VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Sloupcově orientované databázové systémy

Column-Oriented Database Management Systems

2014 Bc. Martin Parma

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Parma**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Sloupcově orientované databázové systémy
Column-Oriented Database Management Systems

Zásady pro vypracování:

Sloupcově orientovaný model je jedním z nových typů fyzického modelu databázových systémů. Využití tohoto modelu je výhodné u některých typů databázových aplikací a dotazů nad specifickými daty.

1. Nastudujte vlastnosti sloupcově orientovaných databázových systémů.
2. Navrhněte a naimplementujte sloupcově orientovaný fyzický model.
3. Proveďte experimenty a porovnejte implementaci s ostatními implementacemi sloupcově i řádkově orientovaných databázových systémů.
4. Výsledky experimentů vyhodnoťte.

Seznam doporučené odborné literatury:

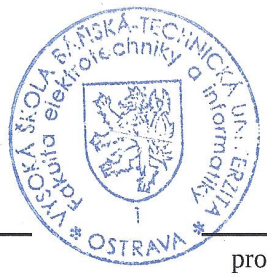
Mike Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran, and Stan Zdonik. 2005. C-store: a column-oriented DBMS. In Proceedings of the 31st international conference on Very large data bases (VLDB '05). VLDB Endowment 553-564.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Michal Krátký, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2014



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2014 .....

Rád bych na tomto místě poděkoval vedoucímu práce doc. Ing. Michalovi Krátkému Ph.D. za cenné připomínky, pomoc a rady během vedení této práce a také Mgr. Jiřímu Tejklovi a Ing. Ondřeji Horákovi za konzultace a praktické rady z praxe.

Abstrakt

Tato práce se zabývá ukládáním dat v sloupcově orientovaných databázových systémech, jejich srovnáním s řádkovými databázovými systémy. V textu je věnovaná pozornost technologiím ukládání dat, popisu implementace vlastního databázového systému a srovnávacím testům. Je zmíněna oblast, ve které je lepší použít vertikální ukládání dat; součástí práce je také vlastní databázový systém pro sloupcové ukládání dat. Následuje srovnání obou systémů, jejich vyhodnocení a doporučení nejlepšího řešení.

Klíčová slova: relační databázové systémy, sloupcově orientované databázové systémy, Business Intelligence, datový sklad, ukládání dat, technologie, uživatel, náklady, metoda, ad-hoc dotazování, akcelerátor, sloupcově orientovaná databáze (CRDB)

Abstract

This work is dealing with data storage in column-oriented database systems, and their comparison with relational database systems.The text concentrates on data storage technologies, description of implementation of its own database system and comparative tests. In the work, I've also included the area in which it is preferred to use vertical data storage; another part of the work is its own database for column-oriented data storage. It is followed by a comparison of the two systems, their evaluation and recommendation of the best solution.

Keywords: Relational Database Systems, Column-Oriented Database Management Systems, Business Intelligence, Data Warehouse, data storage, technology, user, costs, method, ad-hoc query, accelerator, Column Based Relational Database (CRDB)

Seznam použitých zkratek a symbolů

ADO.NET	–	Microsoft ActiveX Data Objects .NET, Představuje množinu tříd, které nabízejí služby pro přístup k datům
ADT	–	Datový typ, který vznikne kombinací základních datových typů
BI	–	Business intelligence jsou dovednosti, znalosti, technologie, aplikace, kvalita, rizika, bezpečnostní otázky a postupy používané v podnikání pro získání lepšího pochopení chování na trhu a obchodních souvislostech
CMP	–	Compare index, Index obsahující výsledky porovnání hodnot různých sloupců
CRM	–	Customer relationship management, Je považován za databázovou technologií podporovaný proces shromažď'ování, zpracování a využití informací o zákaznících firmy
DBMS	–	Database management system, Systém řízení báze dat
DM	–	Data mining, Dolování dat
DOLAP	–	Desktop Online Analytical Processing, Technologie uložení dat v databázi
DSS	–	Decision Support Systems, Systémy pro podporu rozhodování
DTS	–	Data Transformation Services, Sada nástrojů pro automatizaci extraktu, transformace a načítání provozních nebo z databáze
DW	–	Data Warehouse, Datové sklady
ERP	–	Enterprise Resource Planning, Informační systém integrující činnosti podniku (výrobu, logistiku, distribuci, správu majetku, prodej, fakturaci a účetnictví)
ETL	–	Extraction, Transformation, Loading, zkratka slov extrakce, transformace, načítání
FP	–	Fast-Projection index, Index nejvíce demonstrující možnosti vertikálního ukládání dat
HANA	–	High-performance ANalytic Appliance software, Software firmy SAP dodávaný současně s hardware partnerů
HG	–	High-Group index, Index pro data s vysokou kardinalitou, pro které se často provádí grupování pomocí GROUP BY
HNG	–	High-non-Group index, Index navržený pro sloupce s velmi vysokou kardinalitou, kde jsou hodnoty rozkládány na menší části a ty ukládány do samostatných bitmap

HOLAP	–	Hybrid Online Analytical Processing, Technologie uložení dat v databázi kombinující ROLAP a MOLAP
JDBC	–	Java Database Connectivity, Univerzální aplikační rozhraní pro přístup k relačním databázím
JOIN	–	Join index, Index definovaný nad sloupci více tabulek současně
LF	–	Low-Fast index, Index pro sloupce, které mají nízkou nebo velmi nízkou kardinalitu
MOLAP	–	Multidimensional Online Analytical Processing, Technologie uložení dat v databázi
ODBC	–	Open Database Connectivity, Standardizované softwarové API pro přístup k databázovým systémům (DBMS)
OLAP	–	Online Analytical Processing, Technologie uložení dat v databázi
OLTP	–	Online Transaction Processing, Technologie uložení dat v databázi
ORDBMS	–	Object-Relational Database Management System, Objektově-relační databázový systém
QBE	–	Query by Example, Databázový dotazovací jazyk pro relační databáze
RDBMS	–	Relational Database Management System, Relační databázový systém
RDF	–	Resource Description Framework, Základ sémantického webu
ROLAP	–	Relational Online Analytical Processing,Technologie uložení dat v databázi
RTOLAP	–	Real Time Online Analytical Processing,Technologie uložení dat v databázi
SQL	–	Structured Query Language, Strukturovaný dotazovací jazyk pro práci s daty
SŘBD	–	Systém řízení báze dat
SSMS	–	Microsoft SQL Server Management Studio
TCO	–	Total Cost of Ownership, Ekonomická analýza celkových nákladů na vybudování datového skladu
WD	–	Word index, Speciální index pro indexování jednotlivých slov v řetězcích
WOLAP	–	Web-based Online Analytical Processing,Technologie uložení dat v databázi

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato práce se zabývá ukládáním dat v databázových systémech a srovnáním řešení, které jsou na trhu dostupná. V dnešní době máme v této oblasti na výběr mnoho komerčních ale také nekomerčních řešení, bez ohledu na způsob ukládání dat. Tento způsob rozlišení systémů nebude pro naše zkoumání stěžejní; v práci se zaměříme hlavně na způsob ukládání dat, přesto při popisu sloupcově orientovaných databázových systémů krátce představíme zástupce dostupných řešení na trhu, od placených systémů až po neplacená řešení.

Práce bude věnována vertikálnímu ukládání dat v tzv. sloupcově orientovaných databázových systémech. V textu bude představena myšlenka ukládání dat do sloupců, srovnání této metody s relačními databázovými systémy, které jsou v dnešní době nejrozšířenějším řešením pro ukládání informací. Pro srovnání také představíme i stále se rozšiřující řešení tzv. In-Memory Database System, které se využívají u mnohých projektů pro rychlé zpracování většího množství dat.

Při popisu vertikálního ukládání dat budou uvedeny příklady zdrojových dat a zkoumání, pro které je vhodné použití sloupcově orientovaných databázových systémů a představíme si zástupce komerčního řešení.

V textu bude také představen popis řešení vlastní implementace sloupcově orientovaného databázového systému nad existujícím řešením pro relační databáze, představíme si důležité části kódu a řešení otestujeme a porovnáme se stávajícím řešením pro řádkové databáze.

Práce je členěna do sedmi kapitol včetně této první, tedy úvodu. V kapitole 2 jsou představeny technologie pro ukládání dat. V následujících podkapitolách je popsáno datové tržiště a databázový sklad, které jsou určeny primárně pro analýzy dat v rámci Business Intelligence. Bezprostředně za těmito podkapitolami následuje popis relačního modelu, se kterým přímo souvisí také relační databázový systém, jehož součástí je i objektově–relační databáze.

Další podkapitola je věnována In–Memory databázím a cloud computingu. Jsou zde představeni „největší hráči na trhu“ a jedna novější informace, ohledně použití In-Memory databází.
Poslední podkapitola této sekce je věnována krátkému úvodu do sloupcově orientovaných databází.

Kapitola 3 je celá věnována sloupcově orientovanému databázovému systému. Na jejím úvodu je popsán vertikální model a popis rozdílu v ukládání zdrojových dat. Následuje představení speciálních indexů, používaných v sloupcových databázových systémech. V závěru kapitoly je uvedeno, pro jaké aplikace by mělo být nasazení těchto

systémů vhodné a pro které nikoliv.

Kapitola 4 popisuje vlastní implementaci sloupcově orientovaného databázového systému. V úvodu kapitoly jsou zopakovány základní pojmy, které jsou při popisu dále použity.
Pro srovnání je představeno řešení řádkového přístupu, jež je součástí projektu databázového systému QuickDB[1], které bylo využito při implementaci sloupcového řešení.
Následuje návrh fyzického modelu sloupcově orientované databázového systému a popis implementace. Zde jsou představeny některé z důležitých částí implementace.

V páté kapitole nechybí představení testů, které byly použity pro srovnání systémů. Představíme si zde výsledky těchto testů (počet vyhovujících záznamů), dobu vyhodnocení dotazu a počet diskových přístupů.
V závěru kapitoly jsou vyhodnoceny testy a řešení. Porovnáme zde komerční řešení s vlastní implementací sloupcového a řádkového pojetí.

V šesté kapitole je srovnání zástupců komerčních řešení pro řádkový a sloupcový databázový systém. V první podkapitole je představen sloupcový systém HP Vertica[2], jelikož nepatří mezi standardně používané databázové systémy. Microsoft SQL Server[3] v práci popisován není, jelikož se jedná v dnešní době o standard.
Následuje popis konfigurace testovacího prostředí, nahrání testovacích dat a provedení testů.
Na konci kapitoly jsou testy zástupců komerčních řešení vyhodnoceny.

Závěr práce se bude věnovat shrnutí získaných informací a poznatkům z provedených testů.

[1] http://db.cs.vsb.cz/SubPages/Projects/Projects.aspx
[2] http://www.vertica.com/
[3] https://www.microsoft.com/en-us/server-cloud/products/sql-server/

2 Ukládání dat

Informace, kterých kolem nás proudí tolik, že je všechny nedokážeme ani zachytit, jsou hybnou silou naší současné společnosti. V oblasti informačních technologiích narůstala společně s jejich vývojem také poptávka po zpracování dat.

Každý z nás se v soukromém či pracovním životě setkává s potřebou ukládat různé informace (data). S těmito daty má potřebu nadále pracovat, třídit a zkoumat je. V době před rozšířením počítačů mezi veřejnost se jednalo nejspíše o ručně psané zápisky, případně ruční vyplnění předpřipravených formulářů. Práce s daty byla náročná a jejich analýza zabrala spoustu času. Také vyhledávání v těchto datech bylo složité a kromě použití kartoték (které ovšem musely být řádně řazeny), byl proces pouhého nalezení správných dat velmi zdlouhavý.

Databázový systém

Databázový systém je programový systém pro efektivní ukládání, modifikaci a výběr velkého množství perzistentních údajů. Takový systém se skládá ze dvou částí:

- Databáze,
- Systém řízení báze dat (SŘBD či DBMS).

Databázové systémy uchovávají a zprostředkovávají přístup k datům. V současnosti jsou nejrozšířenější relační databázové systémy, ale své místo si již získávají také sloupcově orientované databázové systémy a „in-memory“ databáze.

Databáze

Pojmem databáze rozumíme kolekci logicky souvisejících sdílených dat s popisem datové struktury, organizovaných pro optimální manipulaci s perzistentními daty a získávání informací pro informační systémy [1].

Systém řízení báze dat

Jedná se o programový systém, který je schopný efektivně pracovat s velkým množstvím dat, dále je schopný řídit (vkládat, modifikovat a mazat) a definovat strukturu těchto perzistentních dat [1].

Charakteristické vlastnosti SŘBD:

- podpora definice datových modelů (relační, logický, objektový),
- integrita dat; například nepovolením vložení duplicitního řádku s unikátním klíčem nebo řádku s NULL hodnotami u sloupců, které NULL být nesmějí,

- autentizace uživatelů a jejich autorizace k operacím nad daty (definice typu příkazů, které je oprávněn spouštět),
- robustnost a zotavitelnost po chybách bez ztráty dat,
- uložené procedury,
- triggery,
- správa transakcí, atomicita jednotlivých příkazů,
- správa klíčů: interně implementované indexování, dodržování unikátnosti hodnot ve sloupcích, nad kterými je definován unikátní nebo primární klíč, implementace cizích klíčů,
- profilování, statistické informace o běhu dotazů, procesů, přístupu uživatelů atd.,
- implementace fulltextového vyhledávání pro fulltextové klíče,
- využití některého jazyka vyšší úrovně pro manipulaci a definici dat (SQL, QBE, ...) a vytvoření komunikačního kanálu mezi uživatelem nebo skriptem a SŘBD v tomto jazyku,
- kanály pro hlášení zpráv po úspěšně vykonaných dotazech, chybových hlášek, varování,
- pokročilé funkce jako např. Common Table Expressions, zpožděné zápisy, a jiné.

Známá SŘBD řešení

V oblasti SŘBD je na trhu mnoho řešení, ze kterých můžeme vybírat dle náročnosti projektu, finančních a dalších preferencí.

Patří k nim zejména:

- Oracle[4],
- Microsoft SQL Server[5],
- DB2[6],
- MySQL[7],

[4] http://www.oracle.com/cz/products/database/overview/index.html
[5] http://www.microsoft.com/cs-cz/server-cloud/products/sql-server/
[6] http://www-01.ibm.com/software/data/db2/
[7] http://www.mysql.com/

- PostgreSQL[8],
- Microsoft Access[9],
- SQLite[10].

2.1 Technologie uložení dat - přístup ke zpracování dat

V oblasti technologií pro uložení a přístup k zpracování dat se běžně používají dva způsoby. Tyto způsoby jsou OLAP (Online Analytical Processing) a OLTP (Online Transaction Processing). Základní rozdíl mezi technologiemi OLAP a OLTP je ve způsobu použití – u OLAP se většinou jednorázově nahrají data, nad kterými jsou prováděny složité dotazy. V případě OLTP jsou data průběžně modifikována a v průběhu přidávána, obvykle mnoha uživateli zároveň.

2.1.1 OLAP

Zkratka OLAP je pojmenováním systému pro hromadné zpracování dat. Kompletní přehled tohoto systému nalezneme na internetových stránkách [2]; články z tohoto zdroje sloužily jako inspirace v následujícím textu o OLAP.
V devadesátých letech minulého století byla tato myšlenka a její následná technologická implementace velkým krokem kupředu v oblasti zpracování velkého objemu dat. Zpracování dat metodou OLAP bylo teoreticky popsané již dříve. Tato metoda umožňuje dosáhnout podstatně vyššího výkonu s výrazně menšími náklady, přičemž nemusí dojít ke změně architektury řešení. Mezi známé zástupce tohoto systému zpracování dat patří Hyperion, Cognos, Business Objects, MicroStrategy.

Principem fungování se zabývá článek [6]. Píše se v něm, že se jedná o technologii, která umožňuje uspořádání rozsáhlých obchodních databází a podporuje analytické nástroje. Databáze OLAP jsou rozděleny do jedné nebo více krychlí a jednotlivé krychle uspořádány a navrženy správcem krychlí tak, aby vyhovovaly způsobu načítání a analýzy dat a tím usnadňovaly vytváření potřebných kontingenčních tabulek a grafů.

Technologie OLAP nám dává do ruky nástroj, který umožní přejít od analýzy dat v jednom rozměru k analýze vícerozměrné – multidimenzionální. Umožňuje uspořádat velké objemy dat tak, aby byla data přístupná a srozumitelná uživatelům zabývajícím se analýzou obchodních trendů a výsledků (Business Intelligence).

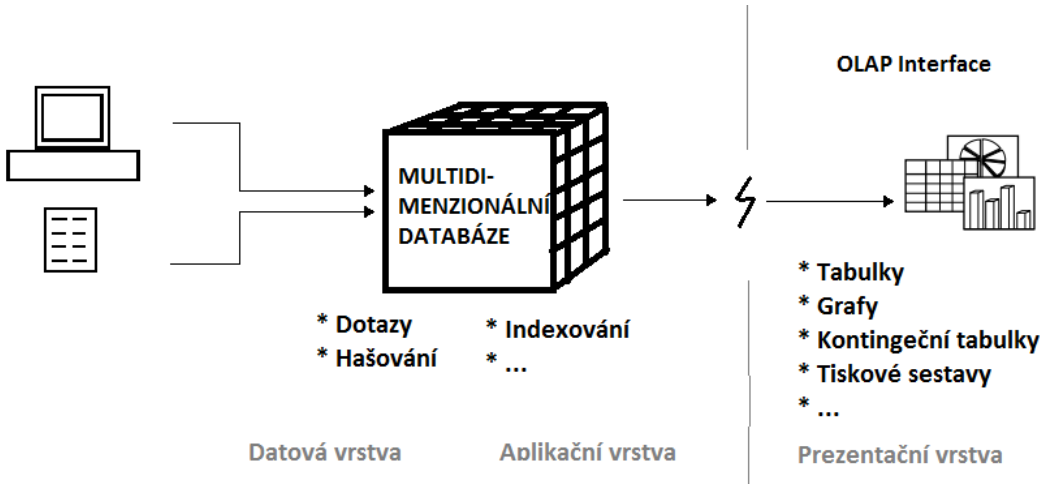
Systémy OLAP dělíme dle následující taxonomie:

- **MOLAP** – je tzv. multidimenzionální online analytické zpracování, jež je ve své podstatě standardní OLAP technologií. U metody MOLAP se všechny potřebné údaje z kostky

[8] http://www.postgresql.org/
[9] http://office.microsoft.com/cs-cz/access/
[10] http://www.sqlite.org/

ukládají do vyhrazené multidimenzionální databáze. Relační data uložená v tabulkách dimenzí a v tabulkách faktů jsou zapsána do optimalizované multidimenzionální databáze. K těmto podrobným datům z kostky jsou uloženy také všechny agregace. Jedná se o optimalizovanou architekturu OLAP pro zpracování dotazu ve službách OLAP a poskytuje nejlepší výkon ze všech metod ukládání dat. Zajímavý je i z hlediska úspory úložného prostoru, protože zahrnuje i specializované kompresní techniky pro uložení dat [4].

Znázornění architektury MOLAP je popsáno na obrázku 1. Z tohoto obrázku je patrné, že všechny potřebné údaje z kostky uložené do multidimenzionální databáze zastupují zároveň datovou (databázovou) vrstvu a aplikační vrstvu z třívrstvé architektury informačních systémů. Prezentační vrstvu tvoří samotné OLAP rozhraní.

