Софийски Университет “Св. Климент Охридски”

Факултет по Математика и Информатика

Катедра “Софтуерни Технологии”

Дипломна Работа

НА ТЕМА:

“Ефективни бизнес анализи чрез използване на класически Data Warehouse технологии и асоциативен In-Memory OLAP”

Дипломант: Димитър Даринчев Димитров, Фак.№ M21481

Специалност: Информатика, Магистърска Програма “Електронен Бизнес”

Дипломен Ръководител: Доц. Петко Русков

София, 2007

Съдържание

Съдържание ..... - 2 -

Увод..... - 4 -

Business Intelligence. Компоненти, архитектура и подходи за внедряване. Предизвикателства пред аналитичните среди..... - 6 -

Business Intelligence - дефиниция ..... - 6 -

Business Intelligence – архитектура и компоненти..... - 13 -

ETL среда..... - 14 -

Data Warehouse среда ..... - 15 -

Аналитична среда – OLAP ..... - 18 -

Практически пример за интегриране на Data Warehouse и In-Memory OLAP – възможност за подобряване на Business Intelligence имплементациите?..... - 23 -

Бизнес сценарии ..... - 23 -

Business Intelligence подход за решаване на проблема ..... - 25 -

Техническа реализация ..... - 25 -

Логически дизайн на корпоративния Data Warehouse – Star Schema ..... - 27 -

Изграждане на ETL процес ..... - 35 -

Създаване на In-Memory OLAP среда и интеграция с корпоративният Data Warehouse. Изграждане на потребителска функционалност. ..... - 39 -

Заключение ..... - 48 -

Литература ..... - 51 -

Приложение ..... - 52 -

Увод

Съвременната бизнес среда е силно конкретна, глобализацията предоставя на клиентите по-големи възможности за избор, очакват се по-високи нива на обслужване, качество и персонализация. Компаниите за да оцелеят и развиват бизнеса си, се нуждаят от детайлна и точна информация за техните партньори, клиенти и доставчици. Също така е необходимо да спазват и отговарят на нарастващ брой законови, регулаторни и одит изисквания.

Всички тези фактори имплицират нарастващи изисквания по отношение на работата с информация и респективно имплементиране и използване на информационни системи.

Голяма част от компаниите вече успешно са внедрили в своите бизнес процеси станалите класически вече информационни системи за обработка на транзакционна информация в реално време (OLTP), като ERP, SCM, CRM и т.н. Оформящи дейността на компаниите, тези системи поддържат ежедневната оперативна дейност на бизнеса в целият му основен цикъл от логистика до финанси и от управление на човешките ресурси до работата с клиенти.

От информационна гледна точка OLTP решенията са често силно фрагментирани и генерират големи обеми от данни ежедневно. Тези данни биват съхранявани в бази данни с публично недостъпен формат, където са изключително трудно за извличане и повторно анализиране. Така процеса по вземане на информирани бизнес решения за компаниите е изключително труден, като те са изправени пред предизвикателството да интегрират големи обеми от информация, записана в различен формат и често със съмнително

качество. Практиката показва, че не са редки случаите при които различни отдели от организацията ползвайки различни OLTP системи или различни модули на една система имат различни версия на истината за случващото се с бизнеса.

За да отговори на очакванията на модерния бизнес за лесен достъп и споделяна на точна информация, създаване на справки, отчети и анализи, прогнози и вземане на информирани бизнес решения като цяло, ИТ индустрията създаден понятието Business Intelligence. Подхода Business Intelligence и неговото ефективно използване чрез интеграцията му с последните разработки в областта на асоциативните бази данни са обект на настоящата работа.

Business Intelligence. Компоненти, архитектура и подходи за внедряване. Предизвикателства пред аналитичните среди.

Business Intelligence - дефиниция

Business Intelligence не е продукт или система.

Това е архитектура и съвкупност от интегрирани оперативни и подпомагащи вземането на решения приложения и СУБД, които предоставят на бизнеса лесен достъп до бизнес данните.

Business Intelligence включва процеси, средства и технологии необходими за превръщането на данните в информация, а информацията в знания и планове, които предполагат успешни и ефективни бизнес действия. [3]

Така дефиниран като концепция, Business Intelligence има за цел да:

- помага да вземаме по-добри решения и да го правим по-бързо.
- превръща огромните обеми от данни, които притежават днешните организации в ценно съдържание за бизнеса.
- позволява достъп, споделяне, обработка и анализ на данни от точните хора, в точното време и чрез предпочитания от тях начин.
- създава на единна версия за истината.

В този ред на мисли Business Intelligence е често разглеждан като “рафинерия за данни”, както е показано на фигура 1. [4]

Фигура 1.

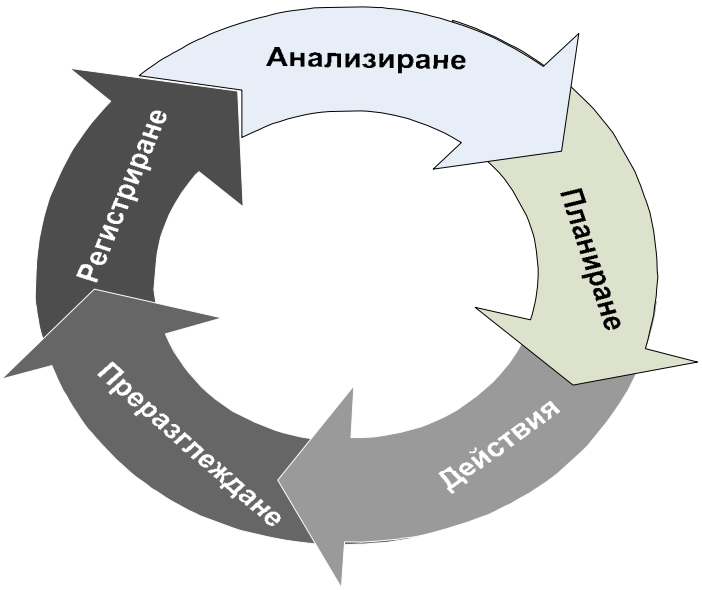
Business Intelligence архитектурата извлича оперативните данните от OLTP системите на организацията периодично, дневно, седмично или месечно и ги записва в собствена база данни наричана Склад за Данни (Data Warehouse). Една общоприета дефиниция за Data Warehouse е:

“Тематично ориентирана, интегрирана, време вариантна, не изменчива съвкупност от данни, подпомагаща вземането на решения”. [1]

Процеса по извличане на данните и зареждането им в Data Warehouse се среща в литературата като Извличане, Трансформиране и Зареждане на данни (Extraction, Transformation

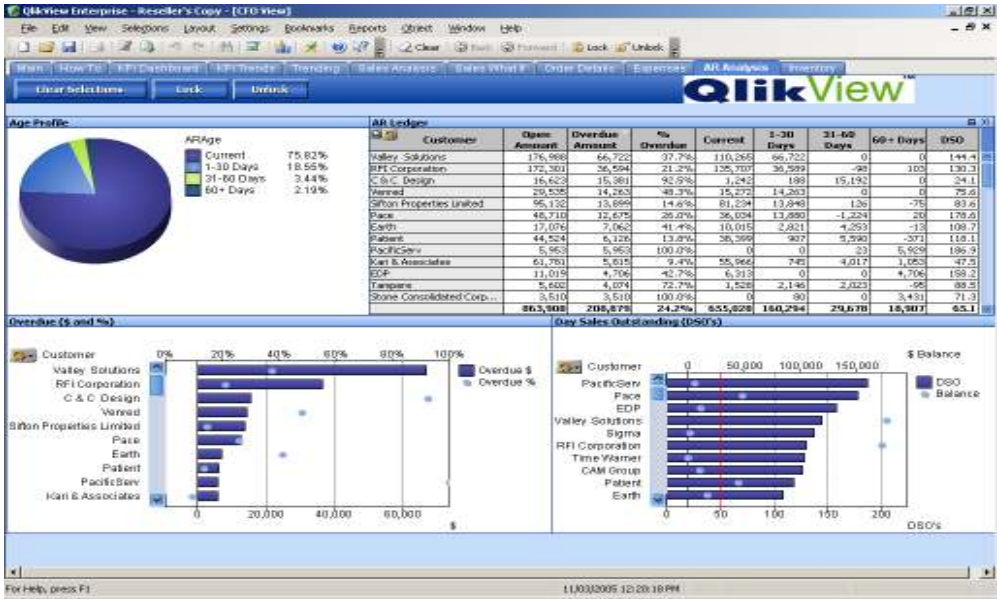
and Loading - ETL) или напоследък като Извличане, Консолидиране, Изчистване и Доставка (Extraction, Comforming, Cleaning and Delivery - ECCD) . [2]

Попадайки и записвани в Data Warehouse-a, данните придобиват контекст, т.е. се превръщат в информация. Тази информация бива анализирана от бизнеса посредством информационни средства за обработка на аналитична информация в реално време (On-line Analytical Processing - OLAP) и на базата на тези анализи придобиваме определени знания за пазара, средата и клиентите. Използвайки бизнес правилата и моделите, които са утвърдени в организацията и знанията ние достигаме до бизнес планове. Тези планове на свой ред реализираме използвайки нашия опит. Резултата от тези действия бива прихванат отново от оперативните системи под формата на продажби, доставки и други. Често този елегантен, но труден за реализиране цикъл е сравняван с цикълът по който човекът сам се учи, показан на фигура 2 [4]:



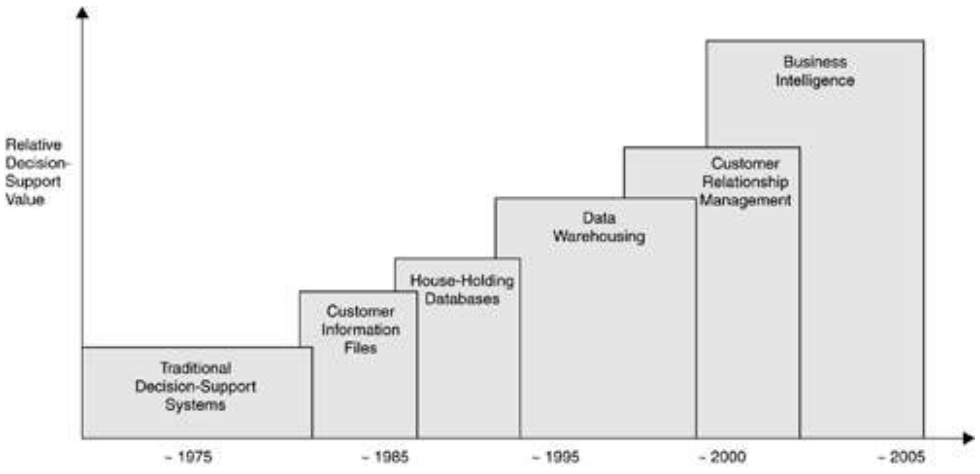
Фигура 2.

