jejich redundancí. [5]



Obr. 4 – Architektura ROLAP [1]

Hybridní online analytické zpracování (*HOLAP*) je kombinací úložišť MOLAP a ROLAP, přičemž se snaží využívat výhod jednotlivých typů úložišť a zároveň eliminovat nevýhody. Údaje tak zůstávají v relačních databázích a jejich vypočtené agregační hodnoty se ukládají do vícerozměrných struktur. Při dotazování se poté vybrané údaje ukládají do vícerozměrné paměti cache pro urychlení další práce. [5]



Obr. 5 – Architektura HOLAP [1]

Dolování v datech

Dolovní v datech (data mining) je proces extrahování validních, dříve neznámých, srozumitelných a využitelných informací a znalostí z velkých databází a jejich využití pro obchodní či jiná kritická rozhodnutí. [1]

Tradiční dotazové prostředky či nástroje pro datové sestavy se týkají realizací dotazů na to, *co* je v databázi. Nástroje OLAP jdou dál, pomáhají zjistit, *proč* jsou některá fakta pravdivá. Klíčovým rozdílem mezi dolováním dat a OLAP je, že OLAP je řízený uživatelem. Analytik vytvoří hypotézu a nástroj OLAP použije k jejímu ověření. Tento přístup tedy spoléhá na intuici analytika, který nakonec zformuluje dotaz a zjemňuje analýzu kladením dalších, potencionálně složitějších dotazů. Když je však počet proměnných (atributů) v řádu desítek či stovek, je pak již obtížné sestavit kvalitní hypotézu. Dolování v datech představuje nástroje, které generují hypotézy. Objevování v OLAP systémech je naváděno uživatelem, kdežto nástroj pro dolování provádí toto objevování sám. Oba nástroje se však mohou vhodně doplňovat: Dříve než se uživatel rozhodne pomocí vzorů odkrytých dolováním, může si hypotézu ověřit OLAP nástroji. [9]

Dolování v datech je tedy proces analýzy dat z různých perspektiv a jejich přeměna na užitečné informace. Z matematického a statistického hlediska jde o hledání korelací, čili vzájemných vztahů nebo vzorů a testování hypotéz. Principiálně je založeno na heuristických algoritmech, neuronových sítích a dalších pokročilých softwarových technologiích a metodách umělé inteligence. Pomáhá sledovat a analyzovat trendy a předvídat události. [5]

Modely dolování v datech

Dolování v datech odkrývá vzory budováním modelů, přičemž rozlišujeme dva jejich typy: prediktivní a deskriptivní. V *prediktivních modelech* je cílem předpovědět hodnoty nějakých vlastností na základě již známých hodnot vlastností jiných. Z historie splátek lze například vybudovat model pro identifikaci osob, které pravděpodobně nesplatí půjčky. V *deskriptivních modelech* se popisují vzory v existujících datech, jenž mohou ovlivňovat rozhodování. Hlavním rozdílem mezi oběma typy modelů je, že v prediktivních modelech se provádí predikce explicitně, naopak deskriptivní modely pomáhají konstruovat prediktivní model nebo učinit implicitní predikci následovanou akcí nebo rozhodnutím.

Mezi disciplíny, na kterých je založeno dolování dat, patří především induktivní učení, strojové učení a statistika. Z nich také vychází používané modely. Pro řešení obchodních problémů se v praxi často uvažuje základních šest typů modelů: klasifikace, regrese, časové řady, shlukování, asociační analýza a objevování posloupností.

Algoritmy *klasifikace* bývají často založeny na rozhodovacích stromech či neuronových sítích. Použití klasifikačních algoritmů začíná na trénovací množině předem klasifikované podmnožiny transakcí. Výpočtem hodnoty kategoriální proměnné přiřazuje klasifikace instance či případy do skupin nebo tříd. Získáním vzorů lze porozumět existujícím datům a předpovědět nové instance.

Regrese se používá k předpovědi jaké hodnoty budou následovat, a vychází z řady existujících hodnot a jejich atributů. V nejjednodušším případě je možné použít standardní statistické techniky, jako je například lineární regrese. Pro regresi se používají algoritmy neuronových sítí, ale i třeba rozhodovací stromy.

U předpovídání podle *časových řad* je výchozí idea podobná jako u regrese, tedy z existujících hodnot se předvídají hodnoty budoucí. Rozdíl je ovšem v tom, že u časových řad jsou hodnoty závislé na čase. Modely berou v úvahu různou časovou granulitu, velikost pracovního týdne, kalendářní výjimky jako prázdniny, svátky a podobně.

Shlukování databázi segmentuje do několika částí (skupin). Cílem je, aby se skupiny co nejvíce navzájem lišily a současně obsahovaly co nejvíce podobné prvky. Na rozdíl od klasifikace není známo, podle kterých atributů jsou nebo mají být data shlukována. Z toho plyne, že analytik musí přiřadit těmto shlukům význam. Záznamy mají obvykle velké množství atributů a jsou rozděleny do relativně malého množství skupin. Shlukování bývá často při dolování v datech počátečním krokem. Identifikuje skupiny vztažených záznamů, které mohou být dále použity pro objevování dalších vztahů.

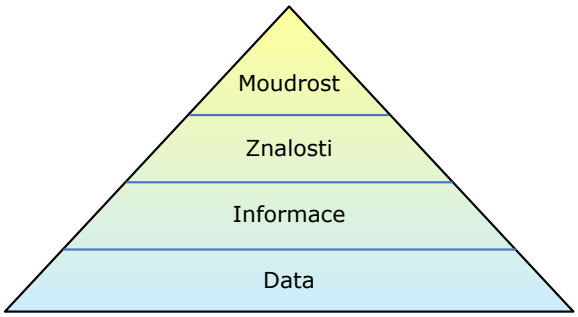
Asociace (vztahy) jsou dány prvky, které se vyskytují pohromadě v jednom záznamu nebo události. Nástroje pro asociace objevují pravidla tvaru „Je-li prvek A součástí události, pak z *x* procent času je i prvek B částí události". Parametr *x* se nazývá faktor důvěryhodnosti. Mezi klasické příklady asociací patří ty, které se objevují v analýzách nákupních košů. Nalezení pravidla může vést ke spuštění další, přídavné analýzy.