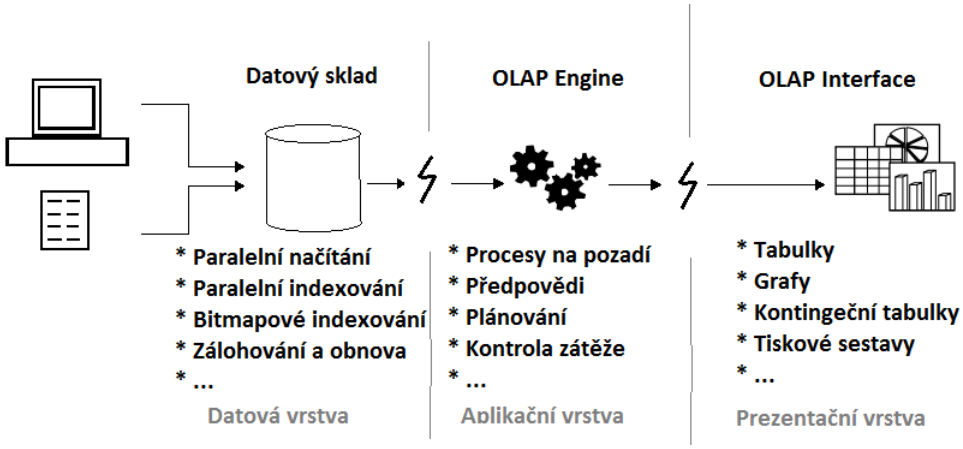


Obrázek 1: Architektura MOLAP

- **ROLAP** – je obecně více škálovatelný. Vyžaduje uložení všech podrobných údajů, ale i agregací v relační databázi. To znamená, že všechna detailní data z kostky, která jsou v tabulkách dimenzí a tabulkách faktů, zůstávají ponechána v jejich přirozené relační databázi. Data se nepřesunují, při ukládání agregací se vytvoří sumarizační tabulky, do kterých budou relační data ukládána službami OLAP jednoduchým příkazem „INSERT INTO ... “. Služby OLAP vytvoří všechny tabulky a indexy samočinně, detailní údaje v kostce zůstávají nezměněny. Ve srovnání s MOLAP nepřináší ROLAP z pohledu výkonu zlepšení, avšak v praxi se jedná o velmi rozšířený způsob ukládání dat [3].

Architektura ROLAP je znázorněna na obrázku 2. Datovou vrstvu zde zastupuje RDBMS (datový sklad), aplikační vrstvu právě ROLAP engine a prezentační vrstvu zastupuje opět OLAP rozhraní.

- **HOLAP** – jedná se o systém ukládání, který je kombinací MOLAP a ROLAP. Jde o ukládání všech detailních informací z kostek do relační databáze. Všechny agregované



Obrázek 2: Architektura ROLAP

údaje jsou ukládány do multidimenzionální databáze. Tato metoda kombinuje to nejlepší z obou předchozích technologií, tedy výkonnost MOLAP a rozšiřitelnost ROLAP [3].

Další méně se vyskytující typy
- **WOLAP** – Web-based OLAP
- **DOLAP** – Desktop OLAP
- **RTOLAP** – Real-Time OLAP

2.1.2 OLTP

Zkratka OLTP (Online Transaction Processing) je pojmenování systému, který slouží pro transakční zpracování dat (pro zápis a získávání). Systémy OLTP dávají odpověď na zadaný požadavek online (tedy ihned).
Tyto principy jsou využity v širokém spektru oblastí, ze kterých můžeme jmenovat bankovnictví, e-obchody, letectví a mnoho dalších [11].

2.2 Datové tržiště

Datové tržiště (angl. „Data Mart“) je zdrojem dat pro analýzy pomocí nástrojů BI (Business Intelligence). Je to specifická forma tématicky orientovaného datového skladu, určená ke zprostředkování informací pro určité oddělení firmy (např. marketing, finanční oddělení, . . .) [7].

Mnohdy bývá následně datový sklad firmy vytvořen právě sjednocením jednotlivých data martů. Jednotlivé data marty jsou budovány na základě požadavků jednotlivých útvarů společnosti.

Z toho vyplývá:

- potřeba vlastních dat,
- používání vlastních definic pojmů,
- vlastní historie dat,
- vlastní periodicita aktualizace dat.

Jedná se o tzv. dvouvrstvou architekturu (koncept Ralfa Kimballa), kterou volíme tehdy, jestliže je potřeba zvýhodnit určité oddělení či pobočku a dodat pro ně výstupy datového skladu v relativně krátké době (obecně do několika měsíců).
Nově vznikající datový sklad se buduje postupně, po jednotlivých datových tržištích dle priorit oddělení. Předávané výsledky, vývoj datového skladu, ale také finance na vývoj jsou rozloženy v čase a nepředstavují jednorázovou finanční zátěž [7].

2.3 Datový sklad

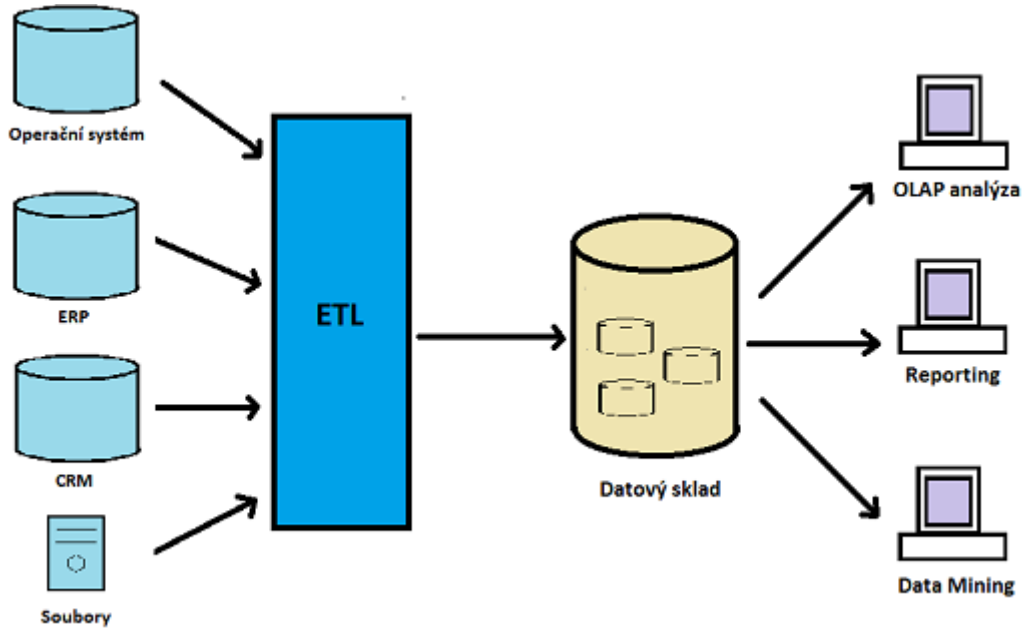
Datový sklad (angl. „Data Warehouse“) je zvláštní typ databáze určený primárně pro analýzy dat v rámci Business Intelligence (oblast analýzy dat sloužící jako podklady pro manažerské rozhodování).

„Datový sklad je podnikový strukturovaný depozitář předmětově (subjektově) orientovaných, vzájemně provázaných, nepodléhající změnám, časově proměnných, historických dat používaný na získávání informací a pro podporu rozhodování. V datovém skladu jsou uložena detailní (atomická) i sumární data“.
Bill Inmon, prezident Data Systems, „otec“ datových skladů.

Data jsou do datových skladů ukládána pomocí tzv. datových pump, které z provozních databází (běžně relační databáze podnikových informačních systémů – CRM, ERP, ...) ukládají do datového skladu, pomocí „ETL“ nástroje (zkratka slov „Extraction“, „Transformation“ a „Loading“). Příkladem ETL v praxi je například Microsoft DTS (Data Transformation Services), který je součástí instalace MS SQL Serveru, nebo Oracle Data Mart Builder [7].
Schéma ukládání dat do datového skladu popisuje i obrázek 3.

2.3.1 Relační databáze vs. datový sklad

Při navrhování relačních databází se snažíme o odstranění redundancí uložených dat, například normalizací či vnitřním provázáním jednotlivých funkčních celků.
V případě datového skladu se snažíme o jasnou vnitřní separaci jednotlivých funkčních celků - tím vznikne struktura, která je čitelnější pro uživatele (business analytika, manažera, ...), ovšem za cenu zvýšení nároků na paměťový prostor. Při popisu struktury datového skladu mluvíme o tzv. multidimenzionální struktuře uložených dat.



Obrázek 3: Schéma fungování datového skladu

Do datových skladů nahráváme data zpravidla ve větších dávkách, tato data pak již obvykle nejsou upravována. U běžných firemních aplikací, kde jsou data uložena v relačních databázích, řešíme specifický okruh úloh nad specifickými daty. V datovém skladu musíme shromáždit informace z různých zdrojů a seskupit je podle logického významu (orientace na subjekt - všechna data z různých zdrojů na jednom místě, týkající se jedné oblasti).

Relační databáze navrhujeme pomocí entitně relačního modelování (ERD), datový sklad vytváříme pomocí dimenzionálního modelování.

Jednotlivé kroky modelování pro relační databáze [8]

- odstranění redundancí (opakování se dat),
- normalizace (což přinese nižší srozumitelnost pro běžné čtení člověkem),
- optimalizace pro zápis a editaci dat.

Jednotlivé kroky pro modelování datového skladu [9]

- optimalizace pro čtení, vyhledávání a složité analýzy dat,
- dosažení srozumitelnosti pro uživatele (dosaženo standardní strukturou – faktová tabulka, dimenze),

- základní přístup – denormalizace, redundance,
- rozšiřitelnost – přidání fakt, dimenzí bez dopadu na aplikace.

Data jsou v datovém skladu uspořádána topologicky. Z logického pohledu jsou členěna do schématu, kdy každé schéma odpovídá jedné zkoumané funkční oblasti. Každé takové schéma obsahuje dva typy tabulek - faktové a dimenzionální. V jedné nebo více faktových tabulkách jsou uložena analyzovaná data (veličiny, které sledujeme, hodnoty které jsou použity k analytickým výpočtům - agregacím, třídění apod.). Většinu velikosti datového skladu zabírají právě faktové tabulky, protože obsahují detailní údaje ze všech zdrojů, což je obecně daleko více, než další potřebné tabulky.
Klíčovým pojmem u faktových tabulek je granularita, která určuje úroveň podrobnosti údajů ve faktové tabulce. Čím nižší je úroveň granularity, tím detailnější jsou data určená k provádění matematických operací.

Dimenze jsou tabulky, které obsahují seznamy hodnot sloužících ke kategorizaci a třídění dat ve faktových tabulkách (atributy, prostřednictvím kterých se „díváme“ na data). Je to vlastně číselník, podle kterého chceme data analyzovat. Pomocí cizích klíčů jsou faktové tabulky spojeny s dimenzemi [7].

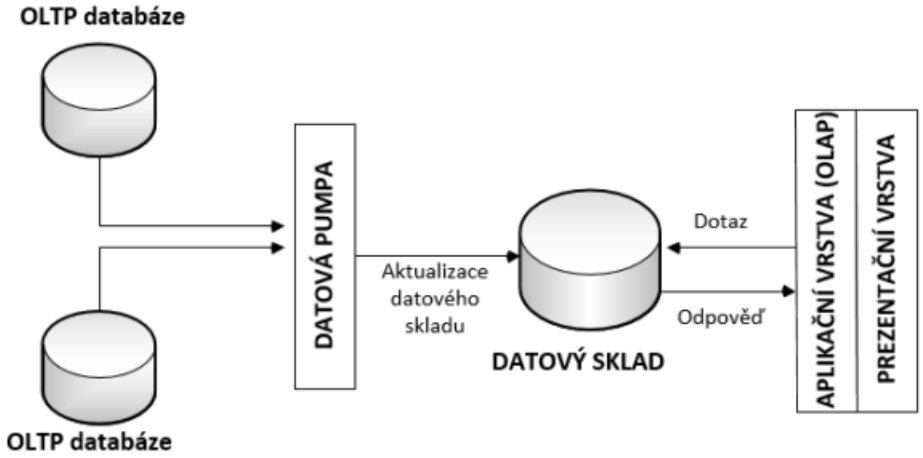
Vlastnosti dimenzí:

1. Vztah mezi faktovou tabulkou a dimenzemi je 1:Nn
2. Dimenze určují úhel pohledu – čas, produkt, zákazník,. . . ,
3. Dimenze udržují hierarchie (vztah 1:N).

Datový sklad je hlavním zdrojem dat pro systémy Business Intelligence. Máme-li vytvořen datový sklad, informace v datovém skladu můžeme dále analyzovat. Pro následnou analýzu se používají například technologie OLAP (základní analýza) nebo techniky „data miningu“ (pokročilejší analýza). Struktura datového skladu je znázorněna na obrázku 4.

Rozdíl mezi OLTP a OLAP

Odlišnost mezi OLAP a OLTP spočívá v tom, že OLTP systémy uchovávají záznamy o jednotlivých transakcích. V dnešní době jsou realizovány obvykle pomocí relační databázové technologie. OLTP uchovává již agregována data, která jsou ukládána do datového skladu. Teprve nad těmito daty se provádí okamžité zpracování analýz pomocí vrstvy OLAP [9].



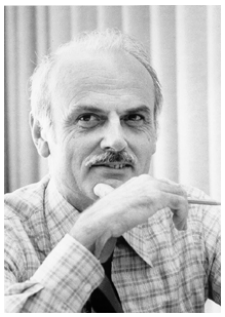
Obrázek 4: Struktura datového skladu

Znak	OLTP	OLAP
Charakteristika	Provozní zpracování	Informační zpracování
Orientace	Transakční	Analytická
Uživatel	Úředník, databázový administrátor	Znalostní pracovník (manažer, analytik)
Funkce	Každodenní operace	Dlouhodobé informační požadavky, podpora rozhodování
Návrh databáze	Entitně-relační základ, aplikačně orientovaný	Hvězda/sněžná vločka, věcná orientace
Data	Současná, zaručeně aktuální	Historická
Sumarizace dat	Základní, vysoce detailní	Shrnutá, kompaktní
Náhled	Detailní	Shrnutý, multidimenzionální
Jednotky práce	Krátké, jednoduché transakce	Komplexní dotazy
Přístup	Číst a zapisovat	Většinou pouze číst
Zaměření	Vkládání dat	Získávání informací
Počet dostupných záznamů	Desítky	Miliony
Počet uživatelů	Tisíce	Stovky
Velikost databáze	100 MB až GB	100 GB až TB
Přednosti	Vysoký výkon vysoká přístupnost	Vysoká flexibilita nezávislost koncového uživatele
Míry hodnocení	Prostupnost transakcí	Prostupnost dotazů a doba odezvy

Tabulka 1: Srovnávací tabulka OLTP vs. OLAP

2.4 Relační databázové systémy

Relační databázové systémy jsou založeny na tzv. relačním modelu uložení dat. Tyto systémy se označují zkratkou RDBMS. Myšlenka relačního modelu uložení dat byla poprvé představena na konferenci pracovníkem firmy IBM panem Edgarem Frankem Coddem 5 již v roce 1970 [10]. Byl to nejslavnější přínos tohoto anglického počítačového vědeckého pracovníka.



Obrázek 5: Edgar Frank „Ted“ Codd

2.4.1 Relační datový model

Data dle relačního modelu jsou ukládána v relacích tvořených n-ticemi [10]. Na těchto relacích jsou definovány operace, které nazýváme relační algebra [10]. Relace je nejčastěji tabulka s n sloupci, mohou to být ale i jiná libovolná data, získaná z řádků a sloupců více tabulek (většinou se jedná o výsledek nějakého dotazu nad tabulkami).

2.4.2 Relační databáze

Relační databáze je kolekcí relací, vztahů mezi relacemi, indexy a dalšími součástmi. Vlastnosti n-tic nazýváme atributy záznamu, a mohou obsahovat jedinou hodnotu z předem dané domény, kterou nelze dále dělit. Jsou reprezentovány jako sloupce tabulky. Doména hodnot, které může daný atribut nabýt, je určena pomocí datového typu při deklaraci relace. Vztahy mezi relacemi jsou vyjádřeny také jako vlastnosti záznamu, kterou odkazují na jinou relaci (nazývané cizí klíč). Vztah typu N:M lze vyjádřit zavedením další relace.

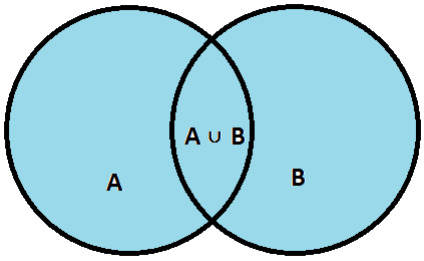
Relační model databáze umožňuje reprezentovat data logickým a konzistentním způsobem. Toho je dosaženo pomocí omezení (constraints). Kandidátní klíč je takový sloupec (nebo skupina sloupců), který jednoznačně identifikuje řádek. Navíc se vyžaduje, aby kandidátní klíč neobsahoval žádný nadbytečný sloupec – tj. odstraněním libovolného sloupce z kandidátního klíče přestane jednoznačně identifikovat řádek. Jeden z kandidátních klíčů je primární klíč. A každá relace by měla mít definovaný právě jeden primární

klíč. Cizí klíč je takový sloupec (nebo skupina sloupců), který je totožný s hodnotou kandidátního (většinou primárního) klíče jiné relace. Cizí klíč v relaci vlastně definuje vztah s řádkem odkazované relace.

Mezi základní operace relační algebry patří projekce (výběr zvolených atributů relace), selekce (výběr těch záznamů relace, které vyhovují zadané podmínce), přejmenování atributu a spojení více relací podle různých kriterií.

Sjednocení množin
Sjednocení dvou množin je taková množina, která obsahuje každý prvek, který se nachází alespoň v jedné ze sjednocovaných množin $A \vee B$ (dle příkladu na obrázku 6).
Sjednocení množin označujeme $A \cup B$.
Pro všechna x platí, že:

$$x \in A \cup B \Leftrightarrow (x \in A \vee x \in B).$$



Obrázek 6: Sjednocení množin A a B

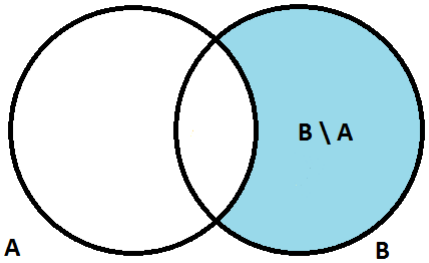
Př: Máme-li množinu A ={1, 3, 5, 7, 9} a množinu B ={2, 4, 6, 8, 10}, je výsledkem sjednocení množin $A \cup B$ = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}.

Rozdíl množin
Rozdílem dvou množin je taková množina, která obsahuje prvky z první množiny B a neobsahuje žádný prvek z množiny A (dle příkladu na obrázku 7).
Rozdíl množiny označujeme $B - A$ nebo $B \backslash A$.
Pro všechna x platí, že:

$$x \in B - A \Leftrightarrow (x \in B \wedge x \notin A).$$

Př: Máme-li množinu A ={1, 3, 5, 7} a množinu B ={1, 4, 5, 6, 7, 10}, je výsledkem rozdílu množin $B - A$ = {4, 6, 10}.

Kartézský součin
Kartézským součinem dvou množin A a B je taková množina, která obsahuje všechny



Obrázek 7: Rozdíl množin B a A

uspořádané dvojice, v nichž je první položka prvkem množiny A a druhá položka je prvkem množiny B.
Kartézský součin obsahuje všechny takové kombinace těchto prvků (příklad kartézského součinu $A \times B$ na obrázku 8).

Př: Máme-li množinu $A = a, b, c$ a množinu $B = 1, 2, 3$, je kartézským součinem $A \times B = (a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3), (c, 1), (c, 2), (c, 3)$.

Kartézský součin dvou množin označujeme $A \times B$.