Симплистично погледнат крайният резултат е лесно достъпна аналитична среда предоставяща информация в разбираем и предпочитан от бизнеса формат. Пример за краен потребителски изглед от информационно приложение от областта на Business Intelligence е показан на фигура 3 – лесно достъпна среда за анализиране на бизнес информация.



Фигура 3.

Така специфицирана Business Intelligence като концепция се явява последната генерация на системите за подпомагане вземане на решения (Decision Support Systems – DSS), чието развитие стартира в средата на 70-години на 20 век е показано на фигура 4. [5]



Фигура 4.

За разлика от предходните разработки в областта на DSS, Business Intelligence предлага:

- обработване на силно нарасналите обеми от информация в наши дни.

- извличане на информация не само от конвенционални източници, но и от :
    - логове на уеб сървър (Web Server logs).
    - сървъри за управление на съдържание (Content Managment Severs).
    - уеб услуги (Web Services)
    - външни източници на информация, като статистика, маркетингови проучвания и други.

- нови и по-комплекси алгоритми за анализи, прогнозиране и предсказване.

- засилени потребителски изисквания в новата е-бизнес среда:
    - бързо доставяне на исканата информация.
    - разнообразни анализи и техники за визуализация на информация.
    - достъп до аналитична информация във всички вертикални нива на организацията.
    - доставяне на информация на различни устройства – факс, уеб, мобилни устройства, и т.н..

За да отговори на този широк спектър от очаквания Business Intelligence включва в себе си голям набор от съвременни и иновативни ИТ технологии, като:

- средства за управление на бази от данни: RDBMS, ODBMS, Data Warehousing, OLTP, OLAP, ETL, Data Mining.

- интернет стандарти и протоколи: WWW, HTTP, FTP, POP3, SMTP,WAP, SMS.

- програмни езици и средства: HTML, WML, XML/XSLT, JSP, ASP и други.

- Стандарти и протоколи за сигурност.

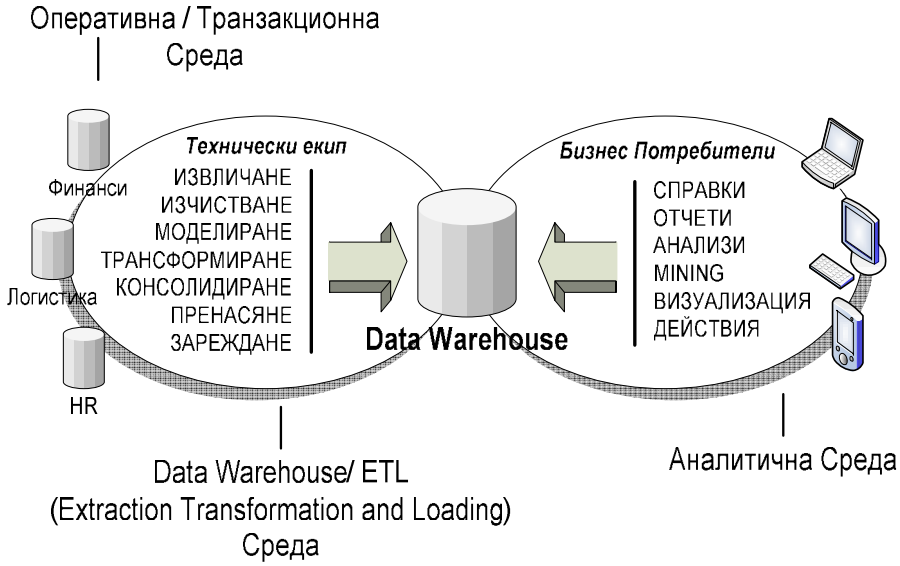
Business Intelligence – архитектура и компоненти

Разглеждана от техническата страна на информационните системи, така дефинираната Business Intelligence концепция, коренно се различава от класическите OLTP системи. Сравнителна таблица 1 показва принципните разлики между двата подхода: [3]

OLTP система	**Business Intelligence система**
Дизайн фокусиран към автоматизиране на процесите.	Дизайн фокусиран към подпомагане вземането на решения.
Работа с текущи данни в реално време.	Работа с периодично обновявани исторически данни.
Атомарно ниво на информацията	Атомарно и агрегирано ниво на информацията

Таблица 1.

На базата на тези особености класическата Business Intelligence архитектура е фокусирана към потока на информация от OLTP системите, през Data Warehouse среда към OLAP средата и е показана на фигура 5. [3]



Фигура 5.

Основните компоненти са както следва:

ETL среда

Това е координиран процес за извличане, изчистване, моделиране, трансформиране, консолидиране пренасяне и зареждане на данни. Работи периодично в така наречения Staging Window – времеви прозорец в който OLTP системите са налични за извличане на данни и това не затруднява тяхната работа (която може да бъде от критичен характер). Обикновено този времеви прозорец е извън работно време примерно през нощта. ETL средата извлича данни най-често на дневна база, като е възможно определена информация да бъде извличана месечно или годишно, зависимост от нейната релевантност – например годишно или месечно планиране. ETL технически работи в среда за временна обработка на информация (Staging Area), която може да включва база данни за съхраняване на информацията по време на трансформиране,

програмни модули за извличане, трансформиране, изчистване и пренасяне на информация. ETL процеса може да бъде разработен конкретно за нуждите на организацията или да се използва готов продукт, който да бъде настроен за конкретните изисквания.

Data Warehouse среда

Както вече беше дефиниран в началото Data Warehouse е релационна база от данни, която има за цел да съхранява всички транзакционни данни на организацията и да ги предоставя за аналитична обработка. В този смисъл Data Warehouse-а е ориентиран да запази пълната историческа информация на организацията, за разлика OLTP системите, които обикновено запазват информация за период от не повече от 2 години, а по-старата информация бива архивирана. Аналитичните изисквания пред Data Warehouse-а предполагат запазването както на информация на транзакционно ниво, така и на агрегирано такова по различни бизнес обекти съобразено с изискванията на бизнеса по отношения на справките, отчетите и анализите. Тези особености имплицират трудна техническа задача пред създаването на Data Warehouse-а.

Като основни техники за справянето с анализите на огромни обеми от информация е използването на денорманлизирани релационни модели, на логическата организацията на информацията, като: [2]

- Star Schema
- Snowflake Schema

и техники за физическото съхранение на данни в СУБД като: [5]

- Клъстери (Clustering - съхранение на свързани таблици и техните редове в последователен ред, което е съществено предимство при работа с големи обеми от исторически данни).

- Partitioning (физическо разделяне на големи по обем таблици на под-таблици, които биват записвани на множество дискове. Това позволява паралелна обработка на информацията) .

- Индексиране (Indexing - налага изграждане на стратегия за индексиране на често използвани колони, чрез различни техники като Bitmap и B-Tree индекси) .

- Паралелна обработка на заявки (дава изключително голямо предимство при обработка на големи обеми от данни, възползвайки се от техниките за клъстъринг, индексиране и partitioning)

- Техники за реорганизация.

- Създаване на резервни копия и възстановяване.

Основните разлики между OLTP ориентираните бази данни и Data Warehouse базите данни са обобщени на таблица 2: [3]

Транзакционна СУБД	Business Intelligence СУБД
Проектирана за автоматизиране на процесите, елиминиране на повторенията.	Проектирана за справки, отчети и анализи.
Очаквани времена за реакция под 1 секунда.	Очаквани времена за реакция в секунди, минути, часове.
Силно нормализирани за подържане на консистентно обновяване.	Силно денормализарани за подпомагане на извличането на големи обеми от информация.
Съхраняват актуални данни. Историческите данни се архивират.	Съхраняват огромни количества исторически данни на различни нива на агрегация.

Таблица 2.

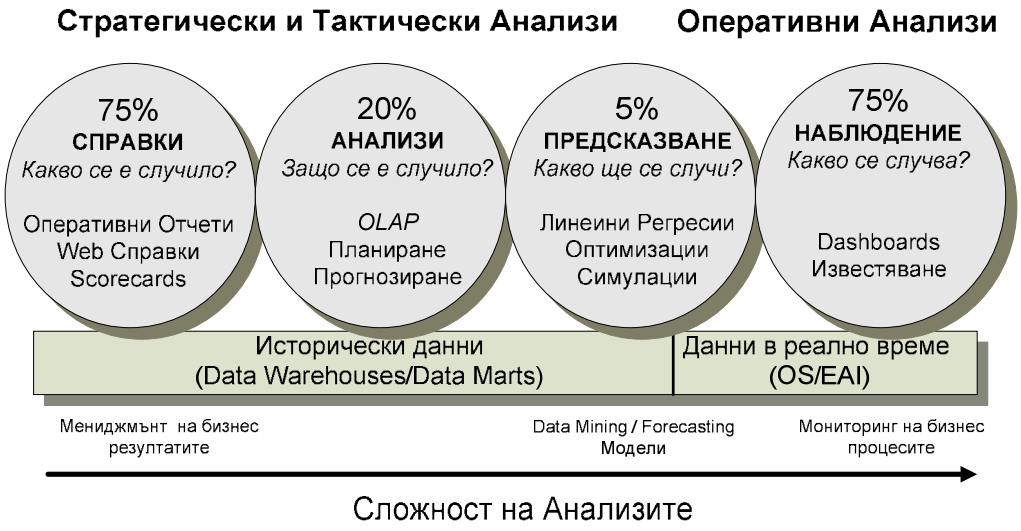
Всички съвременни СУБД предлагат възможности за изграждане на Data Warehouse, като някой са тясно специализирани в големи бази данни (Very Large Data Bases – VLDBs с обем по-голям от 1ТБ) като Oracle, TerraData, IBM DB2 и други.

Аналитична среда – OLAP

Аналитичната среда е програмен продукт, който има за цел да предостави средства за анализ на информацията съхранявана в Data Warehouse-а на бизнес потребителите. Както вече споменахме при дефиницията на Business Intelligence, изискванията към тази среда са високи – лесно достъпен интерфейс, терминология разбираема за бизнеса, бърз достъп и обработка на големи обеми от данни, функционалност като drilldown, sorting, pivot, ranking, техники от data mining, предсказващи анализи, доставяне на информацията в широк спектър от формати – MS Excel, уеб, PDF, графики и голям набор от устройства – персонални компютри, мобилни устройства, факс, и т.н.

Технически погледнато OLAP средата се явява транслатор от SQL езика на релационния Data Warehouse към бизнес езика на потребителите и обратно бизнес въпросите се превръщат в SQL заявки. В тази си същност OLAP средата е програмно средство за работа с мета-данни – данни за самите данни в Data Warehouse -а. Такива продукти обикновено са трудни за самостоятелно разработване и затова най-честия подход е закупуване на готов продукт и настройката му за конкретните бизнес очаквания.