Objevování posloupností je spjato s analýzou asociací, jejichž prvky jsou rozmístěny v časové ose. Například je objevena znalost „Jestliže je proveden zákrok *X*, pak v se 55% případů následně objeví infekce". Není tedy zajímavé pouze nalézt seskupení prvků v jedné události, ale nalézt také uspořádání jejích jednotlivých operací, případně délku časových intervalů mezi nimi. [9]

Od dat k moudrosti

Jak již bylo řečeno cílem zde prezentovaných technik je z dat získat informace či znalosti. Pozastavme se tedy na chvíli nad významy těchto jednotlivých pojmů.

Data se chápou jako náhodné skupiny jednoduchých faktů nebo událostí, složitější fakta, která vzniknou agregací jednoduchých fakt reprezentují *informace*. *Znalosti* jsou výsledkem našich procesů vnímání, jsou organizovány tak, že z nich mohou být odvozovány závěry. Pod *moudrostí* se všeobecně chápe společný smysl či dobrý soud (viz Obr. 6). [9]



Obr. 6 – Pyramidové uspořádání pojmů – od dat k moudrosti [9]

Základním stavebním prvkem všeho vědění jsou data. Ta mohou být vyjádřena a zaznamenána nespočtem různých způsobů, avšak abychom z nich mohli získat (přečíst) informace, musí tato data splňovat dva předpoklady: musí nějaké informace obsahovat a musí být zaznamenána takovým kódem, který jsme schopni přečíst.

Informace, jež některá data obsahují, pak samy o sobě pro nás ještě stále nejsou ničím významné. Je totiž třeba zapojit znalosti, abychom na základě těchto informací mohli něco smysluplného vykonat, či z nich něco odvodit – zkrátka se rozhodnout. Základem znalostí jsou opět informace, ovšem je potřeba také inteligence schopná pozorovat a uvědomovat si vztahy mezi nimi. Za moudrého pak označujeme člověka, který má určité znalosti a na základě informací je dokáže použít tak, že dosahuje požadovaných cílů.

Budeme-li prezentovat rozdílnost těchto pojmů na příkladu, pak data mohou být reprezentována třeba údaji o počasí v určité lokalitě za několik let (např. Morava, 3.3.05, déšť; Morava, 4.3.05, déšť...). Informací potom bude fakt, že na Moravě v březnu průměrně 23 dní z 31 prší s odchylkou ±3 dny. Znalost pak prokážeme závěrem, že na Moravě většinu března prší a bylo by lepší si na cestu tam v tomto období vzít deštník. Za moudrost lze považovat závěr spojení těchto znalostí a znalostí hospodářských rozhodnutím, že nejvýhodnější bude jarní plodiny pěstovat na Moravě a řídit se podle toho.

Analytické dolování znalostí v reálném čase (OLAM)

Žádný prostředek či množina prostředků pro dolování v datech nejsou univerzálně aplikovatelné. Mimo to, stejné typy modelů mohou být vybaveny různými algoritmy, přičemž stejné algoritmy mohou mít různou implementaci. Je tedy vhodnější vyzkoušet více algoritmů pro jeden typ modelu a díky tomu mít možnost nalézt „nejlepší řešení".

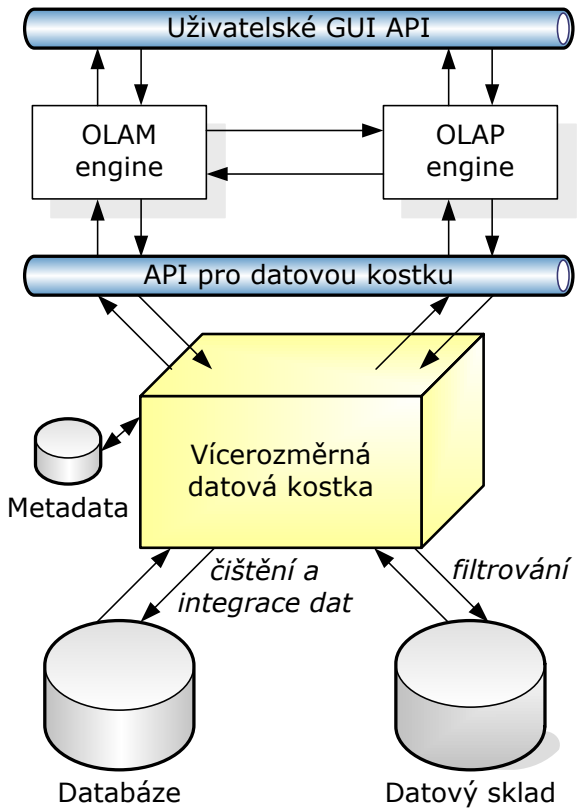
Výzkum v oblasti dolování v datech vyústil v posledních letech minulého století v nové architektury integrující OLAP s dolováním znalostí. Vícerozměrný pohled na data, který je podstatou technologie OLAP, přináší novou situaci a novou nabídku pro dolování, ve srovnání s „plochými daty" uloženými v klasických databázových tabulkách. S rozměry v OLAP souvisí charakterizace dat, tedy budování hierarchie pojmů.

Propojení OLAP a dolování v datech bylo základem pro vznik analytického dolování znalostí v reálném čase (Online Analytical Data Mining – OLAM), jehož základní ideou je provádět dolování nad vícerozměrnou datovou kostkou. Prvním představitelem této generace systémů je software DBMiner [4] [3] [2]. Tento systém poskytuje řadu funkcí pro dolování v datech (charakterizace, porovnávání, asociace, klasifikace, predikce a shlukování). Dolování se pak provádí interaktivně na různých částech vícerozměrné databáze a v různých úrovních abstrakce. Architektura (viz Obr. 7) ukazuje dokonce integraci OLAM a OLAP. Mezi vícerozměrnou datovou kostkou pro OLAP a pro dolování není významný rozdíl. Architekturu lze tedy vybudovat nad existujícím OLAP systémem. Na druhé straně, OLAM může vyžadovat jisté jemnější vlastnosti implementace z hlediska například nároků na data pro zpracování analýz. Analýza pomocí dolo-

vání je z hlediska nároků na data obvykle složitější než OLAP. [9]

Architektura OLAM

OLAM engine vykonává analytické dolování v datové kostce stejným způsobem jako OLAP engine, tudíž je nasnadě propojit architektury OLAM a OLAP, tak jak ukazuje Obr. 7. OLAM a OLAP společně přijímají uživatelské dotazy a zpracovávají je analýzou vícerozměrné datové kostky. Kromě toho OLAM engine může zpracovávat dotazy přímo nad vícerozměrnými daty. Z toho vyplývá, že OLAM je mnohem více sofistikovaný než OLAP, jelikož obsahuje více dolovacích modulů, které vzájemně spolupracují, díky čemuž dosahují efektivnějších výsledků. [3]