Pro všechna x a y platí, že:

$$A \times B = \{(x, y) : x \in A \wedge y \in B\}$$

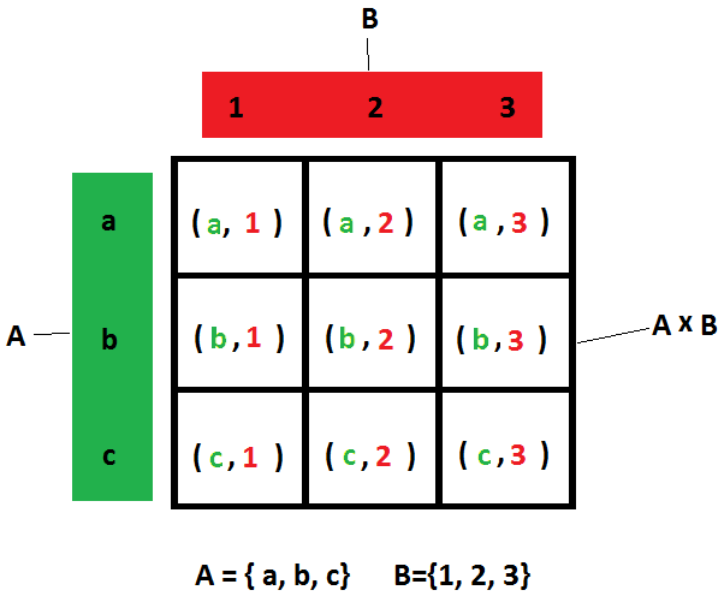
Projekce
Projekce tabulky (relace) R s položkami A na množinu položek B, kde $B \subseteq A$:

- vytvoří tabulku s položkami B a záznamy, které vzniknou z původní tabulky R odstraněním položek $A - B$. Odstraněny jsou i eventuelně opakující se záznamy.
- značení : $R[B]$

Selekce
Selekce tabulky$R$ s položkami A podle logické podmínky Φ:

- vytvoří tabulku s týmiž položkami a ponechá ty záznamy z původní tabulky, které splňují logickou podmínku Φ. Podmínka Φ je logický výraz, jehož formule nad jednoduchými položkami tabulky mají tvar:

$$\Phi = t_1 \Theta t_2$$



Obrázek 8: Kartézský součin množin A a B

kde t_1, t_2 jsou jména položek tabulky, konstanty nebo výrazy
Θ logické operátory $\Theta \in \{<, >, =, \leq, \geq\}$

- značení : $R(\Theta)$

Spojení
Spojení tabulek R a S s položkami A a B:

- vytvoří minimalizovanou tabulku se schématem $A \cup B$ a se záznamy, jejichž projekce na A je z tabulky R a projekce na B je z tabulky S.

- značení : $R * S$

V současnosti je většina databází založena právě na relačním modelu, proto jsou relační databázové systémy prozatím nejrozšířenější variantou ukládání dat. Data jsou ukládána do řádků, kdy jeden řádek tabulky odpovídá jednomu záznamu. K dotazování se používá jazyk SQL (Structured Query Language), který vychází z relační algebry.

2.5 Objektově-relační databáze

Speciálním typem relačních databází jsou tzv. objektově-relační databáze (značíme je zkratkou ORDBMS). ORDBMS je specifikována v rozšíření SQL standardu - SQL3. Tyto relační databáze jsou navíc rozšířené o objektově relační prvky. To přináší nové možnosti, jako definici vlastních datových typů, operátorů, funkcí, uložených procedur nebo triggerů.

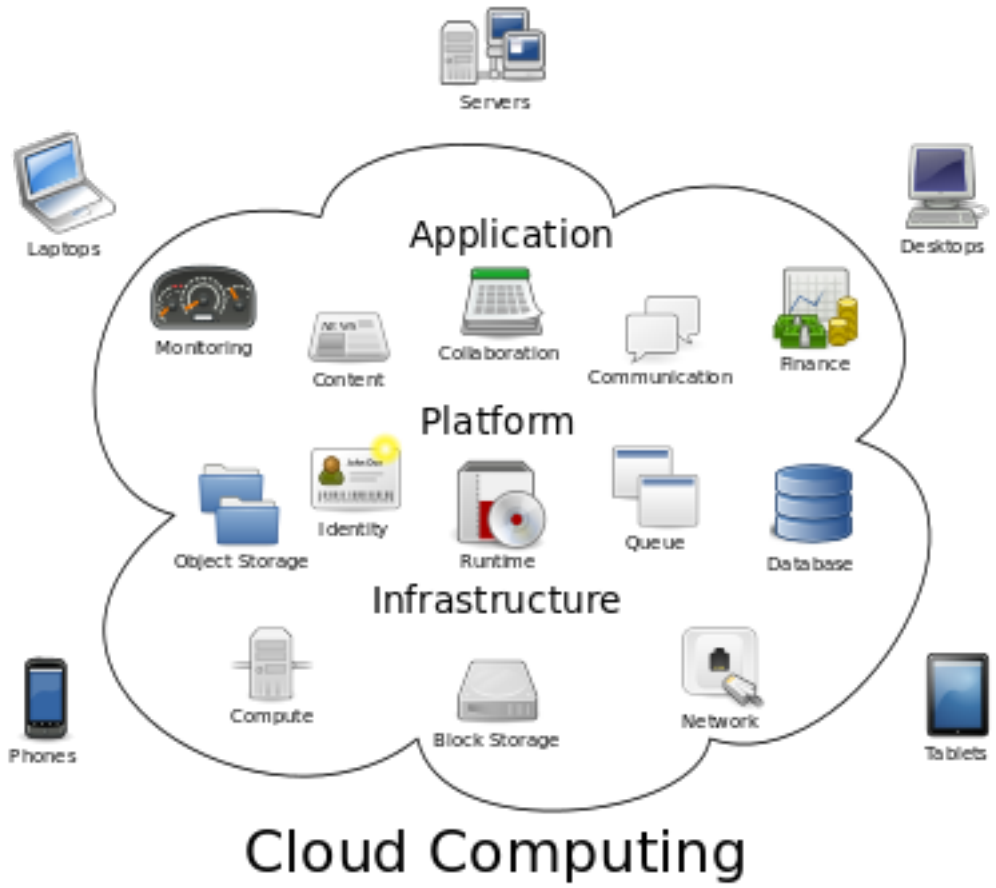
Všechny informace jsou nadále uloženy v tabulkách, ale některé atributy mohou mít bohatší datovou strukturu. Tyto typy atibutů nazýváme abstraktní datové typy (ADT).
ADT je datový typ, který vznikne kombinací základních datových typů. Podpora ADT je pro vývojáře velmi zajímavá, jelikož operace a funkce asociované s novými datovými typy mohou být použity k indexování, ukládání a získávání záznamů na základě obsahu nového datového typu.
ORDBMS jsou nadmnožinou RDBMS, nevyužijeme-li žádné objektové rozšíření, zůstávají nadále ekvivalentní s SQL2 (RDBMS). Z tohoto důvodu má omezenou podporu dědičnosti, polymorfismu, referencí a integrace s programovacím jazykem.

2.6 In-Memory databáze

In-Memory database, tedy databáze uložené v hlavní paměti počítače, patří v současnosti k nejdiskutovanějším tématům spojených se systémy pro řízení báze dat (SŘBD/DBMS). O databázích uložených v hlavní paměti počítače se začalo hodně diskutovat díky Cloud computingu, který díky internetu nabízí jiný (pro nás dříve netradiční) způsob používání počítačových technologií. Lze jej charakterizovat jako poskytování služeb či aplikací uložených na serverech na internetu. Uživatelé k nim mohou přistupovat pomocí webového prohlížeče nebo klienta dané aplikace a používat je prakticky odkudkoliv. Uživatelé neplatí (za předpokladu, že je služba placená) za vlastní software, ale za jeho užití. Nabídka aplikací se pohybuje od kancelářských aplikací, přes systémy pro distribuované výpočty, až po operační systémy provozované v prohlížečích, jako je například eyeOS, Cloud či iCloud [14].
Jestliže u vícevrstvých aplikací mohou být ostatní vrstvy bezstavové, a tudíž lze snadno a rychle zvýšit jejich výkon spuštěním jejich dalších instancí, u databázové vrstvy je pro vytvoření nové instance potřeba duplikovat velké množství dat uchované na discích. Nejde přitom o žádný čistě teoretický problém – u aplikací náročných na zpracování dat se může databázová vrstva cloudu stát snadno úzkým hrdlem brzdícím činnost všech ostatních.

Databázové systémy, které spoléhají na uchování zpracovávaných dat v hlavní paměti místo na disku nejsou trendem, který by se objevil v posledních měsících. Podle komentátorů se v této oblasti od roku 2011 doslova mění „pravidla hry“, a to díky společnosti SAP, pro niž byl rok 2011 ve znamení jejího softwaru HANA (High-performance ANalytic Appliance software). Jde o software, který je klientům dodáván současně s hardwarem partnerů, jako ucelené databázové zařízení. A toto zařízení spoléhá při své práci právě na řešení typu in-memory database.

Jak je uvedeno v článku [14], zástupci výrobce vysvětlují, že jejich přístup je vlastně prostý: Data z IT systémů používaných při podnikání jsou okamžitě ukládána do paměti a analytické výstupy jsou prostřednictvím různých pohledů dle potřeby předkládány uživatelům v reálném čase. Srdcem systému je přitom integrovaná databázově – výpočetní vrstva, která dovoluje zpracování velkého množství dat v hlavní paměti systému a umožňuje poskytnout tedy požadované výsledky velmi rychle. Výrobce uvádí, že systém



Obrázek 9: Znázornění cloud computingu [15]

umožňuje projít jednomu procesorovému jádru dva miliony záznamů za milisekundu a provést deset milionů komplexních seskupení (agregací) za sekundu.

Na reálných datech pak představitelé SAP ukázali provádění komplexních databázových dotazů na databázi s více než 450 miliardami záznamů (komprimovanými do méně než tři terabytů fyzické paměti), jejichž výsledky systém poskytl v čase v řádech sekund. Autor předchozího článku [14] také zmiňuje, že všechny operace jsou prováděny v paměti, jen logování ukládá svůj výstup na vysoce výkonné SSD disky. Systém je podle SAP škálovatelný na systémy s více než tisícovkou procesorových jader [16].

Své in-memory databázové systémy již déle nabízejí i další velcí databázoví „hráči“, jako jsou Oracle a IBM. První jmenovaná firma v roce 2005 koupila firmu TimesTen a nyní její řešení nabízí jak samostatně, tak jako součást svých standardních SŘBD. Po-

dobně tento problém vyřešila i IBM, která v roce 2008 koupila firmu SolidDB.

In-memory database nacházejí využití především v odvětví telekomunikací, e-comerce a obecně v celém finančním sektoru. V této oblasti se pracuje s velkými objemy dat a tudíž jsou kladeny vysoké nároky na rychlost výstupů. Obecné označení In-Memory Database lze aplikovat – a také se tak děje – i na systémy, kde je sice celá databáze v paměti, ale jedná se jen o malý objem dat. Z hlediska technických a finančních nároků však hovoříme pochopitelně o zcela odlišné technologii.

Rok 2013 přinesl do světa in-memory zpracování dat další novinku. Na konferenci OpenWorld 2013 oznámil výkonný šéf společnosti Oracle Larry Ellison dostupnost nové možnosti v databázovém systému Oraclu, díky kterému je možné zpracovávat data v operační paměti. Jedná se o reakci na vydání databáze HANA od SAPu, které jsme se věnovali v předchozím textu.

SAP měl od roku 2011 velikou výhodu v tom, že nabídl právě zpracování dat v paměti, díky čemuž výrazně zrychlil jejich veškeré procesy. I díky tomu se HANA stala nejrychleji rostoucím IT produktem v historii. Za 2 roky nasbírala 1500 zákazníků a v roce 2012 vzrostla o 142 procent. Šéf společnosti SAP Jim Hagemann Snabe tehdy dokonce prohlásil: *„S HANA jsme schopní trh vést, ne ho následovat."*

Oracle proto v roce 2013 představil databázi Oracle 12c, která umožňuje jednoduše přepnout do in-memory módu. Ellison tvrdí, že se díky tomu data dají zpracovat až stokrát rychleji. Oracle vydal prohlášení, že jejich řešení in-memory zvládá odbavit miliardy dotazů za sekundu.Velkou výhodou má být to, že databáze není třeba migrovat a aplikace předělávat. „Jednoduše stačí přepnout možnost na in-memory a vše funguje." Do paměti u 12c přitom není nutné ukládat všechna data, ale například jenom ta aktivní. Zbytek pak může jít také na flash a zálohy na tradiční pevné disky.

Oracle po odkupu firmy Sun Microsystems razí strategii „Applu pro datová centra", v rámci které se snaží kombinovat svůj software a databáze společně s hardwarem. I proto firma společně s in-memory představila také novou serverovou sestavu nazvanou „Big Memory Machine" s označením M6-32. Jde o sestavu za tři miliony dolarů, která zvládá odbavit tříterabytový datový tok za sekundu, má 32 TB operační paměti a 32 nových čipů SPARC M6 vyvinutých Oraclem. Dle šéfa Oraclu se jedná o nejrychlejší databázový stroj na světě [16].

Právě konkurenční boj firem SAP a Oracle by mohl přinést hodně novinek a vylepšení v oblasti in-memory databází a zpracování dat celkově. Můžeme se tak domnívat, že tento konkurenční boj, jež přinesl nový pohled na zpracování dat, bude i nadále „hnacím motorem" v rozšiřování technologií pro zpracování velkého objemu dat.

2.7 SAP HANA v praxi – aplikace SAP Match Insights

Mnozí z nás se řadí mezi fanoušky různých sportů. Někdo se řadí mezi aktivní a svůj oblíbený sport provozuje alespoň na amatérské úrovní, větší skupinu tvoří lidé, kteří jsou pasivními fanoušky některých sportů. Národním sportem číslo jedna je pro Německo bezesporu fotbal, ať už jeho fanoušci patří do první nebo druhé skupiny.

Nový pohled pro trenéry fotbalových týmů představila právě společnost SAP. Bundesligový tým TSG 1899 Hoffenheim využívá systém SAP HANA pro analýzu fotbalistů a různých herních situací. Společnost SAP na základě těchto zkušeností během šesti týdnů usilovné práce vyvinula, a následně těsně před MS ve fotbale představila, novou aplikaci SAP Match Insights, kterou na posledním šampionátu využívala právě fotbalová reprezentace Německa. Možná i tento software pomohl národnímu týmu k titulu z tohoto „sportovního svátku“ a firmě SAP udělal dobrou reklamu.

Aplikace využívá speciální senzory v podkolenkách hráčů a záznamy z osmi kamer. Veškerá sesbíraná data jsou ukládána a následně vyhodnocována téměř online. Trenéři vidí v aplikaci důležité údaje, ať už na svém notebooku, tabletu, případně pomocí Google Glass. Mohou si prohlížet statistické informace o hráčích, ale také zpětně analyzovat situace, které vedly k inkasování nebo vstřelení gólu a podle toho upravit taktiku týmu, v případě zhoršení výkonu vystřídat hráče [12].

Určitě se jedná o zajímavou technologii, která využívá nejmodernější databázová řešení a poskytuje realizačnímu týmu nástroje, které dosud neměl.

2.8 Sloupcově orientované databáze

Sloupcově orientované databáze využívají ukládání dat do sloupců, což je nový přístup k ukládání dat (column-store). U dnes běžných relačních databází jsou data ukládána po řádcích (row-store). Sloupcové databázové systémy označujeme jako Column-oriented DBMS. Každý atribut entit využívá pro uložení záznamů vlastní tabulku. Takovéto řešení je výhodné, pokud dotaz pracuje pouze s omezenou skupinou atributů tabulky, čímž dochází k menšímu množství čtených dat, což zvyšuje rychlost vyhodnocení dotazu [19]. Jako zástupce open-source sloupcových databází můžeme zmínit C-Store [19] nebo MonetDB[11], komerčním řešením je Sybase IQ[12], HP Vertica[13].

Zajímavou vlastností sloupcových databází je možnost, kromě samotného uložení dat, jednotlivým atributům uložit jednotlivé atributy rozšířené o interní identifikátor záznamu, případně bez identifikátoru pouze jako pole hodnot, kdy použijeme jako identifikátor pozici v souboru.

[11] https://www.monetdb.org
[12] http://www.sap.com/pc/tech/database/software/sybase-iq-big-data-management/index.html
[13] http://www.vertica.com/

Soubory všech atributů jsou ukládány ve stejném pořadí. Další vlastností sloupcových databázových systémů je možnost komprimovat data sloupců, pokud jsou seřazeny. Sloupcové databáze jsou efektivní zejména pro data, která často čteme, nikoliv pro často měněná data. Z tohoto důvodu je použití sloupcových databází výhodné pro tzv. „OLAP provozy", kde je běžné provádět velký počet složitých dotazů a téměř žádné průběžné zápisy. Ty jsou prováděny v dávkách, aby nezatěžovaly systém při čtení. Průchod entitami pro nalezení adekvátních hodnot pro menší množství atributů je značně rychlejší z důvodu omezeného počtu načítaných dat [19].

Při zkoumání vhodného využití sloupcově orientovaných databází bylo zjištěno, že jsou vhodné pro ukládání tzv. „širokých a řídkých dat". Tato data jsou definována jako taková data, ve kterých mají entity velmi velký počet možných atributů, ze kterých je ovšem vyplněna jen zanedbatelná část pro konkrétní entitu [17].

Podrobněji si představíme sloupcově orientované databáze v následující kapitole.

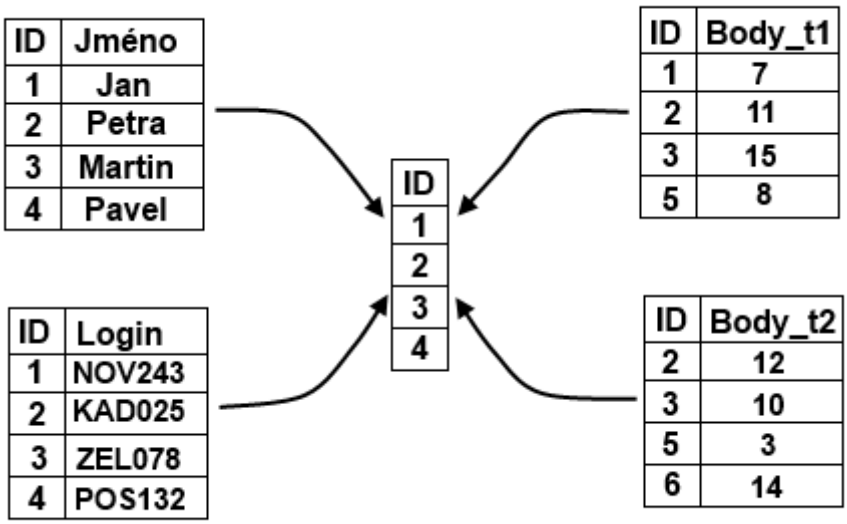
3 Sloupcově orientovaný databázový systém - sloupcově orientovaná databáze

Sloupcově orientované databáze též nazýváme Column Based Relational Database (CRBD). Jak již bylo zmíněno na konci předchozí kapitoly, data jsou v těchto databázích ukládána do sloupců. Vertikální ukládání dat může pro zkušené programátory při prvním kontaktu znít poněkud věděcky. Spíše než teorií na papíru se nás vědci snaží simulacemi a testy přesvědčit o vhodnosti tohoto řešení při konkrétních úlohách a vhodných případech. V této kapitole se budeme podrobněji věnovat popisu sloupcově orientovaných databázových systémů a vhodnému využití sloupcových databází v praxi.

3.1 Vertikální model

Vertikální model je velmi moderní technologický koncept, který je zajímavý z několika důvodů:

- nasazení je velmi efektivní a dokáže ušetřit nemalé finanční prostředky,
- tato technologie rozšiřuje možnosti v oblasti hromadného zpracování dat,
- v praxi prokázány velké přínosy.