От гледна точка на спектъра на работа на OLAP средата на фигура 6 са показани основните видове анализи, които съвременните организации очакват от подобни среди : [4]



Фигура 6.

За реализирането на OLAP среда към днешна дата са се обособили няколко подхода:

- ROLAP (Relational OLAP) – OLAP работещ с релационни данни, т.е. интегриращ се с релационни Data Warehouse решения. Този тип продукти позволяват изключително гъвкави анализи, като при различни запитвания от страна на потребителя, създават една или повече SQL заявки към Data Warehouse-а, извличат информацията и я представят в желания от потребителя вид. Това е най-разпространения вид OLAP, като водещи производители на такива продукти са Microstrategy, Business Objects, Cognos и други. Основен недостатък на ROLAP е трудната обработка на големи обеми от информация, също така в процеса на аналитичната работа се налага често генериране на SQL заявки, което води до

забавяне и изисква определено познания от страна на крайния потребител за логическата структурата на Data Warehouse-а – например е възможно определени обекти да не могат да бъдат комбинирани с други и това да води до тежки и трудно изпълними справки.

- MOLAP (Multi-dimensional OLAP) - многоизмерен MOLAP, фокусиран върху обработка на данни в бинарен формат, организиран под формата на обекти, наричани "кубове". При дефиниране на анализ в MOLAP среда, тя създава структура от комбинациите на всички участващи в анализа бизнес обекти, и за всяка комбинация предварително изчислява всички измерими страни на анализирания бизнес процес. Така получената структура се съхранява като обект в база данни и при искане на информация от страна на потребителя, готовите изчислени стойности се прочитат и предоставят. Това позволява изключително бързи анализи, на цената на гъвкавост и време за поддръжка. МОЛАП кубовете е необходимо да бъдат предварително дефинирани и изчислени, а при искане на потребител за промяна на анализа е необходима намесата на ИТ специалист, който да създаде нов куб и той да бъде изчислен. Тези недостатъци правят MOLAP предпочитан само за строго специфични области на анализи с големи обеми от информация (напр. телекомуникационния сектор) и той трудно може да отговори на очакванията на съвременния бизнес от среден мащаб. Основни производители на MOLAP продукти са Oracle, Panorama и други.

- HOLAP (Hybrid OLAP) – хибридна OLAP среда, която позволява използването както на релационни данни така и на MOLAP обекти от единен интерфейс, като позволява извличане на информация независимо от мястото и на съхранение прозрачно за потребителя. Недостатък на тези продукти е цената и трудното внедряване.

- In-Memory OLAP среди – това са най-иновативните представители на OLAP системите. Тяхната концепция е създаване на асоциативна база данни, която се разполага изцяло в оперативната памет на компютъра – било то персоналната машина на бизнес анализатора или споделен сървър, като извлича определени части от Data Warehouse-a , фокусирани към анализа на определен бизнес процес. Разположени по този начин, потребителя на продукта има възможност да анализира данните изключително бързо по всички налични характеристики и с богат набор от аналитична функционалност. In-Memory OLAP-а е изключително интерактивен, лесен за работа, бърз и богат на функционалност, като неговите възможности за обработка на обеми от информация са единствено ограничени от наличната оперативна памет. Съвременните продукти от този клас възползвайки от 64 битовото адресиране на данни в операционните системи, дават възможности за обработване на милиони записи от информация на настолни персонални машини. Същевременно асоциативния подход е близък до подхода на бизнес анализаторите и предполага интуитивна работа с минимално време за обучение. Възможностите за интеграция на класически Data Warehouse и In-Memory OLAP

дават много добри възможности за увеличаване на добавената стойност на Business Intelligence концепцията и намаляване на разходите и по тази причина интеграцията е обект на анализ на настоящата дипломна работа.

Практически пример за интегриране на Data Warehouse и In-Memory OLAP – възможност за подобряване на Business Intelligence имплементациите?

Интеграцията на класически Data Warehouse сценарии и In-Memory OLAP може да бъде изследвана сравнително лесно чрез бизнес пример, при който част от Data Warehouse фокусиран върху анализи на определен бизнес процес бива свързан и анализиран посредством In-Memory OLAP среда. По този начин бихме добили реална представа за степента на интеграцията, възможностите на една съвременна In-Memory OLAP среда да отговори на потребителските изисквания и да обработи сравнително големи обеми от данни.

Бизнес сценарии

За целта на изследването ще разгледаме следният сценарии: компанията "Altex" е фокусирана върху ретайл бизнеса с домакинска техника и притежава няколко вериги от магазини в Източна Европа. Една от веригите е от тип супермаркети, другите са от малък и среден тип работещи в централните градски части или големи търговски центрове. Всяка от веригите има собствена марка, но управлението е централизирано, като по този начин се оптимизират цикълът от доставки, ценовата и промоционалната политика и други. Като лидер в областта на търговията на дребно с бяла и черна техника, компанията работи с над 37000 продукта, в над 360 продуктови категории, които продава в 190 магазина. За да

оптимизира своята оперативна дейност, организацията е внедрила транзакционна система от среден клас Atlantis ERP, в която отразява всички логистични и финансови операции ежедневно. Компанията също така е изградила процес по бизнес планиране, при който на база на данните от продажбите и очакванията за развитието на пазара, бизнес анализатори създават годишен бизнес план на месечно ниво, който също бива съхранен в ERP системата.

С течение на времето в първите месеци след въвеждане на бизнес планирането, мениджмънта започва да открива пропуски в процеса и би искал да получава точна и навременна информация за реалните продажби, планираните продажби и точността на бизнес плановете, която е съществена поради това, че те влияят на цикъла на доставки, маркетинговите кампании, управлението на персонала и т.н. ERP системата не може да предостави необходимата информация поради големия обем от продажби – над 500,000 милиона записа на година, тъй като анализите върху такъв обем от данни затрудняват оперативната дейност на информационна система. Също така изграждането на справки и отчети в самата ERP система, налага използването на консултанти, които да програмират необходимите отчети, който в последствие не могат да бъдат променяни без тяхна помощ. Организацията също така осъзнава ясно, че достига лимита на своята ERP система и в скоро време ще е принудена да архивира данните съхранявани за предишни години.

Business Intelligence подход за решаване на проблема

Евентуалната имплементацията на Business Intelligence в компанията, рязко би подобрило достъпа до информация, като също така би улеснило работата на ERP системата освобождавайки я от задачата да съхранява исторически данни. Една възможна реализация на тази концепция е създаването на корпоративен Data Warehouse, с прилежащ към него ETL процес, който да извлича периодично данните за реалните продажби и бизнес плана от транзакционната система. След което информацията съхранявана в Data Warehouse-а се предоставя за анализиране от страна на потребителите чрез In-Memory OLAP среда.

Техническа реализация

На база на системният и функционален анализ на информационните изисквания на организацията се установява, че необходимото ниво на данните за анализ са следните бизнес обекти:

- **магазин** (Store), със свързаните в йерархия обекти **град** (City) и **търговска верига** (Retail).

- **продукт** (Material), със свързаните в йерархия обекти **дивизия** (Division), **продуктова фамилия** (Family), **продуктова група** (MaterialGroup) и **продуктова категория** (MaterialCategory).

- **компания** (CompanyCode), тъй като е възможно различните вериги да са организирани в различни юридически лица.

и бизнес метрики:

- **продажна цена без отстъпки**
- **цена на продукта**
- **количество на продажбата**

Също така е необходимо да се направи разграничаване на данните на реални и планирани, чрез обекта **тип** (Type) и анализиране на информацията в обща валута за всички вериги – Евро, чрез конвертиране на транзакционните данни към Евро и запазване, както на конвертираните, така и на оригиналните записи. Разграничаването е наложително да се извършва чрез допълнителен обект **валута** (CurrencyType).

Данните са достъпни в релационната база данни на OLTP системата Atlantis ERP, която е базирана на СУБД MS SQL Server 2000, и има класическа нормализирана структура. Извличането на информацията може да бъде извършено от таблиците:

- altex.dbo.material
- altex.dbo.salesman
- altex.dbo.itemtrans

- altex.dbo.itembalancesheet

- altex.dbo.fintrade

Периодичността на извличане на данните може да бъде седмична, тъй като извършваните анализи не са от оперативно-критичен характер за бизнеса, а детайлността на информация от времеви аспект може да бъде сведена до месечно ниво, поради факта, че съществени бизнес процеси не могат да бъдат изследвани на дневно ниво в този тип търговия.

За да се запази корпоративния стандарт, Data Warehouse би трябвало да бъде изграден също в СУБД MS SQL Server 2000. Поради простота на сценария и целите на изследването, изборът на логически модел на базата данни пада върху Star Schema, а стандартните физически параметри на базата данни няма да бъдат променяни.

Логически дизайн на корпоративния Data Warehouse – Star Schema

Star Schema е единодушно призната, като най-простият и ефективен денормализиран дизайн на релационен Data Warehouse [4]. При този подход всеки бизнес процес се моделира чрез два вида таблици – димeнсии и факти. Във факт таблицата се съхраняват измеримите страни на бизнес процеса, докато в димeнсиите се записват всички свързани с анализирания процес бизнес обекти. Йерархично свързаните бизнес обекти, биват съхранявани в една

димензионна таблица, което поражда денормализация на данните. Връзката между димензиите и факт таблицата е едно към много, като се осъществява чрез бизнес обекта на най-ниско транзакционно ниво. Ключът на димензионната таблицата е ключът на най-детайлният бизнес обект, а ключът на факт таблицата са всички ключове на свързаните димензионни таблици (foreign keys). Връзката между отделните бизнес процеси и респективно техните факт таблици се осъществява чрез споделените им бизнес обекти или съответно техните димензионни таблици.

Този логически дизайн съхранява данните в таблици само на две нива, което позволява изключително бързо изпълнение на заявки, като същевременно съчетава и лесна и бърза поддръжка. Евентуално изпълнение на SQL заявка върху този модел, би довело до прости join операции, без необходимост от вложени или циклични изпълнения.

Основен недостатък на модела е заемането на значително по-голямо дисково пространство от други модели, имайки предвид силната денормализация, но поради сравнително малкия мащаб на проекта и ниската цена за разширяване на хардуерните ресурси, това не може да бъде разглеждано като съществен проблем.

Прилагайки така описания логически модел върху разглежданият бизнес сценарии достигаме до сравнително прост логически дизайн показан на фигура 7.

Фигура 7.