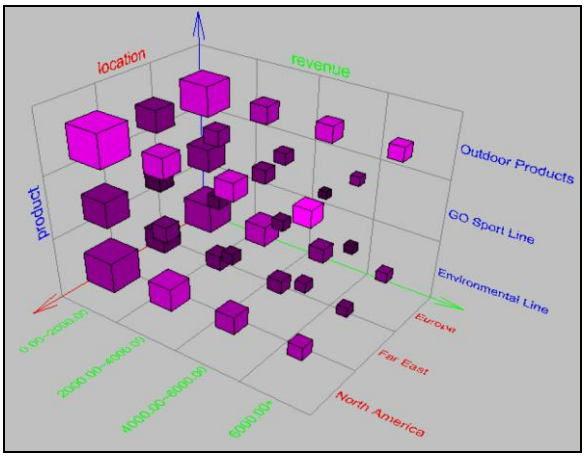


Obr. 7 – Integrace OLAM a OLAP architektury [3]

DBMiner

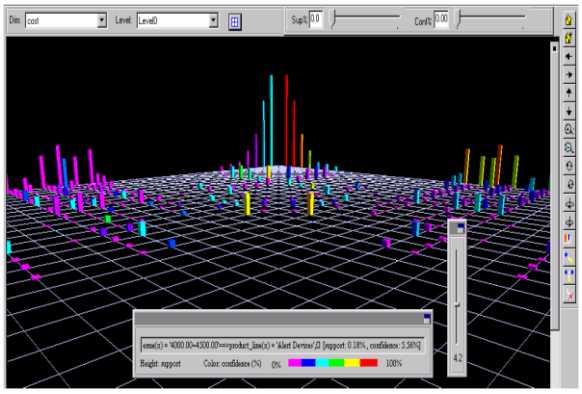
Systém DBMiner tedy integruje technologie OLAP a dolování v datech. Systém dokáže efektivně a účinně dolovat různorodé druhy znalostí na různých úrovních abstrakce z velkých relačních databází a datových skladů a charakteristické pro něho jsou tyto rysy:

- Přináší několik zajímavých dolovacích technik, například datovou kostku a OLAP, vyhledávání zaměřené na vlastnosti, statistické analýzy, progresivní prohledávání pro dolování víceúrovňových pravidel a metapravidla řízená dolováním znalostí.
- Systém vykonává interaktivní dolování v datech na různých úrovních abstrakce na každou uživatelem specifikovanou datovou sadu v databázi nebo datový sklad podporující dotazovací jazyk pro dolování v datech (Data Mining Query Language - DMQL) vycházející ze SQL, nebo grafické uživatelské rozhraní. Uživatelé mohou interaktivně nastavovat a přizpůsobovat různé rozsahy, řídit proces dolování a vykonávat operace *roll-up* a *drill-down*.
- Byly vytvořeny účinné implementační techniky využívající datové struktury, které zahrnují vícerozměrné datové kostky a zobecněné vztahy. Při realizaci došlo k jednoduchému propojení relačních databázových systémů a datových skladů.
- Procesy dolování v datech mohou využívat uživatelsky či expertně definovanou sdruženou sadu nebo návrh úrovňového schématu hierarchií, které mohou být určeny proměnlivě, dynamicky přizpůsobovány podle rozdělení dat a pro číselné položky řízeny automaticky. Tento návrh hierarchií je hlavní propojující komponentou systému a je uložen v relační databázi.
- Systém je založen na klient-server architektuře a běží pod operačním systémem Windows. Lze jej propojit s mnoha různými komerčními databázovými systémy pro dolování v datech, které podporují technologii rozhraní ODBC. [4]



Obr. 8 – Procházení trojrozměrné datové kostky v DBMineru [3]

DBMiner byl úspěšně testován na několika středně velkých a velkých databázích. Některé dolovací moduly pak byly vytvořeny a přidány postupně až během procesu dalšího vývoje systému. [4]



Obr. 9 – Asociační plocha pro vizualizaci dvourozměrných vztahů v DBMineru [3]

DMQL

DBMiner umožňuje používat jak dotazovací jazyk pro dolování v datech DMQL založený na SQL, tak grafické rozhraní pro interaktivní dolování víceúrovňových znalostí.

```
mine charakteristic rules
with respect to produkt, misto,
     datum, sum(prodane_mnoztvi)%
from prodej
where kategorie = "napoje"
```

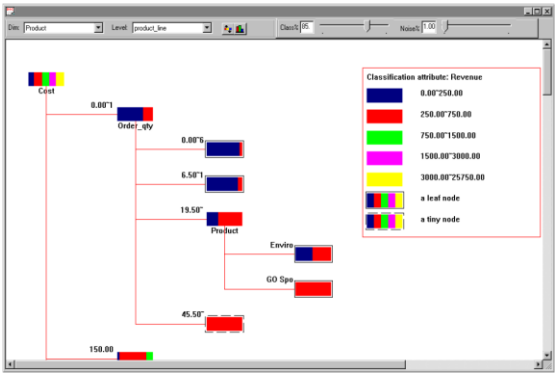
Příklad 1 – DMQL příkaz pro dolování charakteristických pravidel určujících vztah prodaného množství nápojů v určitém místě a čase [4]

Položení dotazu v DMQL DBMineru má následující průběh: Systém shromáždí relevantní data týkající se dotazovaných položek pomocí klasických relačních dotazů, sestaví vícerozměrnou datovou kostku, agreguje data na určitých úrovních hierarchie a prezentuje výsledky v několika názorných formách jako jsou například tabulky, sloupcové a koláčové grafy, křivky, klasifikační stromy, různé formy zobecněných pravidel, jejich vizuální prezentaci atd.

```
mine associations
with respect to misto_narozeni,
     obor, bydliste, avg(znamka)%
from studenti S, hodnoceni H
where H.student_id = S.student_id
set rule template
          obor(s:student, x) ∧
          Q(s,y) → R(s,z)
```

Příklad 2 – DMQL příkaz pro hledání spojení mezi místem narození, místem současného bydliště, studijním oborem a průměrným hodnocením studentů, přičemž je nadefinováno, že hlavním kategorizačním parametrem je studijní obor [4]

Dotazovací jazyk pro dolovaní v datech typu DMQL je standardní součástí dolovacích funkcí vyvíjených dolovacích systémů spolupracujících s komerčními relačními databázovými systémy. Z DMQL mohou být odvozeny různé druhy grafického uživatelského rozhraní. Tato příslušenství pak umožňují interaktivně sestavit a upravovat dotazy pro dolování v datech, měnit hierarchickou strukturu úrovní, různě modifikovat prahy, volit si výstupní formy pro prezentaci nalezených výsledků, zvyšovat či snižovat počty rozměrů a řídit celý dolovací proces během jeho vykonávání. Takovéto rozhraní je implementováno i v DBMineru. [4]



Obr. 10 – Grafický výstup klasifikačního modulu v DBMineru [3]

Informetrie

Jedním z dalších softwarových řešení tohoto přístupu je modifikovaná verze Informetrie. Informetrie je komplexní vícerozměrná aplikace, obsahující data, která souvisejí s různými kontexty, jež přitom mohou některé rozměry sdílet.