Obrázek 10: Vertikální partitioning

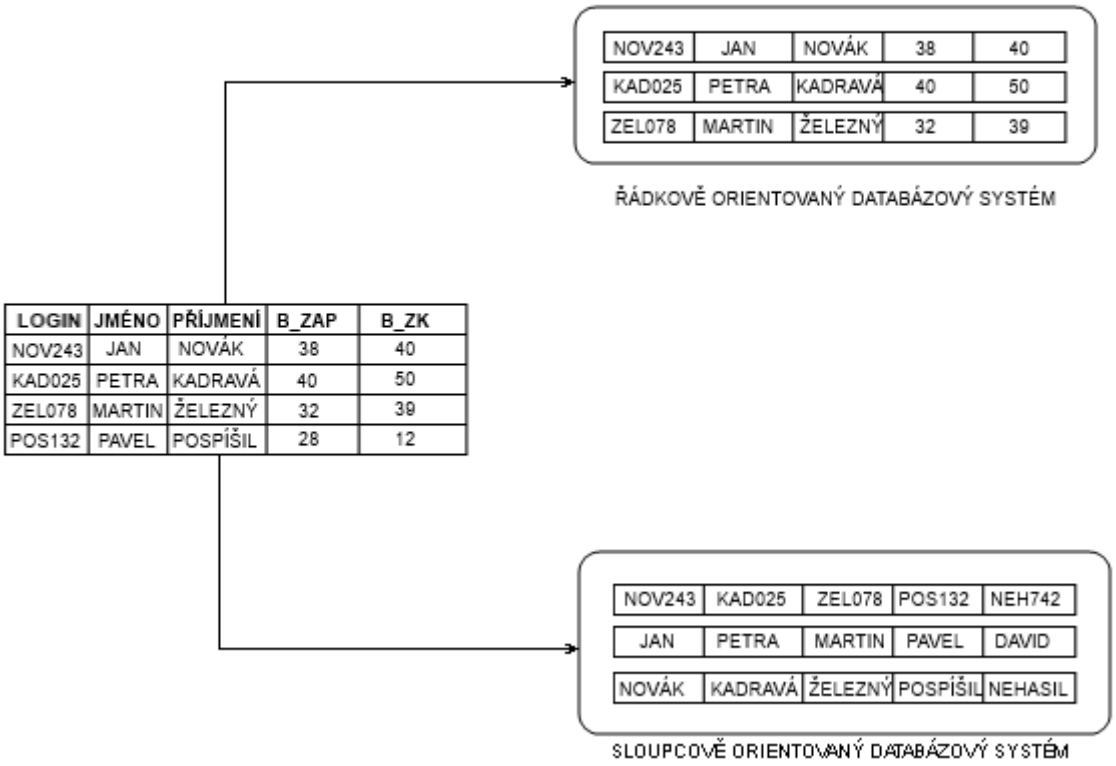
Každý atribut je ve vertikálním modelu uložen ve vlastní tabulce, tato tabulka obsahuje kromě hodnoty atributu pouze identifikátor entity. Takovéto ukládání dat je vlastně

úplný vertikální partitioning relačního modelu, který můžeme označit za nejjednodušší způsob simulace sloupcově orientovaného databázového systému (viz obrázek 10).

Hlavní nevýhodou řešení úplného vertikálního partitioningu v řádkově orientované databázi je potřeba velké režie hlavičky řádků (tzv. „tuple header“). K dalším nevýhodám také patří fakt, že při zápisu nové entity nebo změněné entity je třeba údaje propsat na více míst na disku [17].

3.1.1 Srovnání principu ukládání dat v řádkově vs. sloupcově orientovaných databázích

V této podsekci na obrázku 11 krátce demonstrujeme způsob, jak se ukládají stejná zdrojová data do relační databáze a jak se uloží do sloupcově orientované databáze.



Obrázek 11: Rozdíl ukládání dat

3.2 K čemu slouží vertikální ukládání dat

Technologie vertikálního ukládání dat je určena pro aplikace nazývané systémy pro podporu rozhodování (DSS), datové sklady (DW), Business Intelligence (BI), dolování dat (DM) a dalších. Právě pro všechny tyto systémy je společnou vlastností fakt, že pracují s daty vzniklými v jiných systémech.

Jedná se obvykle o data z podnikových agend (výroba, obchod, účetnictví, personalistika). Tato data jsou analyzována za účelem zjištění úzkých míst, ale také za účelem analýzy rizik, měření efektivity (výkonnosti), sledování trendů, identifikaci problémů, segmentaci zákazníků, hodnocení dodavatelů, modelování vývoje a mnoha dalších faktů obvykle dle požadavků manažerů firem.

Právě tato analýza dat, kde dochází k sekundárnímu zpracování informací z původních operativních systémů, je šita na míru vertikálnímu ukládání dat. Ač sloupcové databáze, dá se říct, vychazejí z konceptu relačních databází a na první pohled se z vnějšku od nich příliš neliší, využívá se odlišných metod, které umožňují složitější dotazy nad velmi rozsahlými daty v daleko kratší době, než při srovnání s relačními databázemi, dokonce i při současné práci velkého počtu uživatelů.

Přitom vyžaduje daleko menší diskový prostor a důraz na výkon hardware není tak vysoký, jako u dnes stále „klasických“ systémů. Dalo by se tedy říct, že změna principu ukládání dat je cestou k vytvoření a provozování relativně velkého datového skladu bez nutnosti velkých investic [13].

3.3 Princip vertikálního ukládání dat

U relačních databází, které jsou stále nejpoužívanější variantou databází pro podnikové aplikace, je optimálně vyvážena jednoduchost použití, funkcionalita a výkonnost, to vše stále s přijatelnými náklady. Proto jsou stále velmi vhodné zejména pro ukládání a zpracování dat pro transakční aplikace.

Základní strukturou relačních databází jsou tabulky, kde řádek tabulky odpovídá jednomu záznamu o jednotlivých objektech daného typu a každý sloupec obsahuje dílčí údaje o tomto objektu. Toto fyzické uspořádání, kdy obsah tabulek je ukládán po řádcích (tedy jednotlivých záznamech - větách), je velmi vhodné právě pro transakční zpracování.

Databázový stroj zde pracuje efektivně, jelikož dokáže velmi rychle vyhledat požadovaný záznam, přečte jeho obsah, uzamkne jej, přepíše atributy a záznam odemkne. Celému tomuto kroku říkáme databázová transakce. K rychlému nalezení záznamu se používají indexy na bázi B-stromů, které slouží právě k velmi rychlé navigaci mezi záznamy.

Relační databáze však díky své struktuře nejsou tak vhodné pro reporty a analytické dotazy vyžadující agregace a výpočty nad velkým objemem dat, která jsou obsažena v různých záznamech. Zde narážíme na omezení relačních systémů, které jsou limitovány právě z pohledu objemu zpracovávaných dat, rychlosti a také počtu současně pracujících uživatelů.
Pokud u těchto systémů překročíme limity, začnou relační systémy využívat extenzivní indexování, masivně paralelní zpracování nebo paralelní zpracování v clusterech. Všechny zmíněné způsoby jsou však velmi náročné na výpočetní výkon, diskový prostor a tím dochází ke zvyšování nákladů za výkonný hardware.

Vhodnou alternativou relačních databází pro danou problematiku je vertikální ukládání dat. Tento přístup k organizaci dat nám umožňuje vytvořit rozsáhlé datové sklady (s daty v desítkách TB) při daleko menších finančních nákladech, než v případě použití relačních databází.

Základem je nahrazení řádkového ukládání používaného u relačních databází za ukládání po sloupcích databázových tabulek [13].

Tato organizace dat přináší:

- atributy a veličiny se stejným obsahem a formátem jsou uloženy pohromadě,
- při složitých výpočtech pracujeme pouze s vybranými relevantními daty pro konkrétní výpočty (omezí se tak čtení dat z disků a zápis do paměti),
- omezíme počet přístupů na disk (o několik řádů), časově nejnáročnějších operací databázového systému,
- díky práci s homogenními poli (vektory) je můžeme fyzicky ukládat různými metodami, které využívají efektivní matematické postupy. V praxi nám to umožňuje pro každý typ dat současně použít zároveň několik metod ukládání a čtení, vždy dle výhodnosti pro daný typ operace (různé mechanismy jsou pro agregované výpočty, pro vyhledávání dat, práci nad různými typy textových položek, atd.),
- homogenita dat nám umožňuje velmi efektivní využití komprimačních algoritmů, díky kterým dojde ke zmenšení ukládaných dat a tím i zmenšení objemu přenášených dat,
- díky kompresi dat se snižují nároky na hardware – místo na úložišti, což nám přináší úsporu nákladů,
- databázové operace nemusí využívat navigaci ve stromech indexů, mohou místo ní použít výpočty booleovské algebry, které přinášejí další zrychlení.

3.4 Technologie vertikálního ukládání dat

V této sekci se zaměříme na popis technologie ukládání dat v sloupcových databázích. Záměna řádků a sloupců v tabulkách je matematicky uskutečnitelná, jedná se o záměnu řádků za sloupce matice (tabulky 2 a 3).

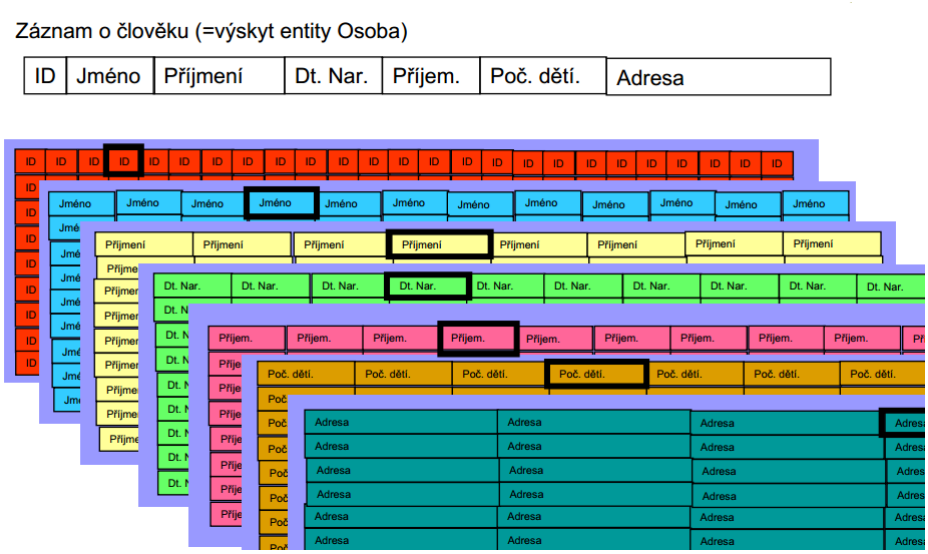
ID	NAZEV	KOD	CENA
1	zbozi1	a1	120
2	zbozi2	b2	220
3	zbozi3	c3	330
4	zbozi4	d4	440
5	zbozi5	e5	550
6	zbozi6	f6	660
7	zbozi7	g7	770
8	zbozi8	h8	880
9	zbozi9	i9	990
10	zbozi10	j10	1100

Tabulka 2: Klasický RDBMS

ID		NAZEV		KOD		CENA
1		zbozi1		a1		20
2		zbozi2		b2		220
3		zbozi3		c3		330
4		zbozi4		d4		440
5		zbozi5		e5		550
6		zbozi6		f6		660
7		zbozi7		g7		770
8		zbozi8		h8		880
9		zbozi9		i9		990
10		zbozi10		j10		1100

Tabulka 3: Sloupcově orientovaný DBMS

V databázích se ovšem o tak jednoduchou operaci nejedná, jelikož stále pracujeme se stejnými výskyty entit, přičemž musí být zachována jejich logická struktura integrity. V praxi musíme vlastně zajistit, aby při výpočtech nad shluky atributů byla ošetřena vazba na konkrétní atributy (jedná-li se například o atributy různých osob, musíme zachovat vazbu mezi konkrétními jedinci). Z tohoto důvodu je vývoj takového systému složitý. Jak vypadá výskyt entity osoba znázorňuje obrázek 12 od pana Kyjonky [13].



Obrázek 12: Výskyt entity osoba v CRDB. [13]

Popisovaná technologie vertikálního ukládání se skládá z několika typů struktur, v nichž jsou data ukládána, lišících se podle typu dat:

- technické (datové) typy – čísla, řetězce, ...
- strukturální typy – primární a cizí klíče, atributy, metriky, ...
- logické typy – adresa, věk, peněžní částka, ...

Podstatná je zde kardinalita dat (počet různých hodnot v celé množině).
Každému typu struktury jsou vytvořeny speciální algoritmy pro zápis, výpočty agregací, projekcí, ... Jednotlivé sloupce tabulek mohou mít současně přiřazeno více typů speciálních struktur zastupujících různé přístupové metody.

Nyní si představíme některé typy indexů, jak jsou ve vertikálním modelu implementovány.

Low-Fast index (LF) – tento typ indexu se používá pro sloupce, které mají nízkou nebo velmi nízkou kardinalitu. K dispozici máme hned několik poddruhů těchto indexů, které jsou variantou klasických bitmap, kde pozice bitu indikuje přítomnost dané hodnoty a jsou zde uplatněny účinné komprimační algoritmy. Tyto indexy jsou nejrychlejší ze všech typů a využívají velkou datovou kompresi. Dokládá to i místo potřebné pro uložení dat v těchto indexech, které odpovídá asi 20 % původní velikosti dat. Nejvhodnější jsou tyto

indexy pro agregace, aritmetické operace, seskupování podobných dat, spojování tabulek, výpočty a projekce v klauzuli „where“ s neurčitými podmínkami (mezi které patří =, <=, >=, <>, IN, NOT IN, ...) [13].

Kombinované bitmapy s invertovanými seznamy – jsou určeny pro data s vyšší kardinalitou, obvykle v řádu stovek (do 1500) různých hodnot. Na určenou pozici zapisujeme maximálně dvoubytový kód, tento kód pak odpovídá v lookup tabulce s komprimovanými bitmapami konkrétní hodnotě.

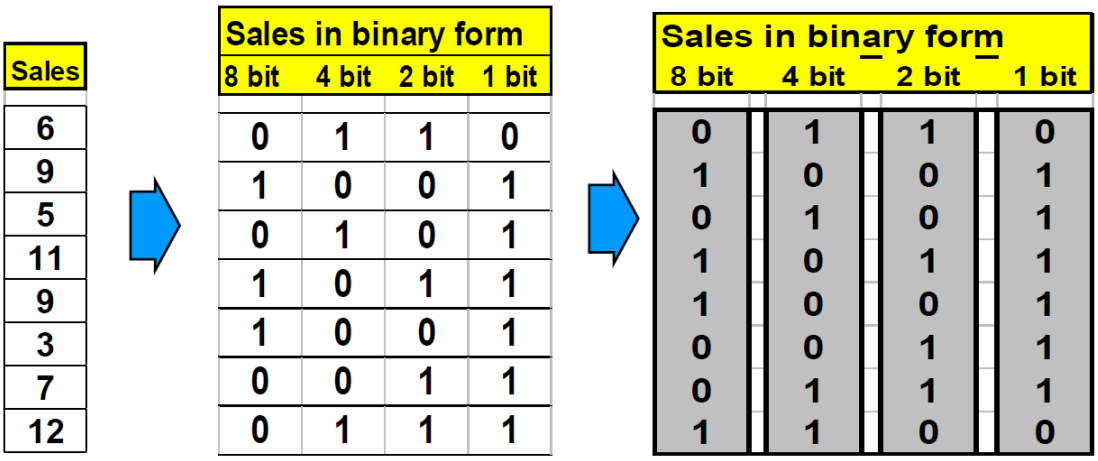
High-Group index (HG) – je prvním zástupcem indexů, který je vhodný zejména pro data s vysokou kardinalitou. Používá se tam, kde se často provádí „groupování“ pomocí GROUP BY. Autor publikace [13] uvádí, že se jedná o indexy na bázi B-stromů, jež jsou vysoce komprimované a jejich nejnižší úroveň je uložena jako speciální zřetězená pole (G-Array).
Je to další velmi rychlý index, který se využívá pro sloupce definované jako primární klíče. Mezi jeho hlavní nevýhody ovšem patří náročnost na místo na disku (80-120 % velikosti původních dat), což přináší i nevýhodu v nejvyšší časové náročnosti při ukládání dat ze všech popsaných indexů.

High-non-Group index (HNG) – je druhým zástupcem indexů, vhodný zejména pro sloupce s vysokou nebo velmi vysokou kardinalitou. Původní data se dělí na menší celky (řetězce na znaky, čísla na bity) a ty následně teprve ukládány do samostatných bitmap. HNG indexy jsou vhodné pro agregace, aritmetické operace a vyhledávání na určitém intervalu (BETWEEN, NOT BETWEEN, <, >, <=, >=, ...).
Opět se bavíme o velmi rychlém typu indexů. Velkou výhodou oproti předchozímu indexu je fakt, že má také velmi malou náročnost na místo na disku (2-20 % původních dat) a dokonce je velmi rychlý i při nahrávání dat.

Při ukládání dat rozkládá High-non-Group index čísla na jednotlivé řády. Ilustraci, jak tento rozklad vypadá ukazuje obrázek 13 z publikace [13].

Fast-Projection index (FP) – představuje univerzální index určený pro různé kardinality. Do jednotlivých podtypů tohoto indexu se ukládají pouze vysoce komprimovaná data dle různé kardinality. Každý z podtypů se vyznačuje specifickými komprimačními algoritmy, používaných právě podle kardinality dat. Výhodou tohoto indexu je jeho velmi malá paměťová náročnost (asi 1 % vstupních dat) a rychlost při nahrávání dat. Používá se pro ad-hoc slučování tabulek, projekci dat a prohledávání řetězců v klauzuli *WHERE*.

Word index (WD) – speciální index pro indexování jednotlivých slov v řetězcích. Je novým indexem v systémech Business Intelligence a datových skladech, který vznikl na základě potřeby indexace dat s méně striktní strukturou (adresy, URL adresy, lékařské diagnózy, popisy událostí, ...).



Obrázek 13: Rozklad čísel na jednotlivé řády pomocí HNG [13]

Compare index (CMP) – index obsahující výsledky porovnání hodnot různých sloupců.

Join index (JOIN) – index definovaný nad sloupci více tabulek současně.

Date, Time, Date Time – indexy pro efektivní práci s datem, časem a jeho kombinací (časovým razítkem).

V současných systémech jsou všem typům dat přiřazovány indexy FP, automaticky je ošetřován přechod mezi indexy a jejich podtypy při změně kardinality [13].

3.5 Výhody vertikálního ukládání dat

V předcházející části jsme uvedli, že sloupcově-orientované databáze jsou vhodné pro zpracování velkého objemu dat, nad kterými se provádí náročné analýzy a reporting. Jedná se tedy o oblast datových skladů, kde se setkáváme s výrazy jako Business intelligence (BI), dolování dat, on-line analytické zpracování dat, aj. V dnešní době se objem světových dat každý druhý rok zdvojnásobí. Obrovské množství dat denně sbírají mobilní operátoři, obchodní řetězce, pojišťovny, banky, bezpečnostní složky státu. Právě při analýze těchto dat je důležité zvolit správné řešení datového skladu.

V této oblasti přináší vertikální ukládání dat výhody:

- ekonomické výhody,
- funkčně-technologické výhody.

V následující části si jednotlivé výhody krátce představíme.

3.5.1 Ekonomické výhody

V případě použití vertikálního ukládání dat je díky použití komprimačních algoritmů potřeba několikrát méně prostoru na disku, než při ukládání stejného objemu vstupních dat v tradičních systémech. Přes relativně příznivé ceny diskových polí přináší nemalé úspory, protože obecně se potřebné množství zpracovávaných dat neustále zvětšuje (meziroční nárůst dat je u datových skladů cca 40 %) [13]. Samotná redukce potřebných diskových polí nám umožní použít také méně výkonný hardware, což umožní menším i středně velkým firmám budovat datové sklady s omezenými investičními náklady. U středních firem, které používají rozsáhlé systémy, může úspora představovat velmi zajímavou částku.