Имаме следата структура:

- Една факт таблица **F_SALES** ориентирана към бизнес процеса по планиране и продажби с ключове към димензионните таблици и полета за бизнес метриките:
  - **себестойност на продукта в локална валута** (COST)
  - **себестойност на продукта в Евро** (COST_EUR)
  - **количество** (CANT)
  - **продажна цена без отстъпки в Евро** (VAL_FARA_TVA_SI_DISC_EUR)

- Димензионни таблици за вече описаните бизнес обекти с прилежащите им свързани йерархии **:**
  - **D_COMPANY_CODE –** компания: идентификатор и описание.

- **D_HTYPE –** тип на транзакционните данни планирани/реални: идентификатор и описание.
- **D_MATERIAL –** продукт: идентификатор, описание, цена на едро, цена на дребно, идентификатор на дивизия, име на дивизията, идентификатор на фамилията, име на фамилията, идентификатор на продуктовата група, име на продуктовата група, идентификатор на продуктовата категория, име на продуктовата атегория.
- **D_TIME –** дименсията време, която е необходима при почти всички бизнес сценарии. За да позволим пълен спектър от справки, включително за периоди, в който няма продажби е необходимо предварително да въведем информация в дименсията за целия период на очаквани анализи – например пет или десет години, като това може да бъде направено със следният T*SQL скрипт:

```
delete d_time;

use altex_dwh
GO

DECLARE
      @sYear INT,
      @eYear INT,
      @wYEAR INT,
      @wMonth INT,
      @wMonthName VARCHAR(10),
      @wQuarter VARCHAR(10)

SET @sYear = 2005
SET @eYear = 2010
SET @wYear = @sYear

WHILE @wYear <= @eYear
BEGIN

      SET @wMonth = 1

      WHILE @wMonth <= 12
      BEGIN
            SET @wMonthName =
                  CASE @wMonth
                   WHEN 1 THEN 'Jan'
                   WHEN 2 THEN 'Feb'
                   WHEN 3 THEN 'Mar'
                   WHEN 4 THEN 'Apr'
                   WHEN 5 THEN 'May'
                   WHEN 6 THEN 'Jun'
                   WHEN 7 THEN 'Jul'
                   WHEN 8 THEN 'Aug'
```

```
                    WHEN 9 THEN 'Sept'
                    WHEN 10 THEN 'Oct'
                    WHEN 11 THEN 'Nov'
                    WHEN 12 THEN 'Dec'
                    END

            SET @wQuarter =
                    CASE @wMonth
                    WHEN 1 THEN 'Q1' + str(@wYear,4)
                    WHEN 2 THEN 'Q1' + str(@wYear,4)
                    WHEN 3 THEN 'Q1' + str(@wYear,4)
                    WHEN 4 THEN 'Q2' + str(@wYear,4)
                    WHEN 5 THEN 'Q2' + str(@wYear,4)
                    WHEN 6 THEN 'Q2' + str(@wYear,4)
                    WHEN 7 THEN 'Q3' + str(@wYear,4)
                    WHEN 8 THEN 'Q3' + str(@wYear,4)
                    WHEN 9 THEN 'Q3' + str(@wYear,4)
                    WHEN 10 THEN 'Q4' + str(@wYear,4)
                    WHEN 11 THEN 'Q4' + str(@wYear,4)
                    WHEN 12 THEN 'Q4' + str(@wYear,4)
                    END
            INSERT INTO
D_TIME(ID,L_MONTH,L_MONTH_DESCR,L_MONTH_NUM,L_QUARTER,L_QUARTER_DESCR,L_
YEAR)
                    VALUES (ltrim(str(@wMonth,2)) + str(@wYear,4),@wMonthName,@wMonthName + ' '
+ substring(str(@wYear,4),3,4),@wMonth,@wQuarter,substring(@wQuarter,1,2),str(@wYear,4))
            SET @wMonth = @wMonth + 1
    END

    SET @wYear = @wYear + 1
END
```

Така написаният код, итеративно попълва дименсията време, като зарежда името и четеримесечието за всеки месец от избрания годишен диапазон.

- **D_STORE –** магазин: идентификатор на магазина, код, описание, търговска верига и град, в който се намира.

- Допълнително създаваме таблица **M_CUR_CONVERSION,** която използваме в ETL процеса за преобразуване на валутите.

За генерирането на така описания модел на данни в корпоративния Data Warehouse, бихме могли да използваме следният SQL скрипт:

```
IF EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'altex_dwh')
    DROP DATABASE [altex_dwh]
GO

CREATE DATABASE [altex_dwh]  ON (NAME = N'altex_dwh_Data', FILENAME = N'E:\Program Files\Microsoft SQL
Server\MSSQL\Data\altex_dwh_Data.MDF', SIZE = 61, FILEGROWTH = 10%) LOG ON (NAME = N'altex_dwh_Log',
FILENAME = N'E:\Program Files\Microsoft SQL Server\MSSQL\Data\altex_dwh_Log.LDF' , SIZE = 344, FILEGROWTH =
10%)
 COLLATE Cyrillic_General_CI_AS
GO

exec sp_dboption N'altex_dwh', N'autoclose', N'false'
GO
```

```
exec sp_dboption N'altex_dwh', N'bulkcopy', N'false'
GO

exec sp_dboption N'altex_dwh', N'trunc. log', N'false'
GO

exec sp_dboption N'altex_dwh', N'torn page detection', N'true'
GO

exec sp_dboption N'altex_dwh', N'read only', N'false'
GO

exec sp_dboption N'altex_dwh', N'dbo use', N'false'
GO

exec sp_dboption N'altex_dwh', N'single', N'false'
GO

exec sp_dboption N'altex_dwh', N'autoshrink', N'false'
GO

exec sp_dboption N'altex_dwh', N'ANSI null default', N'false'
GO

exec sp_dboption N'altex_dwh', N'recursive triggers', N'false'
GO

exec sp_dboption N'altex_dwh', N'ANSI nulls', N'false'
GO

exec sp_dboption N'altex_dwh', N'concat null yields null', N'false'
GO

exec sp_dboption N'altex_dwh', N'cursor close on commit', N'false'
GO

exec sp_dboption N'altex_dwh', N'default to local cursor', N'false'
GO

exec sp_dboption N'altex_dwh', N'quoted identifier', N'false'
GO

exec sp_dboption N'altex_dwh', N'ANSI warnings', N'false'
GO

exec sp_dboption N'altex_dwh', N'auto create statistics', N'true'
GO

exec sp_dboption N'altex_dwh', N'auto update statistics', N'true'
GO

if( (@@microsoftversion / power(2, 24) = 8) and (@@microsoftversion & 0xffff >= 724) )
      exec sp_dboption N'altex_dwh', N'db chaining', N'false'
GO

use [altex_dwh]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[F_SALES]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[F_SALES]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_COMPANY_CODE]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[D_COMPANY_CODE]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_HTYPE]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[D_HTYPE]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_MATERIAL]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[D_MATERIAL]
```

```
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_STORE]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[D_STORE]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_TIME]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[D_TIME]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[M_CUR_CONVERSION]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[M_CUR_CONVERSION]
GO

if not exists (select * from dbo.sysusers where name = N'guest' and hasdbaccess = 1)
    EXEC sp_grantdbaccess N'guest'
GO
CREATE TABLE [dbo].[F_SALES] (
    [D_COMPANY_CODE_ID] [int] NULL ,
    [D_HTYPE_ID] [int] NULL ,
    [D_MATERIAL_ID] [int] NULL ,
    [D_TIME_ID] [char] (8) COLLATE Cyrillic_General_CI_AS NULL ,
    [VAL_FARA_TVA_SI_DISC] [float] NULL ,
    [COST] [float] NULL ,
    [CANT] [float] NULL ,
    [_ID] [int] NULL ,
    [COST_EUR] [float] NULL ,
    [VAL_FARA_TVA_SI_DISC_EUR] [float] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_COMPANY_CODE] (
    [ID] [int] NOT NULL ,
    [DESCR] [char] (255) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_HTYPE] (
    [ID] [int] NOT NULL ,
    [DESCR] [varchar] (20) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_MATERIAL] (
    [ID] [int] NOT NULL ,
    [DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL ,
    [WHSPRICE] [float] NULL ,
    [RTLPRICE] [float] NULL ,
    [L_DIVIZII_ID] [int] NULL ,
    [L_DIVIZII_DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL ,
    [L_FAMILII_ID] [int] NULL ,
    [L_FAMILII_DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL ,
    [L_MATGROUP_ID] [int] NULL ,
    [L_MATGROUP_DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL ,
    [L_MATCAT_ID] [int] NULL ,
    [L_MATCAT_DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_STORE] (
    [ID] [int] NOT NULL ,
    [CODE] [varchar] (25) COLLATE Cyrillic_General_CI_AS NULL ,
    [NAME] [varchar] (255) COLLATE Cyrillic_General_CI_AS NULL ,
    [L_CITY] [varchar] (30) COLLATE Cyrillic_General_CI_AS NULL ,
    [L_RETAIL] [varchar] (20) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_TIME] (
    [ID] [char] (8) COLLATE Cyrillic_General_CI_AS NOT NULL ,
    [L_MONTH] [char] (10) COLLATE Cyrillic_General_CI_AS NULL ,
    [L_QUARTER] [char] (10) COLLATE Cyrillic_General_CI_AS NULL ,
    [L_YEAR] [char] (10) COLLATE Cyrillic_General_CI_AS NULL ,
```

```
	[L_MONTH_DESCR] [char] (10) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_MONTH_NUM] [int] NULL ,
	[L_QUARTER_DESCR] [char] (10) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[M_CUR_CONVERSION] (
	[PERIOD] [char] (8) COLLATE Cyrillic_General_CI_AS NOT NULL ,
	[XVALUE] [float] NULL
) ON [PRIMARY]
GO
```

Кодът представлява стандартният подход за създаване на логическа схема в СУБД, като създава необходимият набор от талблици, колони и техните типове, главни и свързващи ключове. Също така основни параметри на схемата биват дефинирани в началото на скрипта.

Изграждане на ETL процес

Така създадения Data Warehouse е необходимо да бъде периодично зареждан с данни. За целта изграждаме ETL процес, който ще използва вградените възможности на MS SQL Server 2000, чрез изпълнение на T*SQL скриптове. Скриптовете предлагат сравнително прост и ясен подход, като могат да бъдат стартирани периодично използвайки MS SQL Scheduling Engine.

След анализ на очакванията на организацията се установява, че ежеседмичното извличане на информация покрива потребителските изисквания.

Зареждането на информация трябва да е организирано на два етапа:

1. зареждане на данни за бизнес обектите (номенклатурните данни).
2. зараждане на данни за фактите (транзакционните данни).

3. извършване на преобразуването към единна валута.

Това би елиминирало възможността да заредим факти, за които нямаме въведени свързаните с тях бизнес обекти и по този начин да създадем не консистентност в Data Warehouse средата.