V dnešní době datový analytici většinou neumí programovat, neovládají databázové techniky apod. Proto je jimi vysoce ceněno vysokoúrovňové a intuitivní deklarativní OLAP rozhraní. Hodně současných vícerozměrných rozhraní a v nich implementovaných dotazovacích jazyků je zaměřených na ovládání. Uživatel v nich pak může v jednu chvíli textově nebo vizuálně definovat jednu operaci tak, že specifikuje dotaz interaktivně provedením série OLAP operací ve výchozí datové kostce. Z každé operace vzejde nová pomocná kostka sloužící jako základ pro další operace. Samozřejmě ne všechny přechodné kostky jsou pro uživatele důležité, načež je žádoucí k této stupňovité specifikaci najít alternativu.

Dotazovací jazyk implementovaný do Informetrie je založený na teorii proměnných v logickém programování a v deduktivních databázích. Tato teorie umožňuje přímo, flexibilně a intuitivně specifikovat komplexní vícerozměrné dotazy.

Logický model obvykle zobrazuje dostupná data z uživatelovy perspektivy. Tento logický model však představuje variantu hvězdicového schématu, protože obsahuje fakta a tabulky rozměrů.

Vícerozměrná databáze se skládá ze souboru vícerozměrných datových kostek, ze souboru tabulek rozměrů a souboru hierarchických tabulek. Dále tento model zahrnuje výkonné mechanizmy pro specifikaci sémantického propojení dat z rozličných tabulek. Při vizualizaci tabulky se rozlišuje mezi obsahem (úroveň příkladů) a jejími strukturálními aspekty (schematická úroveň).

Tabulky rozměrů popisují jejich specifické vlastnosti. Pro každou rozměrovou vlastnost může existovat nanejvýš jedna tabulka rozměru. Na rozdíl od původního hvězdicového modelu, v tomto mohou být rozměry datové kostky bez tabulek rozměrů. Na schématické úrovni se tabulka rozměru skládá ze jmen vlastností, přičemž na úrovni příkladů pozůstává z hodnot těchto vlastností.

V Informetrii musejí být souhrnná data celkem často analyzována na základě komplexních kritérií, která závisí na jednotlivých vlastnostech rozměrů. Aby bylo možné tato kritéria specifikovat, měla by být informace použita ve více tabulkách rozměrů. Z toho vyplývá, že sémantické vztahy mezi tabulkami rozměrů se vytvářejí na základě sdílených hodnot některých vlastností v rozměrových tabulkách.

Hodnoty vlastností rozměrů v datových kostkách jsou vyjádřeny podle nejnižší úrovně definované pro rozměr. Hierarchické úrovně hierarchické tabulky jsou definované seskupenými hodnotami nejbližší nižší úrovně podle nějakého principu. Při vytváření hierarchických úrovní se nejdříve seskupují hodnoty některých dimenzionálních atributů. Informetrická analýza často využívá zvláštních individuálních hierarchií, které se těžko definují předem. Tento přístup proto nabízí uživatelům flexibilní mechanizmus pro definování nových hierarchií.

Cílem přístupu je pak především nabídnout uživateli srozumitelný, flexibilní a uživatelsky příjemný vícerozměrný dotazovací jazyk pro vícerozměrnou analýzu. Tento dotazovací jazyk nevychází ze SQL, ale je orientovaný vizuálně. Umožňujeme tak uživateli charakterizovat vícerozměrné informace jenom pro výsledný náhled a nemuset specifikovat jednotlivé operace vedoucí k jeho vytvoření. Navíc, uživatel se jenom minimálně pohybuje mezi jednotlivými daty (tabulkami).

V Informetrii je ale potřebné analyzovat souhrnná data ve vícerozměrné databázi na základě specifických kriterií vztahujících se k vlastnostem rozměrů. Nelze totiž předpokládat, že v těchto velkých databázích má uživatel přehled o tom, které hodnoty vlastností odpovídají určitým kritériím. Taktéž nelze ani očekávat, že je uživatel schopen je všechny vyjmenovat. Proto byl navržen účinný ale jednoduchý nástroj pro specifikaci kritérií pro vlastnosti rozměrů. Ve složitějších případech musí uživatel při specifikování kritérií týkajících se vlastností rozměrů navigovat mezi několika rozměrovými tabulkami. Jelikož uživatel sám provádí navigaci, je žádoucí aby ji mohl realizovat přirozeným způsobem.

V tomto jazyce je dostačující, že uživatel správně interpretoval, na které hodnoty proměnná odkazuje.

Kromě proměnných lze ve výrazech odkazujících na tabulky dimenzí použít konstanty. Ty indikují, že uživatele zajímají jenom řádky, ve kterých má vlastnost tuto konstantní hodnotu.

Pokud se objeví stejná proměnná pro různá pod-kritéria v konjunkci, pak se jedná o takzvanou sdílenou proměnnou.

Navigace mezi dvěma tabulkami rozměrů je vyjádřena jednoduše sdílenou proměnnou. Jelikož sdílená hodnota musí mít v obou tabulkách rozměrů stejnou hodnotu, je možné přes tuto hodnotu propojit ta data z obou tabulek dimenzí, která jsou v sémantickém vztahu.