Nemluvíme ovšem pouze o vstupních investicích, jelikož musíme myslet i do budoucna. Při budování datového skladu si musíme položit otázku: „Kolik náš systém bude zabírat diskového prostoru a jak výkonný hardware budeme potřebovat pro analýzu dat, budeme-li počítat s meziročním nárůstem množství dat 40 %?“. Přesně to bychom měli zpracovat v analýze „Total Cost of Ownership (TCO)“.

Neopomenutelnou součástí analýzy Total Cost of Ownership je také lidská práce, která zahrnuje správu a údržbu systému, optimalizaci, ladění, indexování, ...
Vertikální ukládání tuto položku redukuje na minimum ve srovnání s tradičními systémy.

Dle dostupných informací je u vertikálního ukládání zvyšování nákladů s rostoucím objemem dat lineární, stejně tak rostou lineárně i náklady pro obsluhu více uživatelů. U tradičních databázových systémů je tento trend daleko rychlejší.

3.5.2 Funkčně-technologické výhody

Mezi funkčně-technologické výhody sloupcově orientovaných databází patří rozhodně rychlost, flexibilita, snadná implementace a správa a v neposlední řadě i počet současně pracujících uživatelů. Tyto výhody si postupně představíme, začneme první jmenovanou, tedy rychlostí.

Rychlost systému je zřetelná hlavně při vyhodnocování optimalizovaných dotazů pro vertikální systémy, ale projevuje se také u nahrávání dat . V případě shodných dat a při použití stejného hardware je využití vertikálního ukládání daleko rychlejší až o několik řádů. Při zpracování rozsáhlých dat a při použití vhodných dotazů je tato výhoda nejvíce zřetelná. Redukce počtu diskových operací je klíčovou vlastností k právě popsanému zvýšení rychlosti zpracování dat, ke které dochází čtením pouze potřebných dat a zároveň jejich vysokou kompresí.

Vysokou rychlost si systém zachovává i s rostoucím počtem uživatelů. V případě jiných technologií se setkáváme s neúměrným prodlužováním odezvy v závislosti na počtu současně pracujících uživatelů (běžné datové sklady). U tradičních systémů je ty-

pickým jevem , že i při použití velmi výkonného hardware došlo k překročení „hranice“ počtu uživatelů a systém se díky vysoké odezvě stal nepoužitelným. Takovýto systém dokáže standardně obsloužit asi desítku aktivních uživatelů.

U technologie vertikálního ukládání dat se s rozšiřujícím počtem aktivních uživatelů prodlužuje doba odezvy jen mírně. Tento lineární nárůst odezvy je klíčem k tomu, že při použití této technologie systém obslouží daleko více uživatelů.

Jelikož se o aktuálnost veškerých informací starají již metody pro ukládání dat, je u vertikálního ukládání dat třeba pouze minimálně optimalizovat a ladit. Proto můžeme vynechat reindexování, rebalancování a update statistics, používané u tradičních systémů. Systém tak získává neomezenou flexibilitu, kterou využijeme při zadávání velkého množství adhoc dotazů, na které není běžný systém předem možné vyladit. Problémem není ani přidávání nových dat, změna struktur a schémat jejich uložení. Můžeme pracovat s daty ve 3. normální formě, variantách hvězdicových i vločkových schémat, dokonce v modernějších modelech Relational Data Cubes či Data Vault.

Abychom byli schopni určovat správné typy indexů pro zdrojová data, stačí nám osvojit si několik jednoduchých pravidel. Díky použití vhodných indexů pak odpadá potřeba ladit systém, což usnadňuje jeho samotnou údržbu. Poslední výhodou může být menší objem dat, který se jednodušeji zálohuje.

3.6 Omezení vertikálního ukládání dat

Stejně jako relační databáze mají i sloupcově orientované databáze svá omezení, tedy aplikace, pro které není použití vertikálního ukládání vhodné.

Sloupcově orientovaná databáze pro transakční systém – použití vertikálního ukládání pro ERP systém, učetnictví a podobné systémy není rozhodně vhodné. Každý transakční systém je postavený na zamykání záznamů. Systém vertikálního ukládání je však nejefektivnější pouze v případě, že jsou záznamy co nejméně zamykány, protože logický záznam je ukládán v mnoha oddělených fyzických strukturách. Proto použití vertikálního ukládání v prostředí s velkým počtem uživatelů, kdy každý uživatel provádí hodně změn, je krajně nevhodné. Doménou vertikálního ukládání zůstává tedy oblast datových skladů a Business Intelligence. Naopak pro transakční zpracování je vhodné vždy použít tradiční systémy.

Dalším omezením je potřeba kalkulace **vhodnosti použití vertikálního ukládání pro malé aplikace s tzv.„podkritickou“ velikostí**. Nasazení nového systému vyžaduje nějaké zdroje (ať už se jedná o finanční, hardwarové, lidské) a také provozní náklady, proto je třeba dobře analyzovat, zda využít stávající řešení, nebo nasazovat nový systém. Zde opět uplatníme i dříve popsanou analýzu TCO.

Posledním omezením je absolutní nevhodnost vertikálního ukládání pro **archivaci**

dokumentů a binárních objektů. Pro tuto oblast jsou vyvinuta specializovaná řešení, která tento úkol zvládají a jsou pro něj nejlépe optimalizovaná [13].

3.7 Vhodné aplikace pro sloupcově orientované databáze

V předcházejícím textu bylo zmíněno, že se vertikální ukládání doporučuje použít u nových datových skladů a v oblasti Business Intelligence.

Podle [20] jsou sloupcově orientované databáze vhodné zejména pro:

- Datové sklady - Data Warehousing,
- Dolování dat - Data mining,
- Google Big Table,
- RDF - Resource Description Framework - základ sémantického webu [21],
- Vyhledávání informací - Terabyte TREC,
- Vědecká data - SciDB initiative, SLOAN Digital Sky Survey on MonetDB.

Komerční produkt Sybase IQ od společnosti Sybase byl celosvětově nasazen v několika stovkách případů. Mezi nejzajímavější klienty patří Nielsen Media Research (organizace zabývající se sledovaností televizních programů) a Letiště Václava Havla v Praze.

Letiště Václava Havla v referenci datového skladu Sybase IQ uvedlo, že díky tomuto řešení mají spolehlivé, vhodně strukturované a včasné informace o leteckém provozu.

Zástupci firmy Nielsen Media Research byli daleko sdílnější. Kim Ross (Chief Information Officer) prohlásil: *„Výslovně jsme si vybrali Sybase IQ, aby sloužila jako platforma pro naše příznivce datového skladu. Protože je optimalizován pro analýzy v rámci velmi velkých objemů dat. Používáme správné technologie pro konkrétní úkoly. Pro naše příznivce datového skladu je Sybase IQ ta pravá technologie.“*

Dále na svých webových stránkách uvadějí klíčové výhody použití Sybase IQ:

- velmi rychlé vyhodnocení dotazů i nad rozsáhlými historickými daty,
- 70% kompresní poměr přináší zásadní úsporu nákladů za skladování dat,
- 10 až 100 krát rychlejší vyhodnocení dotazů i při zpracovávání terabytů dat,
- snadné a rychlé přidání nových klientských produktů a aplikací,
- odpadlo měsíční zasílání CD-ROMů klientům,
- došlo ke snížení složitosti, nákladů i rizik při zavádění nového datového skladu.

Organizace pro výzkum v oblasti ukládání dat a velmi velkých databází Bloor Research vydala prohlášení, že sloupcově orientované databáze (CRDB) budou tím, čím byl OLAP pro 90. léta, jak ve své práci cituje pan Kyjonka [13].

4 Vlastní sloupcově orientovaný DBMS

Jedním z bodů, které si v práci představíme, je i praktická ukázka toho, zda změna v ukládání dat bude mít opravdu vliv na rychlost čtení velkého objemu dat. Z předchozího textu se můžeme domnívat, že výrazně rychlejší by měl být sloupcově orientovaný systém oproti klasickému řádkovému pojetí v případě projekce.

Abychom porovnali systémy a mohli učinit nějaký závěr, provedeme sadu dotazů nad stejnými daty v následujících implementacích:

- Vlastní sloupcově orientovaný databázový systém,
- Existující implementace řádkového pojetí,
- HP Vertica,
- MS SQL Server 2012 Express.

4.1 Zopakování základních pojmů

- **Typ entity** – je objekt z reálného světa, jehož vlastnosti chceme evidovat (jedná se o zápis hodnot atributů). Např. typ entity *Student* s atributy *jméno, příjmení, login*. Entitní typ zapisujeme jako n-tici: *<název entity> (<název atributu 1>, <název atributu 2>, <název atributu 3>* zapisujeme *Student (jméno, příjmení, login).*
- **Atribut** – vlastnost objektu reálného světa, který evidujeme. Pro entitní typ *Student* budou typickými atributy *jméno, příjmení* a *login*.
- **Entita** – je popsána jménem a množinou atributů, kdy se jedná o konkrétní výskyt typu entity
  Např. entitní typ *Student (jméno, příjmení, login)*. Entitou typu *Student* je např. *('Martin', 'Parma', 'PAR125')*.

4.2 Vlastní implementace sloupcově orientovaného databázového systému

Pro implementaci vlastního sloupcově orientovaného databázového systému bude po konzultaci s vedoucím práce použit existující projekt databázového systému QuickDB[14], který je pro výzkum tradičních řádkových databází na VŠB-TU vyvinut, s využitím databáze QuickDB.

4.2.1 Stávající řešení v projektu QuickDB

Současné řešení pro tradiční databázi (tedy ukládání atributů řádkově) je v projektu QuickDB řešeno skrz datovou strukturu cTuple. Tato struktura obsahuje určitý počet

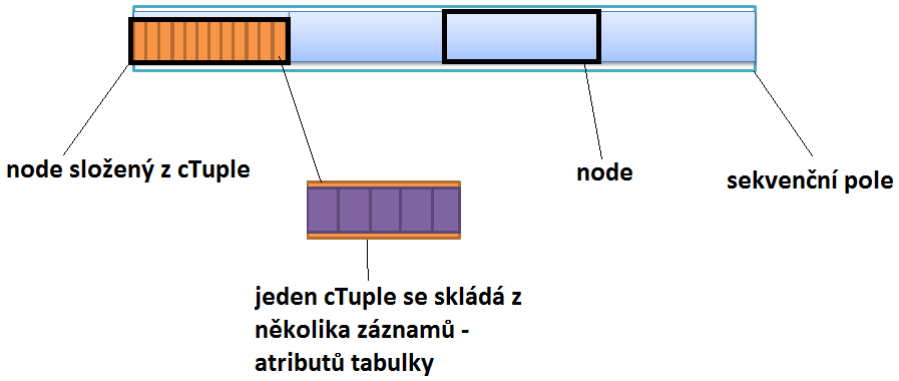
[14] http://db.cs.vsb.cz/SubPages/Projects/Projects.aspx

a pořadí prvků. Zdrojové soubory tohoto řešení jsou k dispozici na přiloženém DVD nosiči a celé řešení je umístěno v třídě RowStore.

Při definici je definován počet prvků a příslušný datový typ prvků. Celý cTuple se pak ukládá do sekvenčního pole. Sekvenční pole je složeno z tzv. nodů. Tyto nody obsahují maximálně 8192 B informací. Do těchto nodů jsou ukládány záznamy – právě předem zmíněné datové struktury cTuple (jak znázorňuje obrázek 14).

Při použití základních pojmů můžeme popsat cTuple a zápis do sekvenčního pole následujícím způsobem. cTuple nám v daném řešení zastupuje jednoduchý zápis entity. Prvky cTuple jsou atributy evidované entity ukládané v této struktuře do databáze. Entitní typ ukládáme v podobě cTuple do nodu bez rozlišení názvu entitního typu. Každý typ entity je ukládán do sekvenčního pole dříve popsaným způsobem, do jednotlivých nodů o maximální velikosti 8192 B.

Z tohoto modelu budeme vycházet i v případě sloupcového systému. Bude ovšem nutné předělat systém tak, abychom nepracovali v řádcích, ale mohli data ukládat (a následně číst) po jednotlivých sloupcích.



Obrázek 14: Existující systém ukládání informací řádkovým způsobem

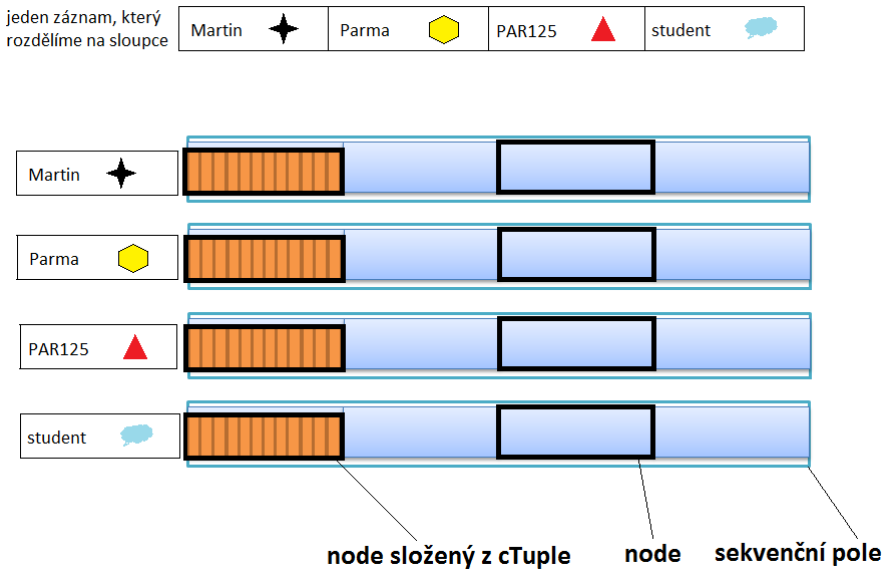
4.2.2 Návrh fyzického modelu sloupcově orientovaného systému

V daném projektu bude nutné upravit systém ukládání dat, abychom s uloženými daty mohli pracovat sloupcově.

V již zmíněném řešení jde o projekt „sequentialArray“. Ten původně obsahoval kód pro vytvoření sekvenčního pole a zápis do databáze tradičním způsobem.

Náš sloupcově orientovaný systém bude využívat, stejně jako v řešení pro řádkový přístup, sekvenční pole a datovou strukturu cTuple, nyní o pevné velikosti jednoho záznamu. Každé sekvenční pole bude odpovídat jednomu atributu tabulky, hodnoty v něm

budou tedy fyzicky oddělené a budeme mít možnost přečíst celý sloupec najednou. Toto řešení je znázorněno na obrázku 15, kde jeden záznam ('Martin', 'Parma', 'PAR125', 'student') rozdělíme do jednotlivých sloupců a jim odpovídajícím sekvenčním polím.



Obrázek 15: Fyzický model navrženého sloupcového systému

4.3 Implementace sloupcového systému

Sloupcový systém je naimplementován v třídě cColumnStore, která je opět přiložena na DVD. Pro každý atribut je zde vytvořeno samostané sekvenční pole v našem sloupcovém systému. Ve výpisu 1 je ukázka zdrojového kódu metody *prepare()*, který se stará o načtení zdrojových dat.

```
void cColumnStore::prepare()
{
        vector<vector<int>> dataRows = fileopen(filename);

        for (unsigned int i = 0; i < ITEM_COUNT; i++)
        {
            for (unsigned int j = 0; j < mNOfAttributes; j++)
            {
                mItemOfFile[i]->SetValue(j, dataRows[i][j], All);
            }

        }
        cout << "Uspesne vytvoreno " << mNOfAttributes << " atributu a naplneno " <<
              ITEM_COUNT << " hodnotami." << endl<< endl;
}
```

Výpis 1: Metoda prepare() pro načtení zdrojových dat.

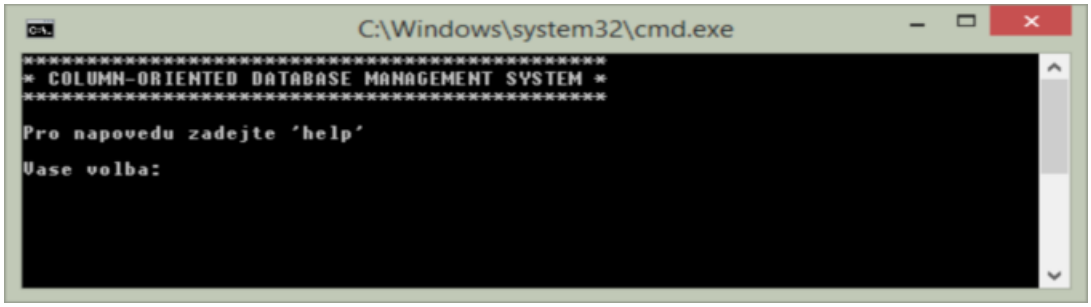
Dále je třeba tato sekvenční pole uložit do databáze. Použijeme tedy databázi QuickDB, kterou nám projekt nabízí. Definice metody *create() 2* nalezneme ve výpisu níže.

```
bool cColumnStore::create()
{
    bool debug = false;
    cTimer runtime;

    if (!quickDB->Create("quickDB", CACHE_SIZE, MAX_NODE_INMEM_SIZE, BLOCK_SIZE))
    {
        printf (" Critical ␣Error:␣Cache␣Data␣File␣was␣not␣created!\n");
        exit (1) ;
    }

    // headers
    for (int i = 0; i < mNOfAttributes; i++)
    {
        mHeader[i] = new cSequentialArrayHeader<tKey>("seqarray_"+i+97, BLOCK_SIZE, A,
              cDStructConst::DSMODE_DEFAULT);
        mHeader[i]->SetCodeType(ELIAS_DELTA);
    }

    // sequential array
    for (int i = 0; i < mNOfAttributes; i++)
    {
        if (!mSeqArray[i]->Create(mHeader[i], quickDB))
        {
            printf ("Sequential␣array:␣creation␣failed\n") ;
            return false ;
        }
    }

    runtime.Start () ;
    for (unsigned int i = 0; i < ITEM_COUNT; i++)
    {
        for (unsigned int j = 0; j < mNOfAttributes; j++)
        {
            mItem[i]->SetValue(0, mItemOfFile[i]->GetUInt(j, All), A);
            mSeqArray[j]->AddItem(nodeid, position, *mItem[i]);
        }

    }
    runtime.Stop();
    runtime.Print("␣-␣Insert␣time\n");
    printf ("Performance:␣%.1f␣Inserts/s\n", (mNOfAttributes*(mSeqArray[0]->GetHeader()->
          GetItemCount())) / runtime.GetRealTime());
    quickDB->Close();
    return true;
}
```

Výpis 2: Metoda create().

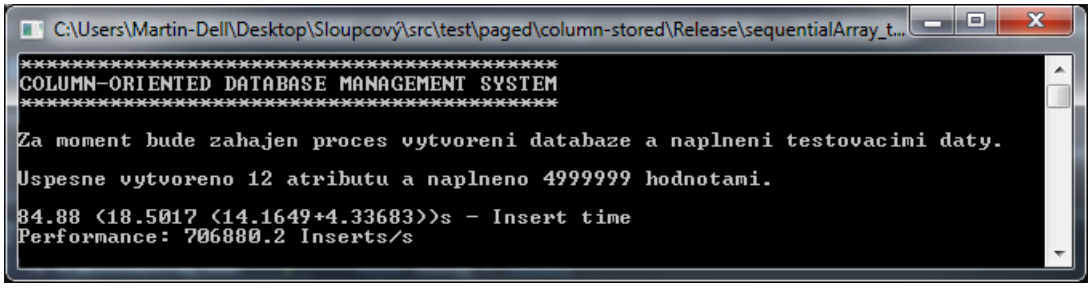
4.4 Představení sloupcově orientovaného databázového systému

Po spuštění programu 16 máme k dispozici příkazový řádek, kde pro nápovědu zadáme příkaz „help“. Důležitá volba se skrývá pod příkazem „prepare“, kterou použijeme v případě, že chceme vytvořit sloupcově orientovanou databázi a naplnit ji daty z přiloženého datasetu.