Зареждането на бизнес обектите може да бъде направено чрез следният примерен скрипт:

```
delete altex_dwh.dbo.D_MATERIAL;

insert into altex_dwh.dbo.D_MATERIAL
select m.id as ID,m.description as DESCR,m.whsprice as WHSPRICE,m.rtlprice as RTLPRICE,
       d.codeid as L_DIVIZII_ID,d.descr as L_DIVIZII_DESCR,
       f.codeid as L_FAMILII_ID,f.descr as L_FAMILII_DESCR,
       igp.codeid as L_MATGROUP_ID,igp.descr as L_MATGROUP_DESCR,
       ict.codeid as L_MATCAT_ID,ict.descr as L_MATCAT_DESCR
  from altex.dbo.material m,
       altex.dbo.r_divizii d,
       altex.dbo.r_familii f,
       altex.dbo.itemcategory ict,
       altex.dbo.itemgroup igp
 where m.igpid = igp.codeid
   and m.ictid = ict.codeid
   and m.r_divid = d.codeid
   and m.r_famid = f.codeid
   and m.comid = 1

delete altex_dwh.dbo.D_STORE;

insert into altex_dwh.dbo.D_STORE
select s.id as ID, s.code as CODE, s.name as NAME, s.city as L_CITY, s.r_retail as L_RETAIL
  from altex.dbo.salesman s
  where s.comid = 1
/*Currency conversion to EUR; Period format MMYYYY*/
delete altex_dwh.dbo.M_CUR_CONVERSION;

insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('12007',3.6445);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('22007',3.5404);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('32007',3.5074);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('42007',3.4911);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('52007',3.5071);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('62007',3.5483);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('72007',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('82007',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('92007',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('102007',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('112007',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('122007',3.5458);

/*Harmonized type - Realizat/Forecast*/
delete altex_dwh.dbo.D_HTYPE;

insert into altex_dwh.dbo.D_HTYPE
    values (1,'Forecast');
insert into altex_dwh.dbo.D_HTYPE
    values (2,'Actuals');

/*Company code - changes are not expected*/
delete from altex_dwh.dbo.D_COMPANY_CODE;

insert into altex_dwh.dbo.D_COMPANY_CODE
select c.codeid as ID,c.name as DESCR
  from altex.dbo.company c
 where c.codeid = 1;
```

Така написаният код изтрива текущото номеклатурно съдържание на дименсионните таблици след което последователно ги зарежда с данни. Тъй като дименсионните таблици съхраняват цялата йерархия от бизнес обекти в денормализиран вид е наложително при извличане на данните, да бъде извършен join на таблиците от силно нормализираната база данни на ERP системата. Също така се задават и стойности на служебните обекти **тип** на данните – планирани/реални и стойности на таблицата за конвертиране към Евро. При реално използване на този сценарии, би било наложително тези стойности за превалутиране да бъдат извличани също от ERP системата, за запазване на консистентност.

Зареждането фактите и конвертирането към Евро може да бъде направено чрез следният примерен скрипт:

```
-- real data
delete altex_dwh.dbo.F_SALES;
insert into altex_dwh.dbo.F_SALES(D_COMPANY_CODE_ID, D_HTYPE_ID, D_MATERIAL_ID, D_STORE_ID, D_TIME_ID,VAL_FARA_TVA_SI_DISC,COST,CANT)
select 1 as D_COMPANY_CODE_ID,2 as D_HTYPE_ID,m.id as D_MATERIAL_ID,s.id as D_STORE_ID,ltrim(str(month(trndate),2)) + str(year(trndate),4) as D_TIME_ID,
    sum(itt.outputquantmode*itt.ltrnvalue) as VAL_FARA_TVA_SI_DISC,
    sum(itt.outputquantmode*itt.primaryqty*isnull(itb.costvalue,0)) as COST,
    sum(itt.outputquantmode*itt.primaryqty) as CANT
 from altex.dbo.material m,
    altex.dbo.salesman s,
    altex.dbo.itemtrans itt,
    altex.dbo.itembalancesheet itb,
    altex.dbo.fintrade ftr
 where ftr.colidsalesman=s.id
  and ftr.id=itt.ftrid
  and m.id=itt.iteid
  and m.id=itb.masterid
  and year(itt.trndate)=itb.fyeid
  and month(itt.trndate)=itb.fipid
  and itt.source=5
  and m.comid=1
  and s.r_retail like 'altex%'
  and ftr.comid=1
  and itt.comid=1
  and itb.comid=1
  and glpurchasecode like '371.-.%'
  and itt.trndate between '2006/01/01' and '2006/12/31'
group by m.id,s.id,ltrim(str(month(trndate),2)) + str(year(trndate),4)

--budget
insert into altex_dwh.dbo.F_SALES(D_COMPANY_CODE_ID,D_HTYPE_ID,D_MATERIAL_ID,D_STORE_ID,D_TIME_ID,VAL_FARA_TVA_SI_DISC,COST,CANT)
select 1 as D_COMPANY_CODE_ID,1 as D_HTYPE_ID,m.id as D_MATERIAL_ID,s.id as D_STORE_ID, ltrim(str(fipid,2)) + str(fyeid,4) as D_TIME_ID,
    sum(budgetvalue1*whsprice) as VAL_FARA_TVA_SI_DISC,
    sum(budgetvalue1*(case when calcdate<'2005/07/01' then isnull(itf.costvalue/10000,0) else isnull(itf.costvalue,0) end)) as COST,
```

```
        sum(budgetvalue1) as CANT
    from altex.dbo.material m,
        altex.dbo.salesman s,
        altex.dbo.itemfindata itf,
        altex.dbo.budgetdata b
    where s.id=dvalue1
      and m.id=dvalue3
      and itf.masterid=m.id
      and fipid between 8 and 12
      and fyeid=2006
      and m.comid=1
      and s.comid=1
      and itf.comid=1
    group by m.id, s.id, ltrim(str(fipid,2)) +  str(fyeid,4)

    -- set EUR values
    update F_SALES
    set COST_EUR = COST / (select xvalue from M_CUR_CONVERSION where PERIOD = D_TIME_ID),
        VAL_FARA_TVA_SI_DISC_EUR = VAL_FARA_TVA_SI_DISC / (select xvalue FROM M_CUR_CONVERSION
WHERE PERIOD = D_TIME_ID)
```

Чрез така написаният код, последователно зареждаме в корпоративният Data Warehouse реалните и бюджетните данни, след което извършваме конвертирането към Евро.

Извличането на данните, налага намирането на ключовете на свързаните бизнес обекти, за да осигурим правилен join на таблиците в Star Schema-та при изпълнение на заявки. Също така е необходими задаването на някой филтри, които да изключат тестови или непълни данни въвеждани в ERP системата.

Самото конвертиране към валута е просто, като е необходимо да бъде извършено обновяване на факт таблицата, при което за всяка стойност в локална валута се извършва превалутиране на база на финансовият период на транзакцията и получената стойност се записва в празната колона за бизнес метрика в Евро.

Създаване на In-Memory OLAP среда и интеграция с корпоративният Data Warehouse. Изграждане на потребителска функционалност.

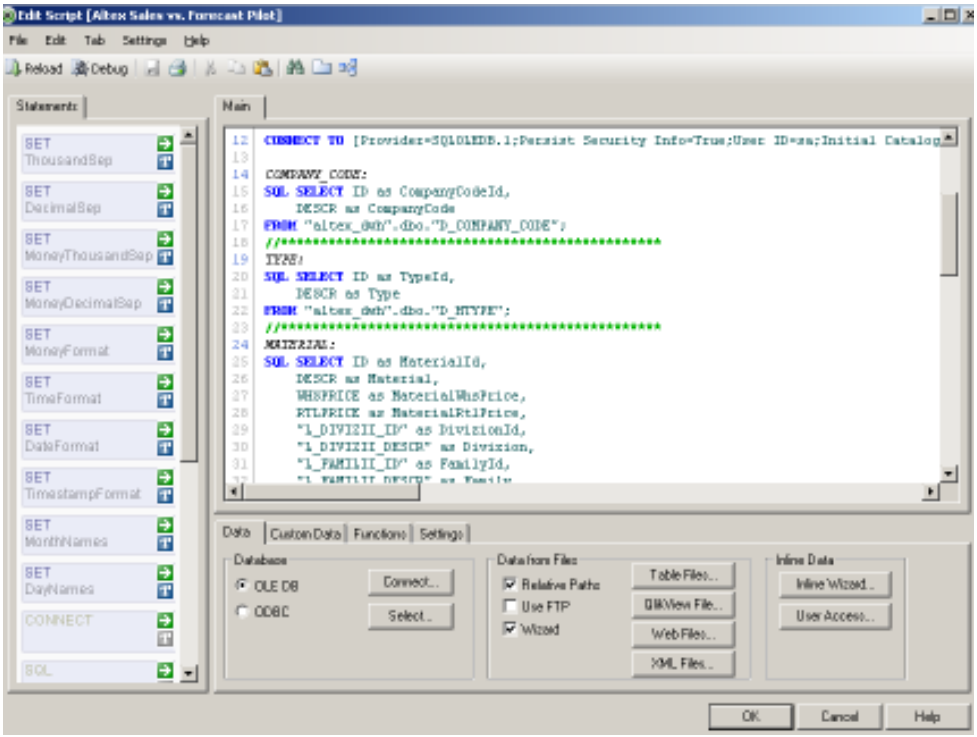
За да завършим успешно проекта е необходимо да интегрираме крайна потребителска аналитична среда с вече зареденият с данни корпоративен Data Warehouse и да предоставим достатъчен набор от аналитични средства на крайният потребител.

За целта на проекта сме се спряли върху иновативен In-Memory OLAP продукт – QlikView (http://www.qliktech.com). По същество, QlikView, представлява асоциативна база данни, която работи изцяло в оперативната памет на персоналния компютър или сървър на който е стартирана. Информацията бива извлечена от Data Warehouse или директно от OLTP система, като начинът на съхранение автоматично открива връзките между отделните бизнес обекти и техните стойности, което позволява изключително лесни анализи от страна на потребителя, основно базирани на филтриране и агрегиране на информация.

QlikView съхранява данните, заедно с мета-данните за бизнес обектите и връзките към източници на информация в единен файл, който може да бъде периодично обновяван и използван както в он-лайн, така и в офф-лайн режим. Това позволява гъвкавост при използване на продукта от страна на крайните потребители.

Също така QlikView проектът може да бъде интегриран с QlikView сървърно приложение, което би позволило достъп до същата функционалност чрез DHTML съвместими Интернет браузъри.

QlikView предлага скрипт интерфейс за извличане на данни, показан на фигура 8.



Фигура 8.