Tento přístup prezentuje vícerozměrný datový model pro Informetrické aplikace, je složený z datových kostek, tabulek rozměrů, hierarchických tabulek a vztahů mezi nimi. Model představuje účinný nástroj pro analýzu v Informetrii. Vytvořen byl i vizuálně orientovaný dotazovací jazyk. Tento přístup umožňuje uživateli specifikovat vícerozměrné informace pro výsledný náhled bez toho, aby musel definovat operace potřebné k jeho vytvoření. Uživatel jenom vyplňuje pole v kostrách tabulek. Dotazovací jazyk, je více deklarativní než běžné, na ovládání orientovaní jazyky nebo OLAP dotazovací jazyky vycházející z SQL. Tato teorie proměnných převzatá z deduktivních databází umožňuje uživateli intuitivně a kompaktně specifikovat kritéria pro vlastnosti dimenzí. Pro podporu deklarativní formulace dotazu byl vytvořen dotazovací jazyk tak, aby uživatel nemusel definovat navigaci mezi datovými kostkami. [7]

Datový sklad plněný triggry

Jednou z problematik, která vyvstává u datových skladů a vícerozměrných databází je systém jejich plnění daty. Obvyklým řešením je jejich periodické načítání z operačních databází v době, kdy je jejich zatížení minimální (v noci, o víkendu, o svátcích apod.). Existuje však celá řada informačních systémů nad databázemi, u nichž se sice možná najde časové období se slabším provozním zatížením než je jinak běžné, nicméně i v něm je operační databáze vytížena natolik, že by její hromadné čtení systémem extrahujícím data pro datový sklad bylo příčinou minimálně významného zpomalení transakčních operačních činností, což může být nepřípustné.

Na druhé straně pak může být požadavek, aby data v datovém skladu byla co možná nejaktuálnější. To má zase za důsledek maximální možné zkrácení periody mezi jednotlivými aktualizacemi dat a tedy i vyšší a častější zatížení operačních serverů. Jak by mohl být celý tento problém vyřešen pouze s minimálním dodatečným zatížením operačních serverů a přitom s okamžitou aktualizací datového skladu si ukážeme v následujících podkapitolách této části.

Inkrementální přístup

Datový sklad může z operačních databází při své aktualizaci vždy načíst veškerá její data bez ohledu na to, četl-li je již při poslední aktualizaci či nikoli, nebo dohledat pouze změny v těchto datech a ty promítnou do agregovaných hodnot ve svém úložišti. Tomuto výpočetně méně nákladnému způsobu můžeme říkat inkrementální (přírůstkový).

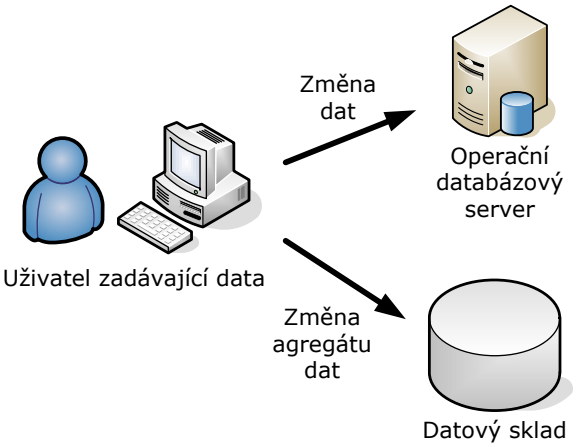
I u tohoto přístupu však musí dojít ke čtení dat z operační databáze. V některých případech lze ovšem toto čtení omezit pouze na určitou časově vymezenou (od poslední aktualizace) část logu, případně změny dohledávat podle časových značek. Systém časových značek však není nejvhodnější, jelikož u operací typu *update* či *delete* je pak třeba v databázi alespoň do průběhu nejbližší následující aktualizace uchovávat i předchozí verze hodnot, aby mohla být podchycena změna v jejich agregátu.

Dalším neduhem pak stále zůstává určitá prodleva mezi zadáním či změnou nových dat a možností s nimi pracovat při analýze.

Jak tedy docílit toho, aby se změněná data dostala co nejdříve do datového skladu a ten je přitom nemusel z operační databáze pracně číst a zpomalovat ji tím? Možností je hned několik.

O aktualizaci se stará klientská aplikace

První možnost se nabízí sama: současně se zápisem změněných dat do operační databáze provádět tento zápis i do datového skladu (viz Obr. 11).



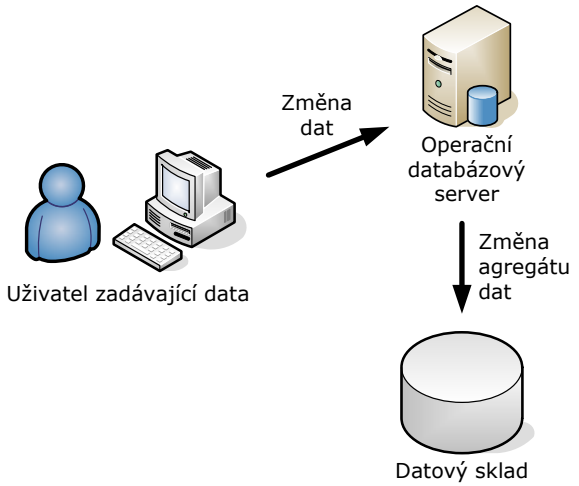
Obr. 11 – Současná změna operačních dat na databázovém serveru i v datovém skladu koncovým uživatelem (klientskou aplikací)

Není ovšem příliš vhodné nechávat tento úkon na aplikaci pro manipulaci s daty. Ta by totiž díky tomu musela přistupovat k datovým agregátům datového skladu, což by sebou přinášelo celou řadu problémů a značně komplikovalo celistvou struk-

turu aplikace. Mezi další podstatné problémy pak patří zbytečné zatěžování komunikačních linek, cca dvojnásobně dlouhá doba ukládání změn dat v aplikaci, řešení logiky práce s daty určenými primárně pouze pro čtení v aplikačním prostředí a především pak stejné zatížení datového skladu operacemi zápisu jako u předchozího přístupu, které by zdržovaly samotnou analytickou činnost nad těmito daty.

O aktualizaci se stará operační server

O něco lepším přístupem je nechat aktualizaci datového skladu na operačním databázovém serveru (viz Obr. 12).