Obrázek 16: Úvodní obrazovka systému.

Zvolíme-li „prepare“, tedy vytvoření databáze a její naplnění daty 17, spustí se nám nahrání dat ze souboru a je vytvořena sloupcová databáze.



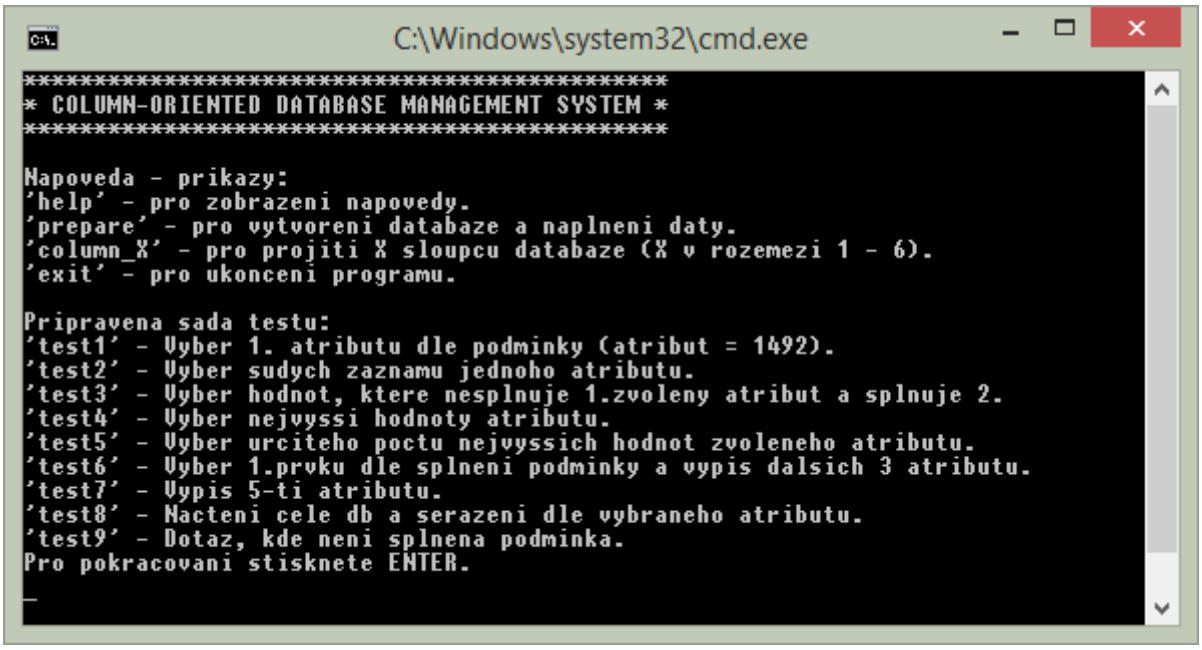
Obrázek 17: Vytvoření sloupcové databáze a naplnění daty.

Dále je nutné, představit si důležité parametry, které v systému používáme. Tabulka 4 obsahuje stručný popis použitých parametrů.

NAZEV	POPIS
ITEM_COUNT	počet záznamů v databázi
BLOCK_SIZE	maximální velikost jednoho nodu (v našem případě 8192 B)
MAX_NODE_INMEM_SIZE	maximální počet nodů v paměti
CACHE_SIZE	maximální velikost cache
TUPLE_SIZE	počet atributů v databázi (pouze u řádkového systému)
mNOfAttributes	počet atributů v databázi (pouze u sloupcového systému)

Tabulka 4: Důležité parametry.

Popis všech příkazů je k dispozici v již zmíněné nápovědě 18, stručně rozepsané v tabulce 5.



Obrázek 18: Nápověda systému.

Příkaz	**Popis**
help	zobrazení nápovědy
exit	ukončení aplikace
prepare	vytvoření databáze a naplnění daty
column_X	vypíše X sloupců (1 – 6)
test1	průchod 1.atributem do nalezení hodnoty 1492
test2	výpis počtu sudých hodnot 7.atributu
test3	průchod 3. a 11. atributem do nalezení hodnoty 37 v 11.atributu
test4	výpis nejvyšší hodnoty 3.atributu
test5	výpíše 10 nejvyšších hodnot 4.atributu
test6	výpis čtyř atributů (2., 1., 3., 6.) pro podmínku „první hodnata 2.atributu = 7033“
test7	průchod pěti atributů (1., 2., 3., 4., 6.)
test8	průchod celé db a seřazení dle 1. atributu
test9	průchod celé db pro podmínku nevyhovující žádnému záznamu

Tabulka 5: Seznam příkazů

5 Testování databázových systémů

V této kapitole si představíme průběh testů, porovnání s komerčními systémy a na závěr uvedeme krátké zhodnocení.

5.1 Naplnění systémů testovacími daty

Pro následující testy je důležité naplnit systémy shodnými daty. Byl zvolen datový soubor, obsahující 12 atributů (všechny záznamy obsahují pouze celočíselné hodnoty) s 4 999 999 záznamy.
Pro naplnění atributů implementovaných systémů je použita metoda *fileopen(string filename)*, která na vstupu požaduje zadání názvu souboru, jak ukazuje výpis 3. Volání metody je již intuitivní, přesto jej uvádíme na konci tohoto výpisu.

```
vector<vector<int>> cColumnStore::fileopen(string filename)
{
    vector<vector<int>> rows;
    ifstream file (filename);
    string line;
    while ( file.good() )
    {
        getline ( file, line, '\n' );
        if (line.length() < 2) continue;
        vector<int> values;
        std::stringstream lineStream(line);
        string hodnota;
        while (getline(lineStream, hodnota, ','))
        {
            values.push_back(stoi(hodnota));
        }
        rows.push_back(values);
    }
    return rows;
}

//zavolani metody
vector<vector<int>> dataRows = fileopen("dataset.txt");
```

Výpis 3: Metoda pro nahrání dat a příklad volání této metody.

5.2 Testování rychlosti čtení ve sloupcové databázi

Na základě popsaných úprav projektu byl vytvořen vlastní sloupcově orientovaný databázový systém. Tento systém porovnáme s původním řešením pro tradiční databáze.

5.2.1 Konfigurace testovacího počítače

Veškeré testy zde popsané provedeme na následující pracovní stanici (viz. tabulka 6). Jedná se o standardní pracovní stanici, na které po dobu testů byly spuštěny další pro-

gramy a služby, které mohly ovlivnit rychlost systému. Jelikož pracujeme s daty o velikosti jednotek milionů řádku, nebude třeba implementaci připravit jako 64 bitovou aplikaci, ale postačí nám standardní „win32“ aplikace.

Parametr	Hodnota
Procesor	Intel® Core™2 Duo P9400
Operační paměť	2 x 2 GB DDR2
Operační systém	MS Windows 7 SP1 64bit
Vývojové prostředí	MS Visual Studio 2012

Tabulka 6: Konfigurace testovacího počítače

5.2.2 Průběh testů

Testy provedeme postupně na čtyřech systémech. První srovnání nám poskytnou dvě implementace nad již zmíněným projektem QuickDB. Zde porovnáme jak čas vyhodnocení dotazů, tak počty diskových přístupů.

Další sadu testů provedeme posléze na komerčních systémech HP Vertica a MS SQL Server 2012 Express. Vytvoříme společné srovnání komerčních a nekomerčních systémů z pohledu doby zpracování dotazů.

Jako poslední z testů pak představíme srovnání komerčních řešení nad složitějšími daty a dotazy, abychom co nejlépe otestovali dostupná řešení a mohli učinit adekvátní závěr v našem zkoumání.

5.2.3 Popis testů

Pro otestování systémů je nachystána sada devíti testů, která je teoreticky namíchána tak, aby pokryla vhodnost použití pro oba druhy přístupu ukládání dat.

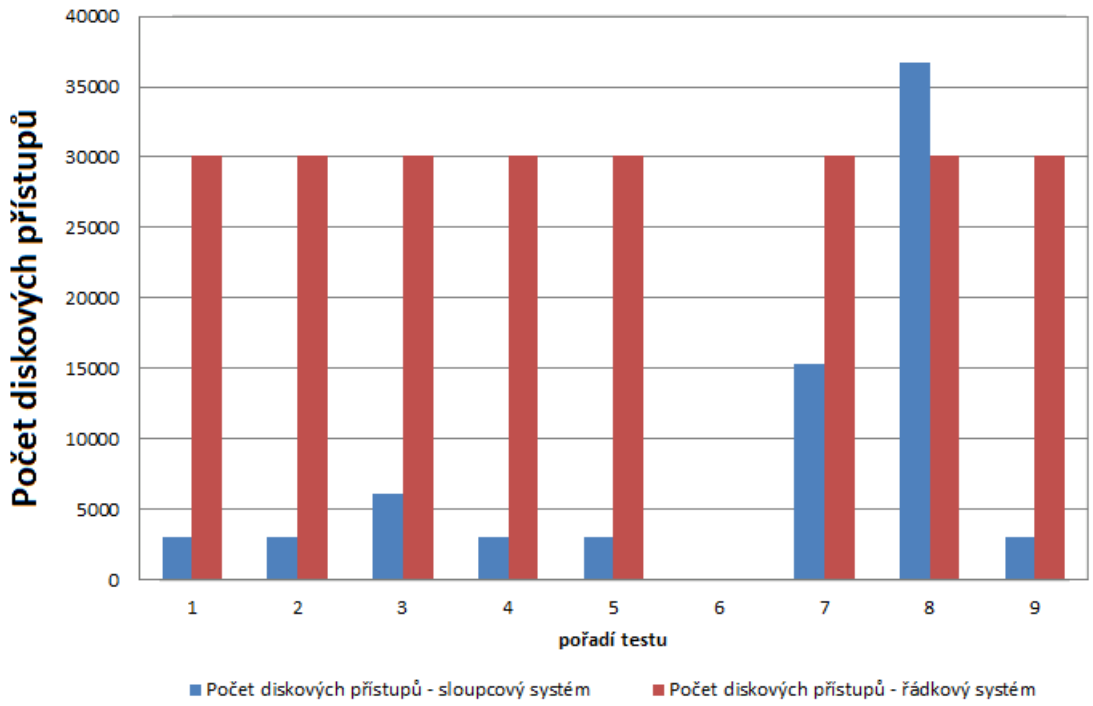
Testy jsou následující:

- výběr jednoho atributu splňující jednoduchou podmínku,
- výběr pouze sudých čísel jednoho atributu,
- počet záznamů, kdy není splnena žádná podmínka v jednom sloupci a je splněna podmínka v dalším (0 + 16827 hodnot)
- výpis nejvyšší hodnoty ze sloupce,
- setříděný výpis deseti nejvyšších záznamů jednoho sloupce,
- výpis prvního prvku splňujícího nějakou podmínku a k němu doplnění dalších tří atributů,

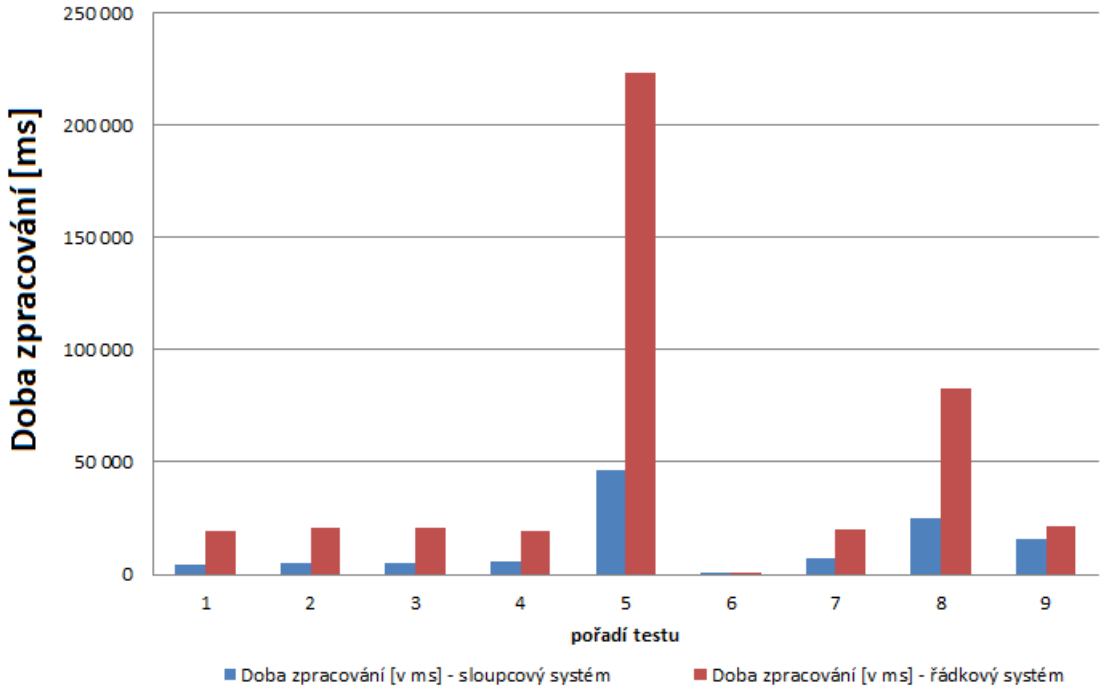
- výběr všech záznamů pro pěti atributů tabulky,
- výběr celé tabulky a setřízení dle prvního sloupce,
- počet prvků, které splňují nesplnitelnou podmínku - 0 vypsaných záznamů.

Výsledky testů jsou znázorněny v tabulce 7, a na přiložených grafech 19, 20. Graf s počty diskových přístupů jednoznačně potvrdil předpoklady, tedy nižší počet diskových přístupů, než u řádkového databázového systému.

Výsledek testů také potvrdil předpoklad, že počet diskových přístupů je v řádkovém systému při nutnosti projít všechny záznamy alespoň jednoho atributu stejný. Musíme totiž projít celé sekvenční pole a číst z něj chtěné záznamy. U sloupcového systému, ve kterém jsou sloupce uloženy samostatně, je počet diskových přístupů několikanásobně menší, jelikož čteme pouze potřebná data, což je graficky znázorněno na grafuh 19. Tomu v našem případě odpovídá i rychlost zpracování, která reflektuje menší množství diskových přístupů(viz. graf 20). Testy ovšem mohly být částečně ovlivněny „slabší“ konfigurací pracovní stanice (notebook), běžícími aplikacemi po dobu testu, bezpochyby také aktuálním zatížením pevného disku při čtení dat.



Obrázek 19: Počet diskových přístupů.



Obrázek 20: Doba vyhodnocení dotazů [v ms].

Pořadí testu	Počet výsledků	Sloupcový systém		Řádkový systém	
		Zpracování [v ms]	DAC	Zpracování [v ms]	DAC
1	3 933	4 056	3 062	19 017	30 121
2	2 496 376	4 867	3 062	20 545	30 121
3	16 827	5 242	6 124	20 499	30 121
4	1	5 444	3 062	19 515	30 121
5	1	46 454	3 062	223 751	30 121
6	1	187	52	78	123
7	5 x 4 999 999	6 957	15 310	19 734	30 121
8	12 x 4 999 999	25 054	36 744	82 618	30 121
9	0	1 498	3 062	21 138	30 121

Tabulka 7: Doba zpracování dotazů srovnávaných systémů [v ms].

5.3 Výsledek testu

V grafu 19 můžeme vidět detailní rozpad doby zpracování dotazů obou systémů. Testy v tomto případě potvrdily předpoklad kratší doby zpracování u sloupcového systému. Rozdíl mezi sloupcovým a řádkovým pojetím se projevil nejvíce v oblasti diskových pří-

stupů (jak můžeme vidět na srovnání v grafu 19).

Průměrná hodnota počtu přístupů je asi čtvrtinová ve prospěch sloupcového pojetí. V případě projekce (1 – 4 atributy) byla doba vyhodnocení dotazů asi čtyřikrát rychlejší, než u řádkové databáze. Dokonce v případě výpisu všech (nebo i více) atributů tabulky je náš sloupcový databázový systém o několik řádů rychlejší, než implementace řádkového pojetí.

Potvrdilo se tedy, že v případě projekce může být změna pojetí ukládání dat přínosem. Můžeme předpokládat, že s rostoucím objemem dat bude časová úspora markantnější.

5.4 Srovnání implementací s komerčními systémy

Nyní srovnáme vlastní implementaci a existující řádkový systém se systémy HP Vertica a MS SQL Server 2012 Express. Jelikož používáme stejná data i dotazy, bude toto zajímavé srovnání nejvíce vypovídající.

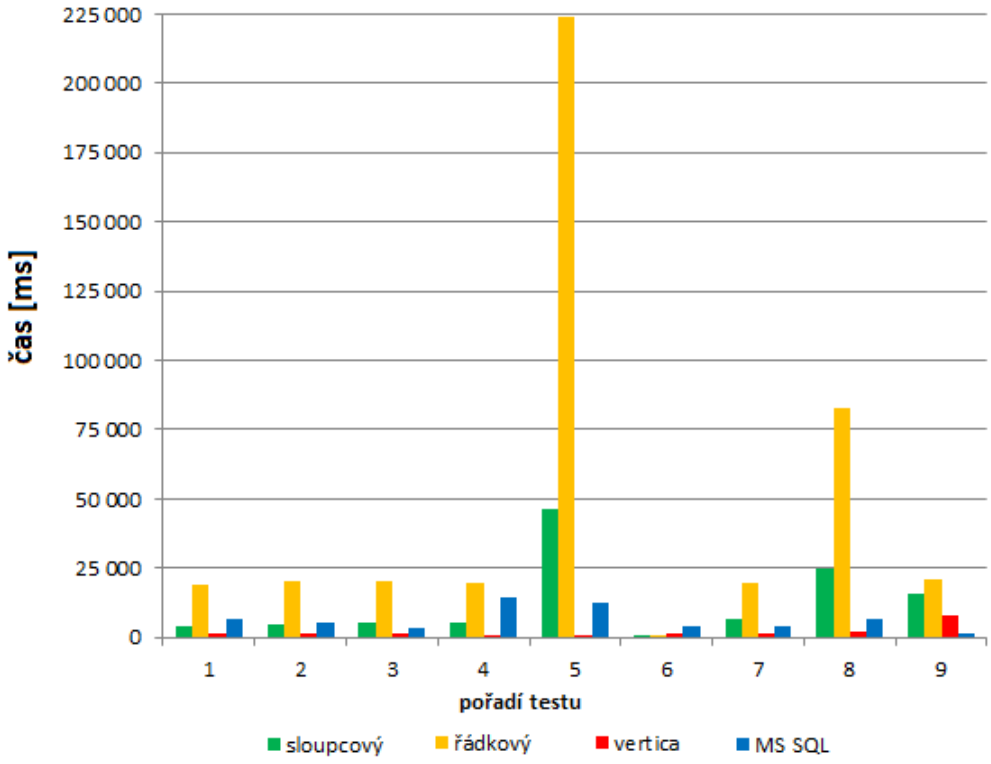
Pořadí testu	Počet výsledků	Sloupcový systém	Řádkový systém	HP Vertica	MS SQL 2012
1	3 933	4 056	19017	1282	6534
2	2 496 376	4 867	20545	1528	5672
3	16 827	5 242	20 499	1279	3691
4	1	5 444	19 515	316	14500
5	1	46 454	223 751	612	12511
6	1	187	78	32	3932
7	5 x 4 999 999	6 957	19 734	1406	4078
8	12 x 4 999 999	25 054	82 618	1812	6382
9	0	15 756	1 498	8241	1332
průměr	279682	12668	47432	1834	6515

Tabulka 8: Srovnání implementovaných a komerčních databázových systémů - doba zpracování [v ms].