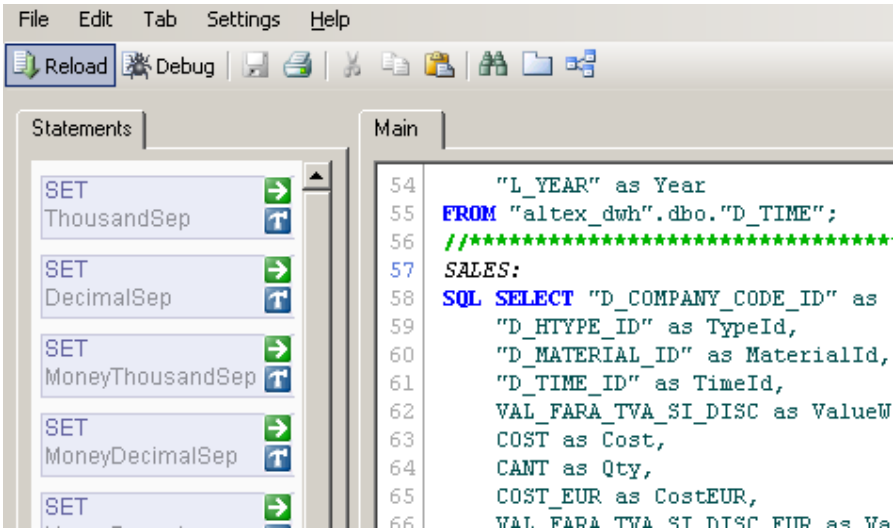
Възползвайки се от вече изчистената и интегриране информация, предоставяна от корпоративният Data Warehouse, ние можем директно да извлечем данните без да е необходимо да прилагаме каквито и да било трансформации. Така, всички данни са бързо и лесно достъпни за крайният потребител. Създаваме следният скрипт, който перидочно ще обновява наличните данни в OLAP средата и създава връзката между наличните бизнес обекти и SQL заявките за тяхното обновяване:

```
CONNECT TO [Provider=SQLOLEDB.1;Persist Security Info=True;User ID=sa;Initial Catalog=altex_dwh;Data Source=user9;Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=BGSSL321A;Use Encryption for Data=False;Tag with column collation when possible=False] (XPassword is AKBRaZFNFTcEXQFGEB);

COMPANY_CODE:
SQL SELECT ID as CompanyCodeId,
   DESCR as CompanyCode
FROM "altex_dwh".dbo."D_COMPANY_CODE";
//**********
TYPE:
SQL SELECT ID as TypeId,
   DESCR as Type
FROM "altex_dwh".dbo."D_HTYPE";
//**********
MATERIAL:
SQL SELECT ID as MaterialId,
   DESCR as Material,
```

```
    WHSPRICE as MaterialWhsPrice,
    RTLPRICE as MaterialRtlPrice,
    "L_DIVIZII_ID" as DivizionId,
    "L_DIVIZII_DESCR" as Divizion,
    "L_FAMILII_ID" as FamilyId,
    "L_FAMILII_DESCR" as Family,
    "L_MATGROUP_ID" as MaterialGroupId,
    "L_MATGROUP_DESCR" as MaterialGroup,
    "L_MATCAT_ID" as MaterialCategoryId,
    "L_MATCAT_DESCR" as MaterialCategory
FROM "altex_dwh".dbo."D_MATERIAL";
//*********************************************
STORE:
SQL SELECT ID as StoreId,
    CODE as StoreCode,
    NAME as Store,
    "L_CITY" as City,
    "L_RETAIL" as Retail
FROM "altex_dwh".dbo."D_STORE";
//*********************************************
TIME:
SQL SELECT ID as TimeId,
    "L_MONTH" as MonthName,
    "L_MONTH_DESCR" as Month,
    "L_MONTH_NUM" as MonthNumber,
    "L_QUARTER" as Quarter,
    "L_QUARTER_DESCR" as QuarterName,
    "L_YEAR" as Year
FROM "altex_dwh".dbo."D_TIME";
//*********************************************
SALES:
SQL SELECT "D_COMPANY_CODE_ID" as CompanyCodeId,
    "D_HTYPE_ID" as TypeId,
    "D_MATERIAL_ID" as MaterialId,
    "D_TIME_ID" as TimeId,
    VAL_FARA_TVA_SI_DISC as ValueWoVATandDisc,
    COST as Cost,
    CANT as Qty,
    COST_EUR as CostEUR,
    VAL_FARA_TVA_SI_DISC_EUR as ValueWoVATandDiscEUR,
    "D_STORE_ID" as StoreId
FROM "altex_dwh".dbo."F_SALES";
//*********************************************
CURRENCY_CONVERSION:
SQL SELECT PERIOD as TimeId,
    XVALUE as ConversionValue
FROM "altex_dwh".dbo."M_CUR_CONVERSION";
```

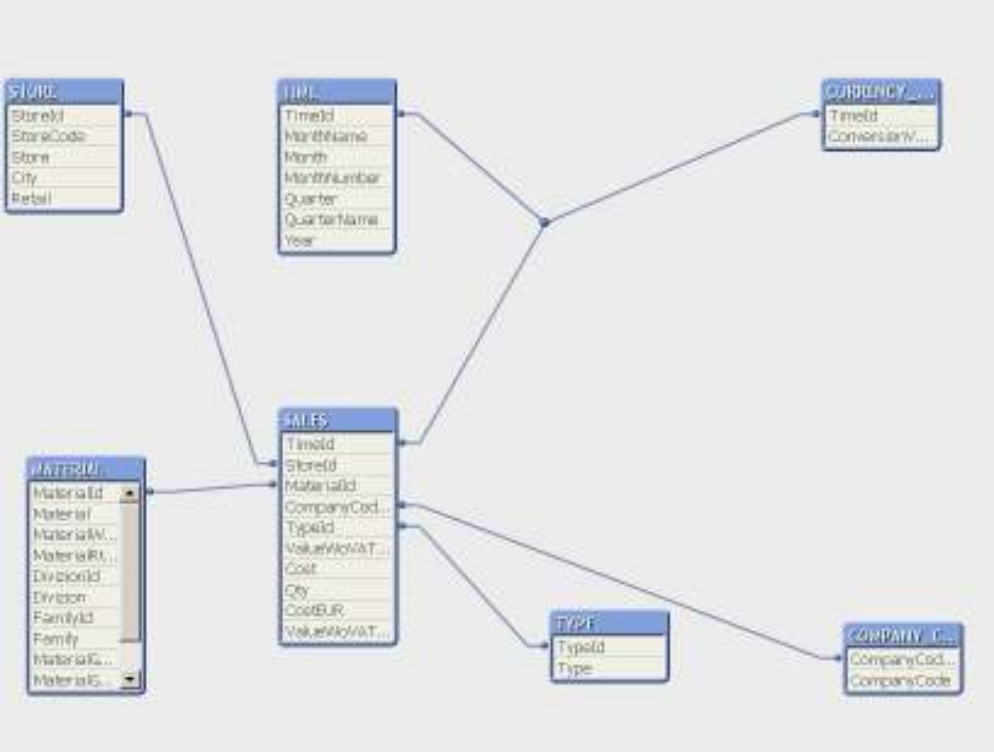
Така написаният код може да бъде изпълнен, чрез натускане на бутона “Reload Data” в QlikView както е показано на фигура 9.



Фигура 9.

При изпълнение, QlikView се свързва с базата данни указана чрез ключовата дума CONNECT TO и извлича данните чрез SQL заявки, без извършване на каквито и да било трансформации за настоящият сценарии.

След изпълнение на скрипта, QlikView автоматично генерира асоциативна база данни, със структура и бизнес обекти отговарящи на входния поток от информация. Това може да бъде проверено, чрез преглеждане на схема на асоциативната база данни и системната таблица, показани на фигура 10 и 11.



Фигура 10.

Tables

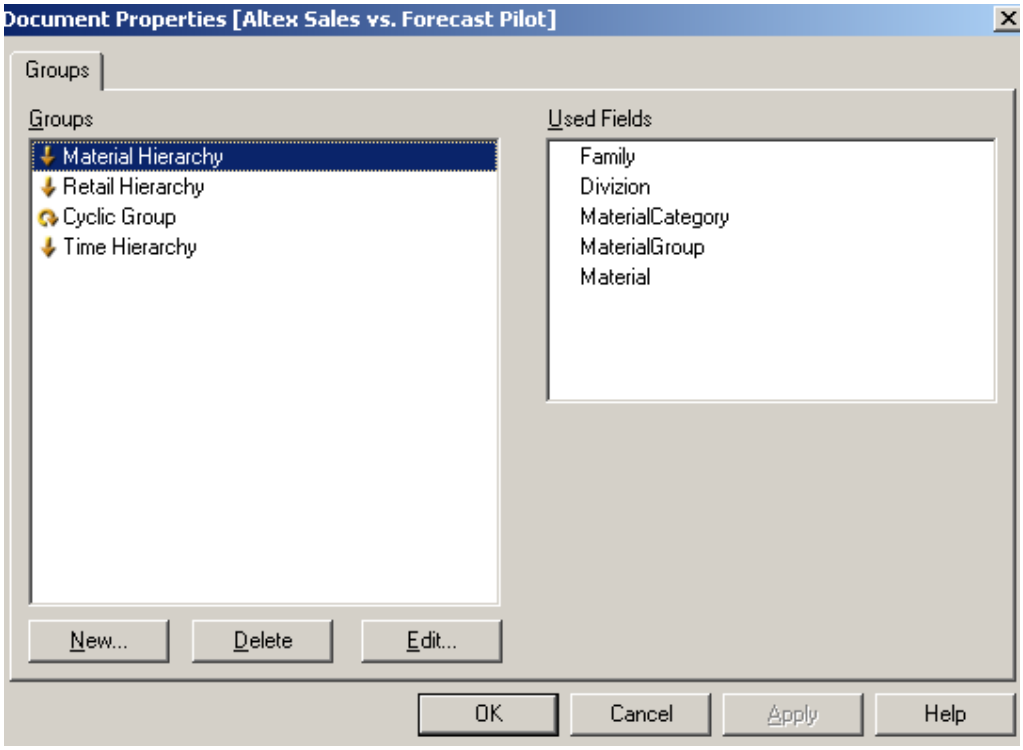
Name	Loosely Coupled	# Records	# Fields	# Keys
COMPANY_CODE	☐	1	2	1
CURRENCY_CONVERSIO	☐	12	2	1
MATERIAL	☐	37522	12	1
SALES	☐	495172	10	5
STORE	☐	190	5	1
TIME	☐	72	7	1
TYPE	☐	2	2	1

Fields

#	Name	# Tables	# Values	# Distinct	# Possible	Comment
6	CompanyCodeId	2	495172	1		
7	CompanyCode	1	1	1		
8	TypeId	2	495172	2		
9	Type	1	2	2		
10	MaterialId	2	0	37522		
11	Material	1	37522	32282		
12	MaterialWhsPrice	1	37522	2017		
13	MaterialRtlPrice	1	37522	2030		
14	DivizionId	1	37522	10		
15	Divizion	1	37522	10		