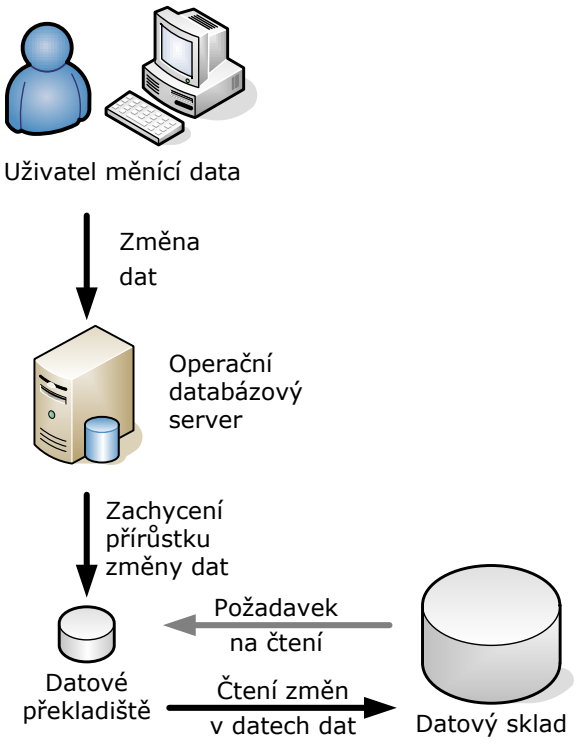
Obr. 12 – Změna agregátů dat v datovém skladu řešená operačním serverem na základě změn prováděných klienty

V tomto případě již klientské aplikace nemusí vůbec řešit existenci datového skladu, neboť ten je od ní touto architekturou zcela odříznut. Aktualizace agregátu dat se provede vždy v okamžiku změny těchto dat na operačním databázovém serveru a je plně v jeho režii. Datový sklad se pak již sám o tyto aktualizace starat nepotřebuje.

Úskalí tohoto postupu jsou ovšem na první pohled zřejmá: Operační server zde kromě zpřístupnění dat uživatelům a uchovávání jejich změn provádí ještě druhotnou činnost – ukládání agregátů těchto změn do datového skladu. Tato činnost pak zbytečně zatěžuje jak operační server a zpomaluje transakce prováděné nad jeho databází, tak zároveň zpomaluje i analytickou činnost prováděnou nad datovým skladem, jelikož provádění změn není přímo v jeho režii a může k němu docházet i v nejméně vhodnou chvíli (při jeho maximálním vytížení). Krom toho, mění-li se data nad nimiž se provádějí výpočty či dolování během této operace, může být jejich výsledek zavádějící.

Zapojení datového překladiště

Řešením, jak učinit obě hlavní jednotky celého procesu (operační datový server a datový sklad) na sobě více nezávislé, je přidáním mezičlánku, který bude zprostředkovávat informace o změněných datech. Tento mezičlánek nezvěme *datové překladiště* (viz Obr. 13).



Obr. 13 – Zapojení datového překladiště

Datové překladiště by měl být samostatný server obsahující jednoduchou databázi zachycující buď absolutní nebo relativní změny v datech. Do překladiště pak operační databázový server pouze zapisuje (vkládá - *insert*) změny ve svých datech. Datový sklad zase naopak pouze tyto změny čte (*read*).

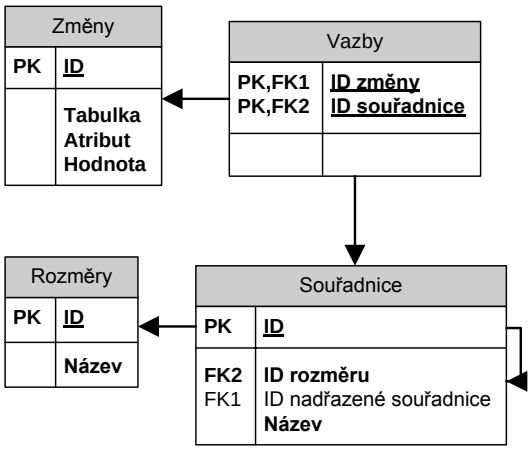
Zapojením datového překladiště dojde k eliminaci hned několika hlavních problémů, jež přinášely předchozí přístupy:

- Činnost, kterou navíc musí vykonávat operační server je redukována na minimum, stačí když pouze při každé změně dat odešle do překladiště informaci o tom, že k operaci došlo a určité podrobnosti o ní (původní hodnotu a novou hodnotu).
- Aktualizace agregovaných dat v datovém skladu je v jeho vlastní režii, může ji tedy provést, kdy potřebuje, například před zahájením analýzy dat a naopak se nestane, aby došlo ke změně dat během ní.
- Při aktualizaci dat není již vůbec zatěžován operační server, nýbrž pouze datové překladiště a to i tak pouze minimálně.
- Změny v datech není potřeba nijak složitě dohledávat, či dokonce provádět jejich kompletní čtení. Veškeré změny jsou přímo přečteny z datového překladiště a triviálními operacemi (sčítáním a odčítáním) promítnuty do agregovaných dat v datovém skladu.

Databáze datového překladiště

Jak již bylo řečeno, je vhodné, aby se datové překladiště fyzicky nacházelo na vlastní výpočetním serveru. Není však vyloučeno jej implementovat na výpočetní jednotce operačního databázového serveru či u datového skladu.

Datové překladiště tedy obsahuje databázi, ve které se zaznamenávají změny provedené v datech operačních serverů. Datový model této databáze je uveden na Obr. 14.



Obr. 14 – Datový model databáze datového překladiště

V tabulce *Rozměry* je definován seznam rozměrů (dimenzí) shodný s tím využívaným datovým skladem. Tabulka *Souřadnice* pak obsahuje seznam všech možných „souřadnic" pro jednotlivé rozměry včetně jejich případného hierarchického rozkladu. Eventuální další vazby mezi rozměry a souřadnicemi zde již řešeny nejsou, jelikož zde jsou irelevantní. V případě vzniku nové souřadnice (třeba zavedením nového produktu na trh), pak lze přímo z operační databáze tuto souřadnici do datového skladu přidat, jelikož ten si ji převezme z datového překladiště. Do tabulky *Změny* pak rovnou zapisuje operační databázový server. Tabulka *Vazby* určuje, které změny se týkají jakých rozměrů v datovém skladu. Týkají-li se díky hierarchickému rozkladu více z nich, je vždy změna přiřazena pouze tomu na nejnižší úrovni (například určitému dnu – datu, nikoli již nadřazeným rozměrům měsíci, čtvrtletí, roku), ty ostatní si ji již převezmou automaticky při aktualizaci datového skladu.

Ať na operačním databázovém serveru dojde již ke kterékoli změně (přidání záznamu – *insert*, úpravě záznamu – *update*, či smazání záznamu – *delete*), jde vždy tuto změnu zcela vystihnout jednoduchým záznamem či záznamy tabulce *Změny* a *Vazby* v databázi datového překladiště.

Příklad obsahu databáze datového překladiště

Příklad obsahu dat ukazují následující tabulky.