Všechny naměřené hodnoty v tabulce 8 jsou uváděny v milisekundách [ms]. Výsledek ukazuje, že komerční řešení má značný náskok. Ač HP Vertica používá ve své image nejslabší konfiguraci, je nejvíce optimalizovaná a téměř všechny dotazy trvají nejkratší dobu, bez ohledu na typ dotazu. Kromě posledního testu, kterému je věnována pozornost v případě HP Vertica v předposlední kapitole, ovlivňovalo dobu výsledku spíše aktuální vytížení pevného disku testovacího počítače.

V případě MS SQL Serveru 2012 je většina dosažených časů nižší, než u obou im-

plementací. Určitě bude hrát roli to, že systémy jsou již samy o sobě do jisté míry optimalizované, proto v souboji s našimi „ve skrze jednoduchými systémy“ mají náskok. Přesto jsme v případě čtvrtého a šestého dotazu s oběma implementacemi dosáhli lepších výsledků a „porazili“ zástupce velmi rozšířeného SQL serveru.



Obrázek 21: Srovnání rychlosti vyhodnocení dotazů všech testovaných systémů.

Pokud se zaměříme na graf 21, můžeme snadno srovnat všechny systémy. Můžeme vidět již popsanou skutečnost, že implementace sloupcového systému drží s těmi komerčními krok.

V případě sloupcového řešení je doba zpracování většiny dotazů mezi 4 000 ms – 7 000 ms. U řádkového systému doba zpracování osciluje okolo hodnoty 20 000 ms. U HP Vertica je doba zpracování většiny dotazů mezi 1 000 ms – 2 000 ms. U MS SQL Server 2012 se pohybuje doba zpracování většiny dotazů mezi 3 000 ms – 7 000 ms.

6 HP Vertica a srovnání s MS SQL Server Express 2012

V této kapitole se budeme věnovat představení řešení sloupcové databáze firmy HP Vertica. Dále zmíníme výsledky testů z M.I.T., článku *One Size Fits All?* [23]. V poslední části provedeme test rychlosti vyhodnocení některých dotazů nad vzorovou databází vmartdb Vertica v porovnání s řádkovou databází v MS SQL Server Express 2012, do které byla nahrána stejná data, jako jsou ve vzorové databázi vmartdb Vertica a dané výsledky rozebereme.

6.1 Vertica

Vertica Systems je softwarová společnost zabývající se analytickou správou databází. Vertica byla založena v roce 2005 výzkumníkem databází Michael Stonebrakerem a Andrew Palmerem. Bývalí ředitelé firmy byli Ralph Breslauer a Christopher P. Lynch. 22.3.2011 získal Hewlett Packard společnost Vertica. Akvizicí společnosti Vertica rozšířila společnost HP své softwarové portfolio pro velké firmy a veřejný sektor.

Obrázek 22: Logo Vertica

Produkty

Sloupcově orientovaná platforma „Vertica Analytics Platform“ je určena pro správu rychle rostoucích objemů dat a poskytuje velmi úspěšné vyhledání dotazů při použití datových skladů a dalších náročných aplikací. V propagaci produktu je uvedeno, že výrazně zlepšuje vyhledávání dotazu ve srovnání s tradičními relačními databázovými systémy, poskytuje vysokou dostupnost a zařazení petabytů na komoditní podnikové servery.

Mezi jeho konstrukční vlastnosti patří:

- Sloupcově orientované paměťové uspořádání, které zvyšuje výkon sekvenčního přístupu k záznamům na úkor běžných transakčních operací, jako je například vyhledávání jednoho záznamu, aktualizace a odstraňování.
- Standardní SQL rozhraní s mnoha vestavěnými analytickými schopnostmi, jako jsou vyplňování časově sériových řad/interpolace. Akce vycházející z měření zaslaných dat a intervalů mezi návštěvou stránek, výběru vhodných vzorů, připojení řady vhodných akcí, statistických výpočtů (např. regresní analýza) a geoprostorové analýzy.
- Externí aktualizace a organizace hybridního uložiště, která zvyšují schopnost dotazování, vkládání a načítání, ale na úkor aktualizace a odstraňování.

- Komprese, která snižuje náklady na skladování a propustnost (vstupní/výstupní). Vysoká komprese je možná, protože sloupce homogenního datového typu jsou uloženy společně, a protože aktualizace na hlavním skladu jsou zpracovávány po dávkách.
- Nezávislá struktura (shared nothing architecture), která snižuje systémové neshody pro sdílené prostředky a umožňuje postupné snížení výkonu navzdory možnému selhání hardware.
- Snadné použití a údržba prostřednictvím automatizované replikace dat, serveru, optimalizace dotazů, a optimalizace skladování.
- Podpora pro standardní programovací rozhraní ODBC, JDBC a ADO.NET.

Specializovaný přístup Vertica si klade za cíl výrazně zvýšit výkon dotazů v datových skladech, a zároveň snížit celkové náklady na vlastnictví snížením hardwarových nároků. Jeden příklad případu použití je podrobně popsán ve výzkumné práci nazvané „One Size Fits All?“ [23] vykazující zlepšení výkonnosti v důsledku použití vertikálního přístupu DBMS .
Vertica Analytics Platform Community Edition, stejně jako verze 2011, je k dispozici zdarma s určitými omezeními, jako jsou: maximálně 1 terabyte „surových“ dat,
tři nody/uzly (rozložení zátěže maximálně na tři servery) a omezená podpora.

Optimalizace

Vertica Analytická databáze funguje síťově na linuxových komoditních serverech. Je také k dispozici jako hostovaná DBMS zajištěná a fungující na Amazon Elastic Compute Cloud. Tento produkt integruje s Hadoop za působení HDFS v rámci Vertica a poskytuje přístup k datům Vertica prostřednictvím MapReduce.
MicroStrategy busines inteligenční platforma je optimalizována pro databázi Vertica přes Vertica specifické syntaxe SQL.
Několik funkcí Vertica bylo původně prototypy v rámci sloupcově orientované databáze C-Store – akademický výzkumný projekt MIT otevřených zdrojů a dalších vysokých škol. Architektura systému je popsána v dokumentu VLDB 2012.

HAVEn

V červnu 2013 Společnost HP představila HAVEn platformu určenou pro analýzu a hledání informací (a jejich významu) z velkých dat - petabytů strukturovaných a nestrukturovaných informací. HAVEn si také klade za cíl identifikovat informace, které nejsou nutné a mohou být umístěny v nízkonákladovém skladu nebo dokonce ve výpisu struktury tabulky. HAVEn používá open source Hadoop s technologií HP získané z Autonomy Corporation, Vertica a ArcSight.

Důležité firemní události

- V lednu 2008 – Sybase podal žalobu na porušování patentových práv proti Vertica.
- V lednu 2010 – Vertica zvítězila v předběžném slyšení.
- V červnu 2010 – Sybase a Vertica vyřešily žalobu u soudu, kterým byly zamítnuty veškeré pohledávky pro nesplnění povinnosti.
- V červnu 2013 – Hewlett-Packard oznámila, že Vertica bude klíčovou součástí její velké datové iniciativy s názvem HAVEn (Hadoop, Autonomy IDOL, Vertica, Enterprise Security, a n Apps).
- V srpnu 2013 – HP Vertica pořádala svoji první konferenci „Big data – velká data“ o událostech v Bostonu, MA USA.

Pod vedením Colina Mahony, Vertica sponzoruje různé technologické akce v databázovém průmyslu [22].

6.2 Srovnání výkonu řádkového a sloupcového pojetí

V předchozím textu jsme slíbili představit článek „One Size Fits All?“. V tomto článku se autoři zabývají rychlostí vyhodnocení dvanácti dotazů v řádkové databázi a těchto stejných dotazů v sloupcové databázi. Zkoumají také nároky na diskový prostor.

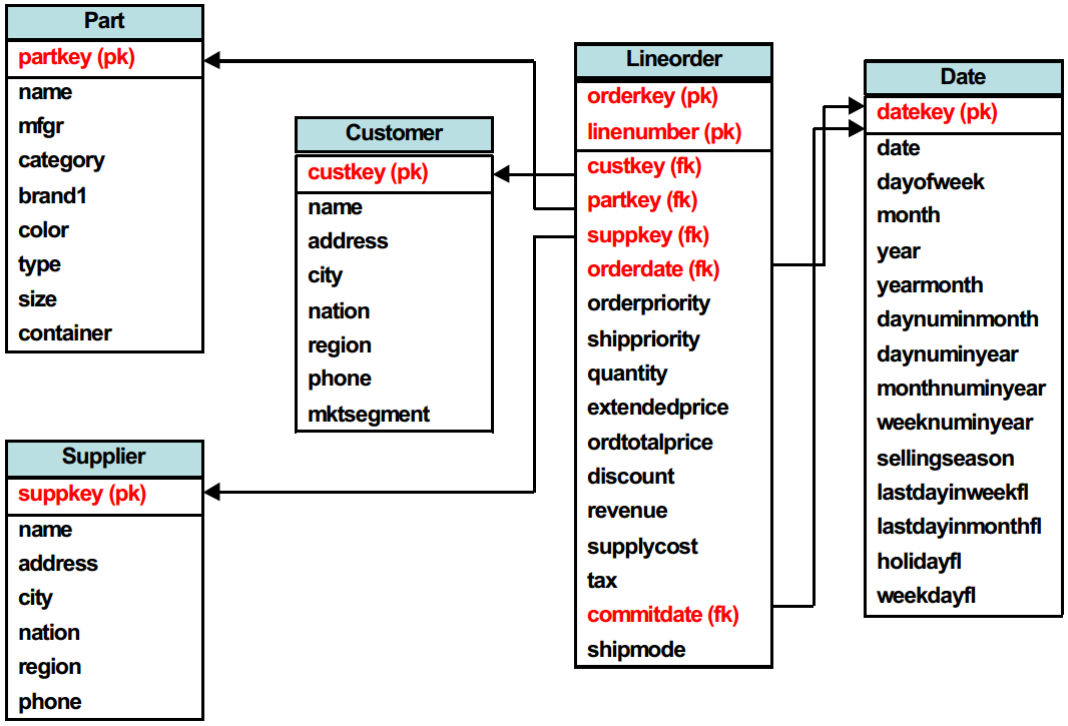
6.2.1 TPC-H

Velmi dobře známé měřítko výkonu TPC-H je využíváno mnoha prodejci, kteří prohlašují nadřazenost ve výkonu datového skladu. Tento ukazatel je chytře konstruován tak, aby se vyhnul použití schématu sněhové vločky a také k tomu, aby učinil uskutečněné náhledy neproduktivními. Při dotazování 2 tuctů CIO (Chief Information Officer), autoři nikdy neviděli datový sklad, který by nepoužíval schéma sněhové vločky.

Hence, Pat O'Neil zjednodušil TPC-H schéma 23 do podoby sněhové vločky a zcela vymezil 12 variant TPC-H dotazů týkajících se tohoto tématu. Tři vybrané dotazy si zde představíme. První vypíše celkovou slevu v roce 1993, pro položky nad 25 ks a slevou 1 až 3 % (výpis 4).

```
SELECT SUM (lo_extendedprice*lo_discount) AS revenue
FROM lineorder, dwdate
WHERE lo_orderdate = d_datekey
    AND d_year = 1993
    AND lo_discount between 1 and 3
    AND lo_quantity < 25;
```

Výpis 4: První dotaz - Query 1.



Obrázek 23: TPC-H - použité schéma [23]

Druhým představeným dotazem je Query 5 – celkově pátý z dvanácti dotazů. Dotaz vypíše celkové příjmy pro určitou značku a rok z regionu Asia (výpis 5).

```
SELECT SUM (lo_revenue), d_year, p_brand1
FROM lineorder, dwdate, part, supplier
WHERE lo_orderdate = d_datekey
      AND lo_partkey = p_partkey
      AND lo_suppkey = s_suppkey
      AND p_brand1 between 'MFGR#2221' and 'MFGR#2228'
      AND s_region = 'ASIA'
GROUP BY d_year, p_brand1
ORDER BY d_year, p_brand1
```

Výpis 5: Pátý dotaz - Query 5.

Posledním představeným dotazem, který je označený jako Query 8 – celkově osmý dotaz z dvanácti dotazů. Výsledkem je součet příjmů od zákazníků stejné zeměpisné oblasti jako jejich dodavatelé, dále v členění podle geografie (United States) a roku (výpis 6).

```
SELECT c_city, s_city, d_year, sum(lo_revenue) as revenue
FROM customer, lineorder, supplier, dwdate
WHERE lo_custkey = c_custkey
  AND lo_suppkey = s_suppkey
  AND lo_orderdate = d_datekey
  AND c_nation = 'UNITED␣STATES'
  AND s_nation = 'UNITED␣STATES'
  AND d_year between 1992 and 1997
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, revenue desc;
```

Výpis 6: Osmý dotaz - Query 8.

Výsledek testu je uveden v přiložené tabulce 24.

	Row Store Low Space	*Column Store Low Space*	*Row Store High Space*	*Column Store High Space*
Query 1	32.0	3.4	3.9	1.1
Query 2	31.6	4.1	1.6	0.3
Query 3	30.8	2.8	0.9	0.2
Query 4	29.3	3.4	7.2	0.6
Query 5	26.1	3.2	2.1	0.2
Query 6	22.2	3.0	0.7	0.1
Query 7	60.9	1.7	15.6	2.4
Query 8	4.1	3.2	2.5	0.3
Query 9	3.7	3.0	2.0	0.2
Query 10	24.1	1.1	11.4	1.8
Query 11	5.1	0.2	6.3	0.3
Query 12	0.7	0.2	1.0	0.2
Weighted Average	13.6	1.8	2.9	0.4
Space Required	60.3	36.3	76.7	40.2

Obrázek 24: Výsledek testu - hodnoty časové (v sekundách) a v řádku Space Required je požadavek na místo (v GB) [23]

Z tabulky můžeme vyčíst, že vždy bylo vhodné použít pro tyto dotazy sloupcově orientovaný databázový systém. Jeho rychlost byla ve všech případech několikanásobně rychlejší, než při použití populárního řádkově orientovaného databázového systému [23].

6.3 Příprava testovacího prostředí HP Vertica

Příprava testovacího prostředí HP Vertica je velmi jednoduchá. Po registraci na webu https://my.vertica.com/ dostaneme třicetidenní přístup do download sekce. Zde si stačí vybrat předkonfigurovaný image virtuálního stroje pro VMware Workstation. Image *HP Vertica Analytic Database Server* je nakonfigurován pro dnešní běžné pracovní stanice.

Pro virtuální stroj je vyhrazen 1 procesor (1 jádro), 1 GB operační paměti a 50 GB diskového prostoru. Po stažení cca 1,5 GB image stačí virtuální stroj pouze spustit 25.

Obrázek 25: Spouštění image HP Vertica Analytic Database Serveru

Po spuštění virtuálního stroje se přihlásíme uživatelským jménem *dbadmin* a heslem *password*, které je nastaveno pro všechny uživatele včetně „super usera“.
Jakmile jsme úspěšně přihlášeni, spustíme *Terminal* a v něm vytvoříme vzorovou databázi následujícími kroky:

- Přepneme se do adresáře

  ```
  \opt\vertica\examples\VMart_Schema
  ```

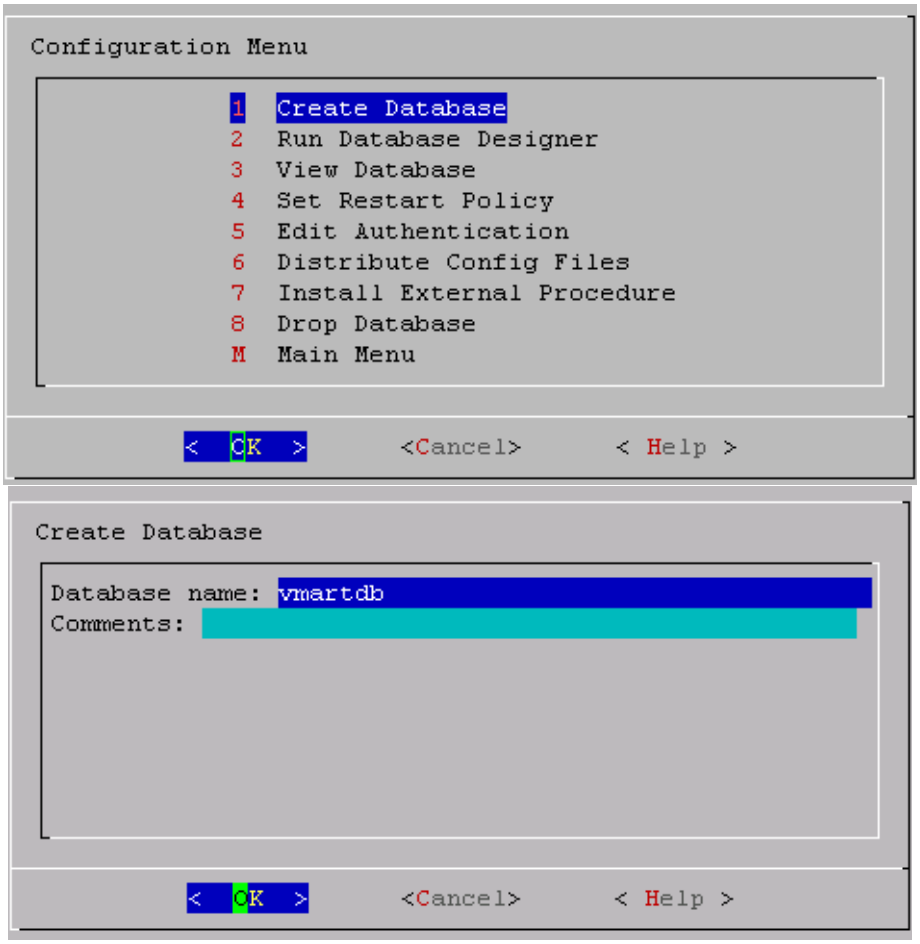
- Spustíme příkaz

  ```
  $ ./vmart_gen
  ```

- Po vygenerování záznamů si spustíme nástroj *admintools* zadáním příkazu

  ```
  $ admintools
  ```

- V admintools vybereme položku *Create database* a v dalším okně zvolíme název *vmartdb* (viz. obrázek 26)
- Dále zvolíme heslo pro databázi (můžeme nechat prázdné) a v následujícím okně vybereme z nabídky *Hosts:* nabízený 127.0.0.1



Obrázek 26: Vytvoření vzorové databáze vmartdb

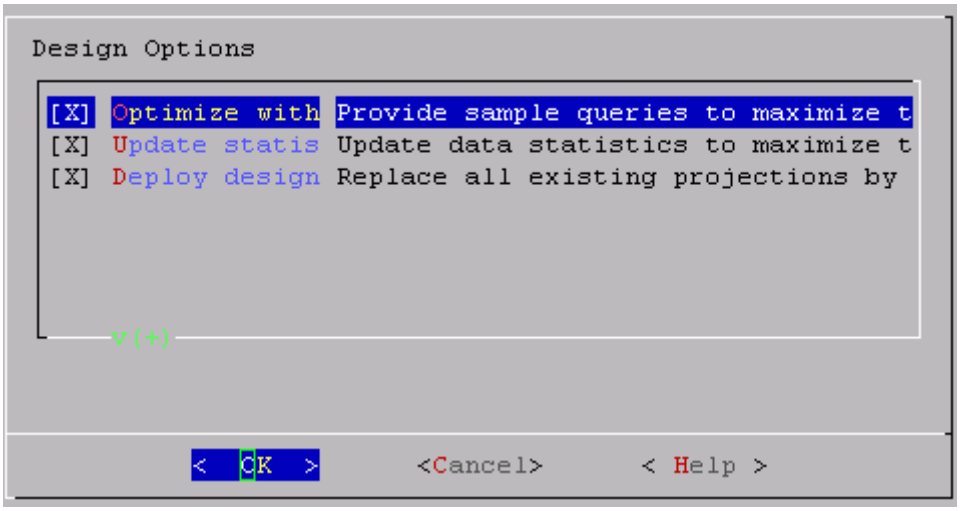
- Celé vytvoření v následujícím okně potvrdíme tlačítkem *YES* v dialogu *Create the database?* a databáze se vytvoří
- V admintools zvolíme položku *Connect to Database* a zde můžeme zadávat příkazy. Následujícím příkazem vytvoříme v databázi logické schéma

```
vmartdb=> \i vmart_define_schema.sql
```

- Do tohoto schématu nahrajeme data příkazem

```
vmartdb=> \i vmart_load_data.sql
```

- Nyní si vytvoříme komplexní návrh. Vrátíme se do hlavního menu admintools, zvolíme *Configuration Menu* a v něm *Run Database Designer*. Zde označíme všechny volby (obrázek 27) Následně budeme v okně *Optimize with queries* vyzváni k zadání úplné cesty k souboru s dotazy, aby byl systém optimalizován.