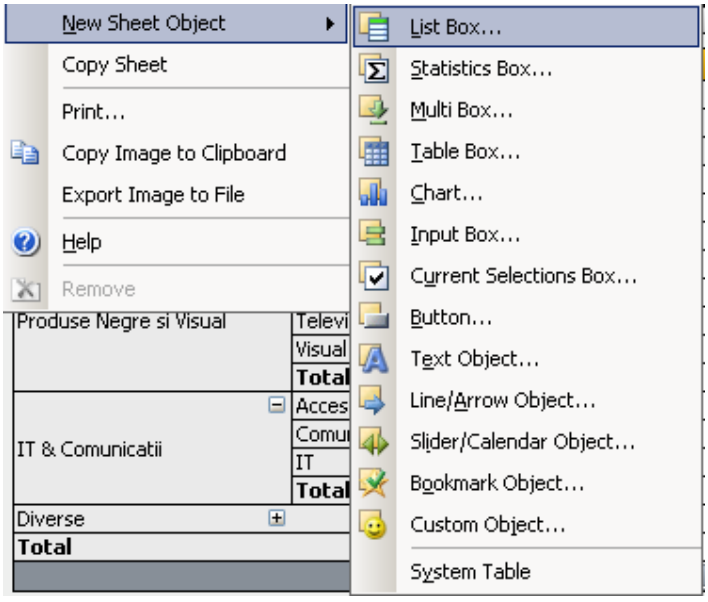
Фигура 11.

Така генерираните автоматично обекти, могат да бъдат използвани за динамични анализи, като единствено е необходимо да създадем йерархичните връзки, както е показано на фигура 12.



Фигура 12.

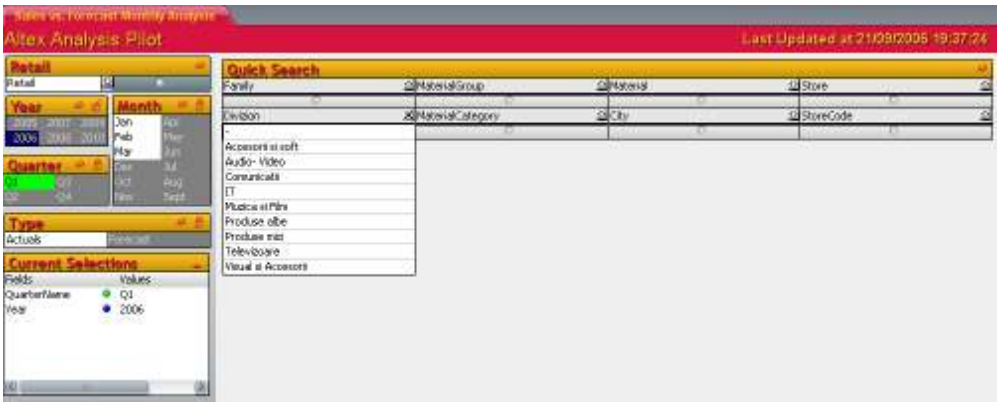
QlikView позволява лесно и бързо създавне на таблици, графики, филтри и множество други аналитични обекти, чрез използване на контекстното меню показано на фигура 13.



Фигура 13.

Посредством него, можем да изградим аналитична среда, която да отговаря на очакванията на бизнес анализаторите и да създава аналитична пътека на работа. Подобен подход би подобрил използването на системата, елиминирайки нуждата потребителите да са запознати детайлно с продукта.

За целта създаваме обекти за филтриране и бързо търсене на информация по търговска верига, година, месец, четиримесечие, тип на данните (планирани/реални), продуктова и магазинна йерархия. Функционалността е показана на фигура 14.



Фигура 14.

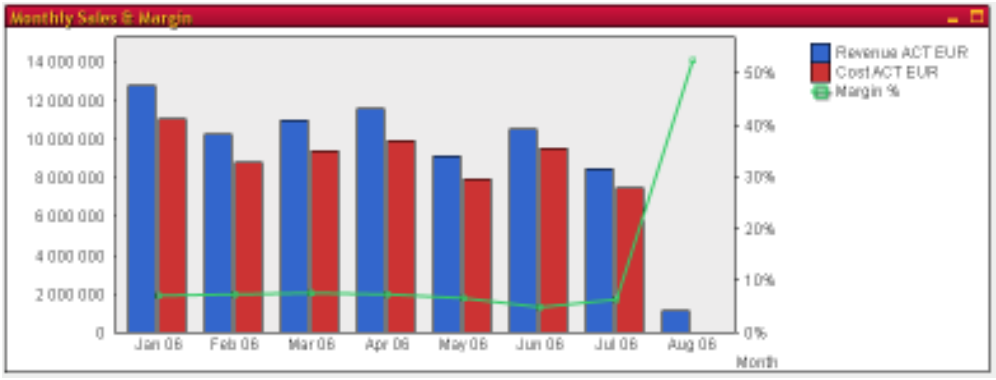
При избиране на конкретна стойност на даден обект, In-Memory OLAP средата автоматично филтрира целият набор от данни заредени в асоциативната база данни, което позволява лесни и динамични справки и отчети. Всичко обекти създавани в QlikView позволяват интерактивна работа с мишката на персоналният компютър.

За нуждите на бизнес анализаторите създаваме няколко таблици и графики, представящи планираните, реалните продажби,

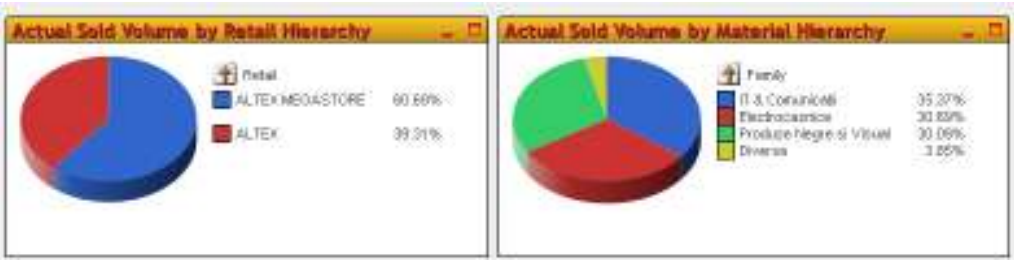
оборота и печалбата по търговски вериги и продуктова йерархия във времето показани на фигури 15 – 18.

Фигура 15. Pivot таблица за анализиране на продажбите по месец, продуктова дивизия и фамилия, показваща отношението на планиран/реален оборот, печалба и очаквани резултати в Евро към края на текущата година.

Фигура 16. Графика за динамичен анализ на реалната печалба и оборот по продуктовата йерархия. Позволява drill-down анализ на йерархията, чрез интерактивна работа с мишката.



Фигура 17. Месечен trend анализ.



Фигура 18. Интерактивни графики за анализ на реалните продажби по магазинна и продуктова йерархия.

Заключение

Така изградена и достъпна за потребителите OLAP средата завършва Business Intelligence имплементацията, като позволява бърз и ефективен достъп до необходимата информация за бизнес планиране и анализиране на продажбите в различните търговски вериги на компанията от страна на централният офис. Избраната архитектура позволява лесно надграждане, чрез добавяне на допълнителни бизнес процеси и респективно факт таблици към съществуващият корпоративен Data Warehouse и разширяване на ETL процеса.

От своя страна In-Memory OLAP средата разчитайки на доставяните от Data Warehouse-а изчистени, интегрирани и консистентни данни е фокусирана изцяло към предоставяне на добър краен интерфейс на бизнес потребителите и позволява следването на индивидуални практики и подход към анализите, справките и отчетите. Уникалната функционалност на QlikView да съхранява асоциативно данните заедно с прилежащите им мета-данни в единна структура зареждана в паметта позволява гъвкавост на работа в on-line и off-line режим, както и лесно дистрибутиране на аналитична информация по е-майл и създаване на PDF отчети за отпечатване.

От гледна точка на производителност, при така изградениият модел на данни, Data Warehouse-а се явява основна среда за историческо запазване на информацията на компанията и вероятно в дългосрочен план би съхранил от два до пет милиарда записа на най-детайлно транзакционно ниво. Практическите опити показват,

че QlikView е способно да обработи Star Schema с размер на факт таблицата в рамките на два до десет милиона записа на персонален компютър. Това предполага извличането на агрегирани подмножества на информацията съхранявана в Data Warehouse-а, което е логично с оглед на това, че такива големи обеми от данни, обикновено не носят добавена стойност на бизнес анализаторите.

Друг възможен подход е внедряването на QlikView Server, който би позволил значително по-голям обем на аналитично достъпна информация, чрез динамично управление на зарежданите в паметта асоциативни аналитични данни.

Като изводи от изследваната интеграция на класически Data Warehouse и In-Memory OLAP среда при Business Intelligence имплементации бихме могли да посочим:

- Постигането на гъвкава, лесно управляема, предоставяща възможност за бъдещ растеж Business Intelligence архитектура, чрез ясно разделение на функциите на отделните компоненти: Data Warehouse фокусиран върху съхраняване на историческата информация, ETL процес доставящ чисти и консистентни данни и OLAP среда предоставяща ефективен краен потребителски интерфейс.

- Сравнително бързо достигане на приемлив за бизнеса резултат и намалено проектно време, благодарение на богатата стандартна функционалност на In-Memory OLAP средата и лесното настройване за конкретни потребителски очаквания. Интуитивният интерфейс предполага значително по-кратко

време за обичение на крайните потребители в сравнение с подобни ROLAP и MOLAP имплементации

- Производителността при малки и средни обеми от данни (два-десет милиона записа) на In-Memory OLAP, рязко надхвърля тази на ROLAP средите, като се доближава до тази на MOLAP, без да жертва типичната за ROLAP гъвкавост. При по-големи обеми от данни е необходимо изграждането на стратегия за доставяне на агрегирана информация в зависимост от нуждите на потребителите за анализиране на конкретни бизнес процеси.