Rozměry	
ID	**Název**
1	Produkt
2	Místo
3	Čas

Tabulka 1 – Příklad obsahu tabulky *Rozměry* z databáze v datovém překladišti

Souřadnice			
ID	**ID rozměru**	**ID nadřazené souřadnice**	**Název**
1	1		Nápoje
2	1	1	Minerální voda
3	1	1	Kola
4	1		Zákusky
5	2		Česká republika
6	2	5	Praha
7	3		Rok 2007

Tabulka 2 – Příklad obsahu tabulky *Souřadnice* z databáze v datovém překladišti

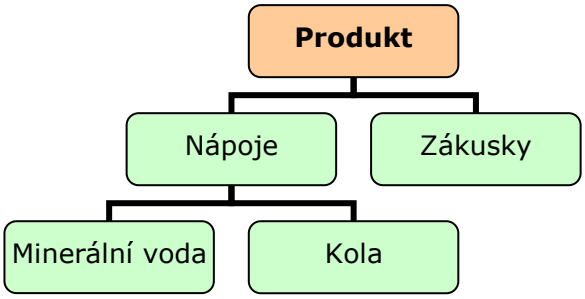
Změny			
ID	**Tabulka**	**Atribut**	**Hodnota**
1	Prodej	Celkem	+200
2	Prodej	Celkem	-50

Tabulka 3 – Příklad obsahu tabulky *Změny* z databáze v datovém překladišti

Vazby	
ID změny	**ID souřadnice**
1	2
1	6
1	7
2	3
2	6
2	7

Tabulka 4 – Příklad obsahu tabulky *Vazby* z databáze v datovém překladišti

Tabulka 1 udává, že budou rozlišovány tři rozměry – *produkt*, *místo* a *čas*. Jak dále definuje Tabulka 2, rozměr *produkt* se dělí na hlavní kategorie *nápoje* a *zákusky*, přičemž pod *nápoje* se řadí *minerální voda* a *kola* (viz Obr. 15). Rozměr *místo* obsahuje *Českou republiku* a pod ní spadá *Praha*. *Časový* údaj pak pro jednoduchost může spadat pouze *rok 2007*.



Obr. 15 – Grafické znázornění hierarchického rozdělení souřadnic pro rozměr produkt v prezentovaném příkladu

Tabulka 3 je tabulkou změn. Do ní v tomto příkladu operační databázový server zaznamenal, že došlo ke změně v tabulce *prodej* a to hodnoty *celkem*. Ta byla nejprve navýšena o *200,-* a poté snížena o *50,-*. K tomu mohlo dojít například úpravou záznamu, tedy nejprve byl vložen záznam s prodejem za 200,- a následně byl tento záznam upraven na hodnotu 150,- tedy došlo ke snížení hodnoty o 50,-, což bylo zaznamenáno v tabulce *Změny*.

Operaci *UPDATE* (změna záznamu) lze do tabulky *Změny* zaznamenat dvojím způsobem. Buď jako dva záznamy, kdy se nejprve odečte celá původní hodnota a poté se přičte celá hodnota nová, nebo jako jeden záznam, kdy se pouze připočte rozdíl těchto dvou hodnot. Operace *DELETE* pak vždy pouze odečte celou svoji „bývalou" hodnotu.

V tomto příkladu je tedy použit relativní přístup ke změnám hodnot. To znamená, že se ve sloupci *Hodnota* v tabulce *Změny* uvádějí přírůstky, o které se daná hodnota změnila. Výsledné hodnoty se pak docílí jejich součtem podle souřadnic a přičtením k aktuální hodnotě v datovém skladu.

Pokud bychom chtěli použít absolutní přístup ke změnám hodnot, muselo by se zároveň v tabulce změn uchovávat ID záznamu na operačním serveru a pak by platila vždy ta nejposlednější hodnota (s nejvyšším ID). To by však bylo možné použít pouze v případě, že by se v datovém skladu uchovávali veškeré hodnoty zvlášť, nikoli pouze agregovaně,

což je celkem výjimečný případ pojetí datového skladu.

Tabulka 4 určuje, do kterých souřadnic jednotlivých rozměrů má být změna hodnoty v datovém skladu přičtena. Přitom je samozřejmě třeba, aby se změna promítla i do nadřazených souřadnic. Tedy konkrétně první řádek tabulky určuje, že hodnota *+200* bude přičtena do sumy prodeje *minerální vody* a zároveň do sumy prodeje *nápojů* (viz Tabulka 2, druhý řádek odkazuje na první přes *ID nadřazené souřadnice*). Dále se pak tato hodnota přičte do sumy prodeje pro *Prahu* a *Českou republiku* a také pro *rok 2007*.

Průběh aktualizace

Průběh aktualizace agregovaných dat v datovém skladu z tabulky změn na datovém překladišti může narazit na řadu překážek. V následujících podkapitolách se pokusíme podrobněji nastínit řešení hlavních z nich.

Triggry

Nejsnadnějším způsobem, jak ze strany operačního serveru zajistit odesílání změn do datového překladiště je využít triggry. Trigger je určitý kód (algoritmus) navázaný k jedné databázové tabulce, který se provede, dojde-li k některé ze zvolených operací (*insert*, *update*, *delete*) na libovolném z jejich záznamů a to buď před vykonáním této změn, nebo až po ní.

Pro potřeby odesílání změn bude nejvhodnější nadefinovat každé sledované tabulce triggry reagující na ukončení každé z těchto tří možných operací. Kód triggru pak může vyvolat jednotně vytvořenou funkci (třeba ve formě uložené procedury či UDF – uživatelsky definované funkce), která se postará o zapsání změny do datového překladiště.

Transakční zpracování

Změny, které triggry provedou v databázi spadají pod tutéž transakci, která zaštiťuje změnu dat, jež tento trigger vyvolala. Ovšem zde jde o dvě oddělené databáze – operační databázi a datové překladiště, přičemž každá z nich má své vlastní transakce.

Během jedné transakce na operačním serveru může dojít k mnohonásobnému spuštění různých triggerů, z nichž každá vyvolá požadavek na zapsání změny do datového překladiště. Pokud by však po každém vložení změny došlo k odpojení (transakce v databázi datového překladiště byla úspěšně zakončena *commit*), mohlo by dojít k tomu, že po uložení několika těchto změn bude transakce na operační databázi zrušena (*rollback*) a veškeré v ní změněné hodnoty „vráceny" do původního stavu, ovšem pouze v operační databázi. Změny uložené v datovém překladišti by tam pak již zůstaly a při aktualizaci z datového skladu způsobily závratné škody na jeho agregovaných datech.