Obrázek 27: Vytvoření komplexního návrhu

- V dalším okně budeme dotázáni na způsob návrhu – vybereme *Balanced* – ten vytvoří vyvážený návrh mezi velikostí databáze a rychlostí vyhodnocení dotazu. Následně optimalizaci potvrdíme stiskem tlačítka *Proceed*
- Jakmile je optimalizace dokončena, přejdeme do hlavního menu admintools a zvolíme *Connect to Database*
- Nyní můžeme spouštět předpřipravené dotazy. Před jejich spouštěním si zapneme měření času vyhodnocení dotazu. To provedeme příkazem

```
\timing
```

- Jednoduchými příkazy nyní dokážeme vyhodnotit dotazy

```
vmartdb=> \i vmart_query_01.sql
```

Spustili jsme sadu testů nad sloupcovou databází, kterou provedeme také u zástupce řádkového pojetí. Celkem jsme provedli devět dotazů a měřili čas jejich vyhodnocení.

6.4 Příprava testovacího prostředí s Microsoft SQL Server 2012

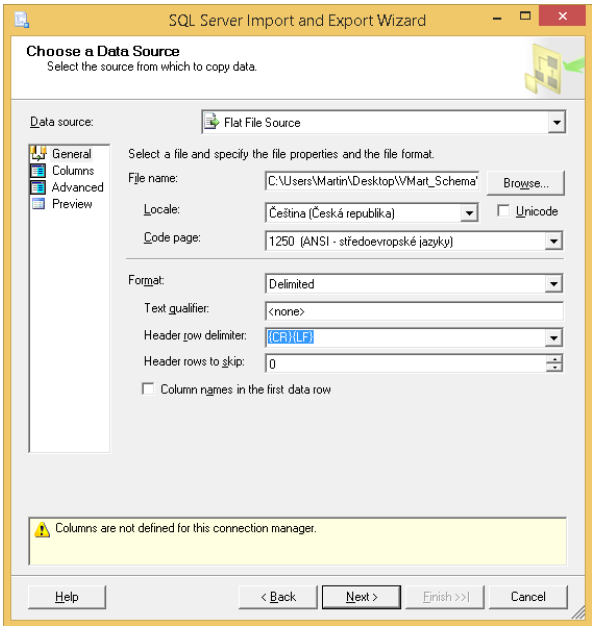
Pro testování tradičního zástupce řádkových databází zvolíme MS SQL Server 2012 Express, které je na stránkách společnosti Microsoft ke stažení zdarma. K verzi Express je možné stáhnout také MS SQL Server Management Studio 2012, které využijeme pro celý proces testů. Na virtuální počítač bylo nutné nainstalovat operační systém (zvolili jsme MS Windows 8.1 Professional x64). Z tohoto důvodu byl virtuální stroj nastaven s jinými parametry, než virtuální stroj s HP Vertica. Konfigurace pro testování MS SQL Serveru byla 2 procesorová jádra, 2 GB paměti, 60 GB místa na pevném disku.

Jelikož by bylo vhodné testovat na stejných datech, která byla k dispozici v rámci testovací databáze vmartdb u HP Vertica, bylo nutné pomocí skriptu *vmart_define_schema.sql* vytvořit tabulky v databázi. Po úspěšném vytvoření tabulek bylo nezbytné naimportovat data do příslušných tabulek. Zdrojová data jsme si stáhli z virtuálního stroje HP Vertica, která se nacházela v adresáři

```
\opt\vertica\examples\VMart_Schema
```

a naimportovali je podle následujícího návodu.

Na vytvořené databázi v MS SQL Management Studio klikneme pravým tlačítkem, zvolíme „Tasks“ a v následném podmenu zvolíme položku „Import Data…“. Otevře se průvodce pro import a export dat. Jako zdroj dat vybereme položku „Flat File Source“, jak ukazuje obrázek 28.



Obrázek 28: Import souboru s daty do tabulky MS SQL Serveru 2012

Zde označíme položku *„Column names in the first data row“*, jelikož **.tbl* soubory obsahují pouze surová data. V levém menu si vybereme „Columns“ a zkontrolujeme výběr oddělovače „|“, ve stejném menu následně vybereme „Advanced“ a zkontrolujeme nebo opravíme datové typy sloupců, aby vyhovovaly definici tabulek a bylo možné data naimportovat. Poté již stiskneme tlačítko „Next“, vybereme cílovou tabulku a můžeme zvolit tlačítko „Finish“, čímž se zahájí samotný import.

Protože importujeme velké množství dat, je import časově náročný a musíme se na to připravit. Po naimportování všech patřičných dat do příslušných tabulek zkontrolujeme data v každé z nich. Důležitá je pro nás jak struktura dat, tak také jejich počet. V našem případě proběhlo vše v pořádku a po několika desítkách minut importů máme databázi připravenou ke spuštění sady testů, kterou jsme předtím provedli na sloupcově orientované databázi. Samotné vykonání testů zabralo jen zlomek času nutného pro přípravu prostředí a dat.

6.5 Výsledky porovnání komerčního sloupcového a řádkového databázového systému

S přihlédnutím k nastudované teorii a k výsledkům zkoumání ve vlastním sloupcovém databázovém systému jsme se domnívali, že jedni z hlavních představitelů každého z řešení potvrdí fakt, že pro projekci určitých atributů je evidentně vhodnější použití sloupcového řešení.

Při porovnání sady testů nad sloupcovým řešením a nad řádkovým řešením vyšel skoro ve všech případech lépe sloupcový systém. Testy prokázaly, že sloupcový databázový systém je pro vhodná data asi 8krát rychlejší, než tradiční řádkový databázový systém.

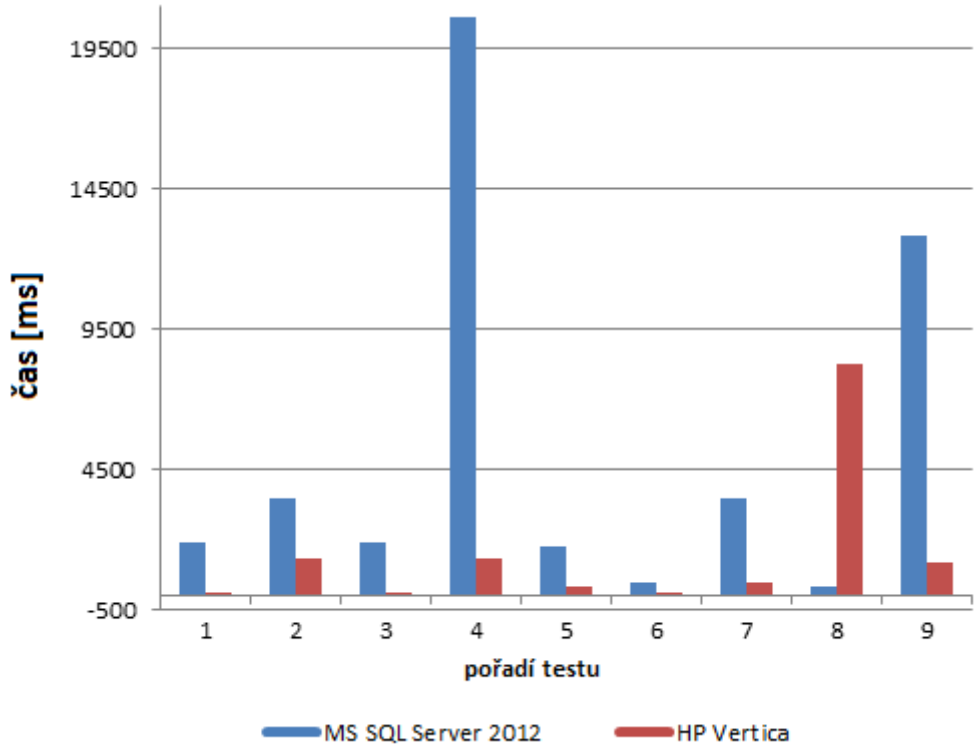
Výsledky testů uvádíme v tabulce 9.

Naměřené údaje z tabulky 9 byly podrobeny další analýze. Ze získaných hodnot byl vytvořen sloupcový graf, který ukazuje vyšší rychlost vyhodnocení většiny dotazů zástupcem sloupcového řešení.
Zajímavostí je skutečnost, kdy pro vyhodnocovaný dotaz systém nenajde žádný výsledek. V tomto případě byl tradiční databázový systém daleko rychlejší, nežli zástupce sloupcového řešení. Zmiňovaný graf najdeme na obrázku 29.

Z grafu můžeme usoudit, že vyhodnocování dotazů trvá u řádkového řešení v rozmezí 2000 ms až 3600 ms. U sloupcového řešení je rozmezí 400 ms až 1500 ms. Ač je počet měření relativně nízký, můžeme z provedených testů potvrdit, že pro specifické dotazy s projekcí vybraných atributů, je v případě sloupcového systému vyhodnocení dotazů několikanásobně rychlejší, než u tradičních DBMS.

Pořadí testu	Počet výsledků	MS SQL Server	HP Vertica
1	5	1914,117	24,928
2	348	3453,192	1277,97
3	2	1861,626	54,923
4	1000	20601,240	1303,517
5	474	1750,931	316,346
6	9	422,591	58,266
7	204	3483,419	436,748
8	0	315,898	8240,253
9	1000	12859,690	1153,838
průměr	338	5184,745	1429,643

Tabulka 9: Čas vyhodnocení dotazu [v ms]



Obrázek 29: Graf srovnání rychlosti

Použité schéma a vyhodnocované dotazy nad databázemi jsou obsaženy v příloze na DVD.

7 Závěr

Jak je vidět, tak sloupcové databáze přinášejí změnu v pohledu na databáze a snaží se řešit některé výše popsané problémy. Většina z nich je ve stádiu raného vývoje a ještě bude zajímavé sledovat jejich další rozvoj a vylepšení.

V práci jsme představili vlastní implementaci sloupcového systému, kterou jsme porovnali jak s existující řádkovou implementací, tak i s komerčními produkty. Implementovaný systém se blíží výkonu komerčního HP Vertica. V případě další profilace kódu a následných úprav by systém mohl dosahovat srovnatelných hodnot.

Ač projekt „cColumnStore“ v porovnání s HP Vertica lehce „ztrácí“, dosáhl celkově velmi zajímavých výsledků. Ve všech případech předčil v době odezvy představitele implementace řádkové databáze, dokonce v několika případech bylo vyhodnocení rychlejší, než v případě komerčního databázového systému. V oblasti diskových přístupů vykázal implementovaný systém nadějné výsledky, které potvrdily předpokladáné výhody sloupcových systému, nastudované při psaní této práce.

Provedené testy, které byly popsané v této práci potvrdily, že na poli databázových technologií mají sloupcové databázové systémy zaslouženě své místo a určitě najdou mnohá uplatnění. Ať již schopnými analytiky, kteří navrhnou levné řešení datového skladu společnosti, tak také pro velké „hráče na trhu“ ve vývoji podnikového software. Právě v této oblasti můžeme předpokládat největší možný rozmach sloupcově orientovaných databázových systémů.

Budoucnost vývoje software, ať již úrovně velkých podnikových systémů, tak i menších aplikací pro malé firmy a živnostníky, možná zažije další změnu v přístupu k vývoji software. Můžeme se domnívat, že hlavně u velkých řešení bude docházet k rozdělení dat, některá budou ukládána v dnes tradičních řešeních, jiná data sloužící k dlouhodobým analýzám pro některá oddělení budou využívat sloupcové databázové systémy.

Jak vidíme, vývoj sytémů a řešení dnes směřuje do oblasti, kde se prodávají kompletní řešení, jak hardwarové, tak softwarové. S tímto se dodává veškerá podpora a tak v rámci firemního tajemství ani netušíme, zda není tato technologie již využita, například v některém z modulů nového řešení společnosti SAP.

Ze své dosavadní praxe mohu soudit také to, že osvěta a dosažitelnost sloupcových databází bude největší překážkou jejich rozmachu. Jsem si vědom, že má praxe z bankovnictví, které je co se ve striktnosti a množství bezpečnostních pravidel jednou z nejstřeženějších oblastí, může být zkreslená. Mé zkušenosti však jsou takové, že pro vytvoření datového skladu pro interní vývoj určitého oddělení bývají poskytována často nevhodná řešení.

Nezbývá nám nic jiného, než se těšit na dobu, kdy bude možné i v této komerční

sféře použít sloupcové databázové systémy a ušetřit spoustu času a svému zaměstnavateli nemalé finanční náklady. Bude to doba, kdy se sloupcové databázové systémy stanou standardem. Zda to bude v řádu let, nebo tato doba nenastane, nyní nedokážeme odhadnout.

Čtenář, který si tuto práci přečte za několik let, již možná bude znát odpověď na otázku, kterou si nyní můžeme pouze položit: „Stanou se sloupcové systémy standardem pro vytváření datových skladů, nebo se bude vývoj v oblasti databází ubírat jiným směrem?“

8 Reference

[1] GARCIA-MOLINA, Jeffrey D ULLMAN, Hector a WIDOM, Jennifer. *Database Systems: The Complete Book. 2nd ed*. New Jersey: Prentice Hall, 2002, 1119 s. ISBN 01-303-1995-3.

[2] OLAP Council. *OLAP Counci – About OLAP*. [cit. 2013-03-19]. Dostupný z WWW: <http://www.olapcouncil.org>.

[3] CALDERS, Toon. *Data Warehousing and OLAP: MOLAP and ROLAP*. [online]. [cit. 2013-03-19]. Dostupný z WWW: <http://wwwis.win.tue.nl/ tcalders/teaching/advancedDB08/slides/OLAP3.pdf>.

[4] MicroStrategy, Incorporated. *The Case for Relational OLAP*. [online]. [cit. 2013-03-19]. Dostupný z WWW:<http://www.cs.bgu.ac.il/%7eonap052/uploads/Seminar/Relational%20OLAP%20Microstrategy.pdf>.

[5] *Online transaction processing* Wikipedia [online]. [cit. 2013-03-19]. Dostupný z WWW: < http://en.wikipedia.org/wiki/OLTP>.

[6] *Přehled technologie OLAP (Online Analytical Processing)* Microsoft [online]. c2013 [cit. 2013-03-19]. Dostupný z WWW: < http://office.microsoft.com/cs-cz/excel-help/prehled-technologie-olap-online-analytical-processing-HP010177437.aspx>.

[7] Danel, Roman. *Datový sklad* [online]. 2010 [cit. 2013-03-20]. VŠB - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. Dostupné z WWW: <http://homel.vsb.cz/ dan11/is_skripta/IS%202010%20-%20Danel%20-%20Datovy%20sklad.pdf>.

[8] Oracle Corporation. *Hardware and Software. Engineered to Work Together*. [online]. [cit. 2013-03-20].Dostupný z WWW: <http://www.oracle.com/us/sun/index.htm>.

[9] *Datové sklady a OLAP* [online]. c2002 [cit. 2013-03-20].
Dostupný z WWW: < http://datamining.xf.cz>.

[10] POKORNÝ, Jaroslav. *Relační model datl* [online]. c2013 [cit. 2013-07-23]. . Dostupný z WWW: < http://www.ksi.mff.cuni.cz/∼pokorny/vyuka/srbd/rmd/>.

[11] SÁGHY, Tomáš. *Modelování entit s dynamickými atributy v relačních databázích*. [2010] [cit. 2013-08-19]. Diplomová práce. Masarykova Univerzita Brno, Fakulta Informatiky.

[12] *SAP Match Insights* [online]. [cit. 2014-07-14]. Dostupný z WWW: <http://sapsponsorships.com/m-news/412-germany-world-cup-final>.

[13] KYJONKA, Vladimír. *Vertikální uládání dat* [online]. Praha: Sybase Software s.r.o., 2004. [cit. 2013-03-19]. Dostupné z WWW: <http://si.vse.cz/archive/proceedings/2004/vertikalni-ukladani-dat.pdf>.

[14] *Nejžhavější trendy v databázích* BusinessIT [online]. 2011 [cit. 2013-09-12]. Dostupný z WWW: <http://www.businessit.cz/cz/nejzhavejsi-trendy-databaze-in-memory-cloud.php>.

[15] JOHNSON, Sam. *Cloud_computing.svg* [online]. [cit. 2013-09-11]. Dostupný z WWW: <http://commons.wikimedia.org/wiki/File:Cloud_computing.svg>.

[16] *Oracle dohání SAP a má databázi běžící v operační paměti* Connect [online]. 24.9.2013 [cit. 2014-01-12]. Dostupný z WWW: <http://connect.zive.cz/clanky/oracle-dohani-sap-a-ma-databazi-bezici-v-operacni-pameti/sc-320-a-170662>.

[17] ABADI, Daniel J. *Column-Stores For Wide and Sparse data* [online]. [cit. 2014-01-19]. Dostupný z WWW: < http://db.csail.mit.edu/projects/cstore/abadicidr07.pdf>.

[18] *Why Should I Check Out a Column Database?* [online]. [cit. 2014-01-28]. Dostupný z WWW: <http://infinidb.org/resources/tech-articles/174-why-should-i-check-out-a-column-database>.

[19] STONEBRAKER, Michael,*C-Store - A Column-Oriented DBMS*, VLDB '05. VLDB Endowment 553–564.

[20] HARIZOPOULOS, Stavros, ABADI, Daniel, BONCZ, Peter. *Column-Oriented Database Systems*, VLB 2009 Tutorial, 2009.

[21] *Resource Description Framework* W3C.org [online]. [cit. 2014-01-30]. Dostupný z WWW: <http://www.w3.org/RDF/>.

[22] *Vertica white papers* [online]. [cit. 2014-04-19]. Dostupný z WWW: <http://www.vertica.com/resources/white-papers/>.

[23] STONEBRAKER, Michael, BEAR, Chuck, ÇETINTEMEL, Uğur, CHERNIACK, Mitch, GE, Tingjian, HACHEM, Nabil, HARIZOPOULOS, Stavros, LIFTER, John, ROGERS, Jennie, ZDONIK, Stan. *One Size Fits All?* [online]. [cit. 2014-04-19]. Dostupný z WWW: <http://nms.csail.mit.edu/ stavros/pubs/osfa.pdf>.

A Obsah DVD

/root/		kořenový adresář
	abstraktCS.txt	abstrakt v češtině
	abstraktEN.txt	abstrakt v angličtině
	DiplomovaPrace.pdf	diplomová práce
	Vmart_schema.zip	archiv s databázovými skripty
	/Apl/	zdrojové kódy vlastní implementace
	/Testy/	soubory výsledků testů