Jak tento problém vyřešit? Možností je hned několik:

- Seznam změn, které se mají uložit do datového překladiště neodesílat hned po jejich vyvolání, ale pouze je lokálně schraňovat třeba do nějaké cache tabulky a odeslat je všechny naráz až při *commitu* transakce.
- Změny do datového překladiště ukládat s určitým příznakem, že jsou součástí zatím stále ještě běžící transakce a tento příznak při jejím *commitu* zrušit. To lze například příznakem u každého záznamu zvlášť, či u nich ukládat pouze ID transakce a v externí tabulce pak evidovat její stav.
- Řešit celou problematiku distribuovaným přístupem, který umožňuje běh transakcí napříč více databázemi.
- Implementovat datové překladiště v rámci operační databáze.

Nelze opomenout, že je také nezbytnou podmínkou, aby případné nezdaření se zapsání změn dat do datového překladiště zapříčinilo i zrušení transakce, která je na operačním serveru vykonala.

Časové značky a čištění datového překladiště

Je třeba taktéž zabezpečit, aby si datový sklad z datového překladiště nepřečetl a nezapočítal tatáž data dvakrát či snad dokonce vícekrát. Každá změna musí být totiž přečtena a započítána právě jednou. Toho lze opět docílit mnoha způsoby.

Jednou možností je, aby datový sklad vše co přečte ihned po té rovnou také smazal v rámci téže transakce. Další alternativou je, že si datové překladiště bude svá data mazat samo. K tomu potřebuje seznam ID záznamů změn, případně alespoň jeho rozsah či maximum, které bylo datovým skaldem při poslední kontrole přečteno. Bude-li tento údaj v datovém překladišti uchován, může na něj brát i ohled případná další aktualizace, jež by předstihla mazání. V potaz by se braly pak pouze záznamy novější (s větším ID) než poslední při minulé aktualizaci přečtený. Naopak při mazání by byly odstraněny pouze starší záznamy (mající ID menší nebo rovno tomu maximálnímu přečetnému při poslední aktualizaci).

Perioda aktualizace

Jednou ze zmiňovaných nevýhod periodického načítání dat do datového skladu bez využití datového překladiště byla neaktuálnost dat v datovém kladu. Nyní, díky faktu, že k datovému překladišti přistupují obě strany pouze s triviálními operacemi a navzájem se nezdržují, je možné data aktualizovat častěji.

Lze tedy nastavit periodu aktualizace velmi nízkou (hodiny, minuty...) nebo pomocí vyvolávání událostí (events) na datovém překladišti vždy informovat datový sklad, že jsou v něm připravena nová data a ten už by si v reakci na toto oznámení data aktualizoval, s přihlédnutím na vlastní vytíženost. Určitou možností také je, data vždy aktualizovat, až když je to třeba, tedy před vykonáváním nějakého analytického výpočtu či dolování v datech. Možné je i nechat tyto aktualizace na uživateli (analytikovi), který by je mohl vyvolávat ručně až dle jeho uvážení a potřeby.

Závěr

Analytické dolování znalostí v reálném čase je velmi užitečnou technologií, jak z velkých databází snadno a rychle dolovat informace a dokonce i znalosti, které mohou být nejen nápomocny při důležitých rozhodováních, ale zároveň mohou odhalovat i nové dosud neobjevené teorie. Mechanizmy těchto dolovacích technik jsou neustále zdokonalovány, vylepšovány, dělány efektivnějšími, aby zvládaly práci se stále se zvyšujícími objemy dat a uživatelsky přívětivějšími, aby je mohli obsluhovat nejen jejich tvůrci, ale i analytici neznalí programování, manageři firem či lékaři.

V tomto článku byly uceleně od základů představeny hlavní technologie pro analytické zpracování dat v reálném čase (OLAP) a dolování v datech (data mining). Datovými strukturami, nad nimiž tyto technologie pracují, jsou datové sklady a vícerozměrné datové kostky. Integrací obou technologií (OLAP a dolování v datech) vznikla nová technologie pro analytické dolování znalostí v reálném čase (OLAM). Mezi softwarová řešení tuto technologii přímo využívající patří DBMiner a nepřímo i vylepšená aplikace Informetrie. V závěru článku byla prezentována teorie, pomocí které je možné řešit jedno z úskalí celé technologie OLAM – plnění datových skladů a to bez zátěže operačních serverů a datového skladu a přitom tak, aby data v datovém skladu byla vždy co nejaktuálnější.

Technologie umožňující získávání znalostí z dat, mají tedy nemalý potenciál. Teoretická řešení nových přístupů těchto technologií ještě zcela určitě zdaleka nedosáhla svého vrcholu, nicméně jejich postupný vývoj je nadějným příslibem do budoucna.

Literatura

[1] Connolly, T.M.; Begg, C.E.: *Database systems: a practical approach to design, implementation, and management*, 4th edition, Addison-Wesley, UK, 2005, ISBN 0-321-21025-5.

[2] *DBMiner 2.0 (Entrprise) User Manual*, DBMiner Technology Inc., dbminer.com, 2003.

[3] Han, J.: *Towards On-Line Analytical Mining in Large Databases*, ACM Press New York, NY, USA, 1998.

[4] Han, J.; Chiang, J.Y.; Chee, S.; Chen, J.; Chen, Q.; Cheng, S.; Gong, W.; Kamber, M.; Koperski, K.; Liu, G.; Lu, Y.; Stefanovic, N.; Winstone, L.; Xia, B.B.; Zaiane, O.R.; Zhang, S.; Zhu, H.: *DBMiner: a system for data mining in relational databases and data warehouses*, IBM Press, Canada, 1997.

[5] Lacko, L.: *Databáze: datové sklady, OLAP a dolování dat s příklady v Microsoft SQL Serveru a Oracle*, Computer Press, Praha, 2003, ISBN 80-7226-969-0.

[6] Lacko, L.: *Oracle – správa, programování, a použití databázového systému*, Computer Press, Praha, 2002, ISBN 80-7226-699-3.

[7] Niemi, T.; Hirvonen, L.; Järvelin, K.: *Multidimensional Data Model and Query Language for Informetrics*. In: Journal of the American Society for Information Science and Technology, 54, č.10, s. 939-951, 2003.

[8] Pinto, H.; Han, J.; Pei, J.; Wang, K., Chen, Q.; Dayal, U.: *Multi-dimensional sequential pattern mining*, ACM Press, New York, NY, USA, 2001, ISBN:1-58113-436-3.

[9] Pokorný, J.: *OLAM = skladování dat + OLAP + dolování dat*, UK, Matematicko-fyzikální fakulta, Sborník semináře Moderní databáze, Beroun, 1999.

[10] Shoshani, A.: OLAP and statistical databases: similarities and differences, ACM Press, New York, NY, USA, 1997, ISBN:0-89791-910-6.