Литература

1. Inmon, William H.: Building the Data Warehouse. New York: John Wiley & Sons, 1996.

2. Ralph Kimball University - http://www.kimballgroup.com: ETL Toolkit, Data Warehouse Toolkit

3. Димитров, Димитър: Лекции "Business Intelligence на фирмата", СУ "Св. Климент Охридски", 2004-2007

4. The Data Warehouse Institute: http://www.tdwi.org

5. Larissa T. Moss , Shaku Atre: Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications.

Приложение

Скриптови файлове използвани в ETL процеса и корпоративният Data Warehouse:

- **db_init.sql:**

```
IF EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'altex_dwh')
	DROP DATABASE [altex_dwh]
GO

CREATE DATABASE [altex_dwh]  ON (NAME = N'altex_dwh_Data', FILENAME = N'E:\Program Files\Microsoft SQL Server\MSSQL\Data\altex_dwh_Data.MDF', SIZE = 61, FILEGROWTH = 10%) LOG ON (NAME = N'altex_dwh_Log', FILENAME = N'E:\Program Files\Microsoft SQL Server\MSSQL\Data\altex_dwh_Log.LDF' , SIZE = 344, FILEGROWTH = 10%)
 COLLATE Cyrillic_General_CI_AS
GO

exec sp_dboption N'altex_dwh', N'autoclose', N'false'
GO

exec sp_dboption N'altex_dwh', N'bulkcopy', N'false'
GO

exec sp_dboption N'altex_dwh', N'trunc. log', N'false'
GO

exec sp_dboption N'altex_dwh', N'torn page detection', N'true'
GO

exec sp_dboption N'altex_dwh', N'read only', N'false'
GO

exec sp_dboption N'altex_dwh', N'dbo use', N'false'
GO

exec sp_dboption N'altex_dwh', N'single', N'false'
GO

exec sp_dboption N'altex_dwh', N'autoshrink', N'false'
GO

exec sp_dboption N'altex_dwh', N'ANSI null default', N'false'
GO

exec sp_dboption N'altex_dwh', N'recursive triggers', N'false'
GO

exec sp_dboption N'altex_dwh', N'ANSI nulls', N'false'
GO

exec sp_dboption N'altex_dwh', N'concat null yields null', N'false'
GO

exec sp_dboption N'altex_dwh', N'cursor close on commit', N'false'
GO

exec sp_dboption N'altex_dwh', N'default to local cursor', N'false'
GO

exec sp_dboption N'altex_dwh', N'quoted identifier', N'false'
GO

exec sp_dboption N'altex_dwh', N'ANSI warnings', N'false'
GO

exec sp_dboption N'altex_dwh', N'auto create statistics', N'true'
GO
```

```
exec sp_dboption N'altex_dwh', N'auto update statistics', N'true'
GO

if( (@@microsoftversion / power(2, 24) = 8) and (@@microsoftversion & 0xffff >= 724) )
	exec sp_dboption N'altex_dwh', N'db chaining', N'false'
GO

use [altex_dwh]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[F_SALES]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
drop table [dbo].[F_SALES]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_COMPANY_CODE]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[D_COMPANY_CODE]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_HTYPE]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
drop table [dbo].[D_HTYPE]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_MATERIAL]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[D_MATERIAL]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_STORE]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
drop table [dbo].[D_STORE]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[D_TIME]') and OBJECTPROPERTY(id, N'IsUserTable') =
1)
drop table [dbo].[D_TIME]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[M_CUR_CONVERSION]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[M_CUR_CONVERSION]
GO

if not exists (select * from dbo.sysusers where name = N'guest' and hasdbaccess = 1)
	EXEC sp_grantdbaccess N'guest'
GO

CREATE TABLE [dbo].[F_SALES] (
	[D_COMPANY_CODE_ID] [int] NULL ,
	[D_HTYPE_ID] [int] NULL ,
	[D_MATERIAL_ID] [int] NULL ,
	[D_TIME_ID] [char] (8) COLLATE Cyrillic_General_CI_AS NULL ,
	[VAL_FARA_TVA_SI_DISC] [float] NULL ,
	[COST] [float] NULL ,
	[CANT] [float] NULL ,
	[D_STORE_ID] [int] NULL ,
	[COST_EUR] [float] NULL ,
	[VAL_FARA_TVA_SI_DISC_EUR] [float] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_COMPANY_CODE] (
	[ID] [int] NOT NULL ,
	[DESCR] [char] (255) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_HTYPE] (
	[ID] [int] NOT NULL ,
	[DESCR] [varchar] (20) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_MATERIAL] (
```

```
	[ID] [int] NOT NULL ,
	[DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL ,
	[WHSPRICE] [float] NULL ,
	[RTLPRICE] [float] NULL ,
	[L_DIVIZII_ID] [int] NULL ,
	[L_DIVIZII_DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_FAMILII_ID] [int] NULL ,
	[L_FAMILII_DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_MATGROUP_ID] [int] NULL ,
	[L_MATGROUP_DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_MATCAT_ID] [int] NULL ,
	[L_MATCAT_DESCR] [varchar] (50) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_STORE] (
	[ID] [int] NOT NULL ,
	[CODE] [varchar] (25) COLLATE Cyrillic_General_CI_AS NULL ,
	[NAME] [varchar] (255) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_CITY] [varchar] (30) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_RETAIL] [varchar] (20) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[D_TIME] (
	[ID] [char] (8) COLLATE Cyrillic_General_CI_AS NOT NULL ,
	[L_MONTH] [char] (10) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_QUARTER] [char] (10) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_YEAR] [char] (10) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_MONTH_DESCR] [char] (10) COLLATE Cyrillic_General_CI_AS NULL ,
	[L_MONTH_NUM] [int] NULL ,
	[L_QUARTER_DESCR] [char] (10) COLLATE Cyrillic_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[M_CUR_CONVERSION] (
	[PERIOD] [char] (8) COLLATE Cyrillic_General_CI_AS NOT NULL ,
	[XVALUE] [float] NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[D_COMPANY_CODE] WITH NOCHECK ADD
	CONSTRAINT [PK_D_COMPANY_CODE] PRIMARY KEY  CLUSTERED
	(
		[ID]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[D_HTYPE] WITH NOCHECK ADD
	CONSTRAINT [PK_D_HTYPE] PRIMARY KEY  CLUSTERED
	(
		[ID]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[D_MATERIAL] WITH NOCHECK ADD
	CONSTRAINT [PK_D_MATERIAL] PRIMARY KEY  CLUSTERED
	(
		[ID]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[D_STORE] WITH NOCHECK ADD
	CONSTRAINT [PK_D_STORE] PRIMARY KEY  CLUSTERED
	(
		[ID]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[D_TIME] WITH NOCHECK ADD
	CONSTRAINT [PK_D_TIME] PRIMARY KEY  CLUSTERED
	(
		[ID]
	)  ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[M_CUR_CONVERSION] WITH NOCHECK ADD
	CONSTRAINT [PK_M_CUR_CONVERSION] PRIMARY KEY  CLUSTERED
	(
		[PERIOD]
	)  ON [PRIMARY]
GO
```

- **dwh_d_time_generator_init.sql:**

```
/*
ALTEX QLIKVIEW PILOT DWH
SCRIPT FOR GENERATION OF TIME DIMENSION
Dimitar Dimitrov
*/

delete d_time;

use altex_dwh
GO

DECLARE
	@sYear INT,
	@eYear INT,
	@wYEAR INT,
	@wMonth INT,
	@wMonthName VARCHAR(10),
	@wQuarter VARCHAR(10)

SET @sYear = 2005
SET @eYear = 2010
SET @wYear = @sYear

WHILE @wYear <= @eYear
BEGIN

	SET @wMonth = 1

	WHILE @wMonth <= 12
	BEGIN
		SET @wMonthName =
			CASE @wMonth
			 WHEN 1 THEN 'Jan'
			 WHEN 2 THEN 'Feb'
			 WHEN 3 THEN 'Mar'
			 WHEN 4 THEN 'Apr'
			 WHEN 5 THEN 'May'
			 WHEN 6 THEN 'Jun'
			 WHEN 7 THEN 'Jul'
			 WHEN 8 THEN 'Aug'
			 WHEN 9 THEN 'Sept'
			 WHEN 10 THEN 'Oct'
			 WHEN 11 THEN 'Nov'
			 WHEN 12 THEN 'Dec'
			END

		SET @wQuarter =
			CASE @wMonth
			 WHEN 1 THEN 'Q1' + str(@wYear,4)
			 WHEN 2 THEN 'Q1' + str(@wYear,4)
			 WHEN 3 THEN 'Q1' + str(@wYear,4)
			 WHEN 4 THEN 'Q2' + str(@wYear,4)
			 WHEN 5 THEN 'Q2' + str(@wYear,4)
			 WHEN 6 THEN 'Q2' + str(@wYear,4)
			 WHEN 7 THEN 'Q3' + str(@wYear,4)
			 WHEN 8 THEN 'Q3' + str(@wYear,4)
			 WHEN 9 THEN 'Q3' + str(@wYear,4)
			 WHEN 10 THEN 'Q4' + str(@wYear,4)
			 WHEN 11 THEN 'Q4' + str(@wYear,4)
			 WHEN 12 THEN 'Q4' + str(@wYear,4)
			END
		INSERT INTO
D_TIME(ID,L_MONTH,L_MONTH_DESCR,L_MONTH_NUM,L_QUARTER,L_QUARTER_DESCR,L_YEAR)
```

```
                        VALUES (ltrim(str(@wMonth,2)) + str(@wYear,4),@wMonthName,@wMonthName + ' ' +
substring(str(@wYear,4),3,4),@wMonth,@wQuarter,substring(@wQuarter,1,2),str(@wYear,4))
                    SET @wMonth = @wMonth + 1
            END

            SET @wYear = @wYear + 1
END
```

- **dwh_fact_full.sql:**

```
/*
ALTEX QLIKVIEW PILOT DWH
TRANSACTION DATA FULL RELOAD
Dimitar Dimitrov
*/

-- realizat
delete altex_dwh.dbo.F_SALES;

insert into
altex_dwh.dbo.F_SALES(D_COMPANY_CODE_ID,D_HTYPE_ID,D_MATERIAL_ID,D_STORE_ID,D_TIME_ID,VAL_F
ARA_TVA_SI_DISC,COST,CANT)
select 1 as D_COMPANY_CODE_ID,2 as D_HTYPE_ID,m.id as D_MATERIAL_ID,s.id as
D_STORE_ID,ltrim(str(month(trndate),2)) + str(year(trndate),4) as D_TIME_ID,
      sum(itt.outputquantmode*itt.ltrnvalue) as VAL_FARA_TVA_SI_DISC,
      sum(itt.outputquantmode*itt.primaryqty*isnull(itb.costvalue,0)) as COST,
      sum(itt.outputquantmode*itt.primaryqty) as CANT
  from altex.dbo.material m,
      altex.dbo.salesman s,
      altex.dbo.itemtrans itt,
      altex.dbo.itembalancesheet itb,
      altex.dbo.fintrade ftr
 where ftr.colidsalesman=s.id
   and ftr.id=itt.ftrid
   and m.id=itt.iteid
   and m.id=itb.masterid
   and year(itt.trndate)=itb.fyeid
   and month(itt.trndate)=itb.fipid
   and itt.source=5
   and m.comid=1
   and s.r_retail like 'altex%'
   and ftr.comid=1
   and itt.comid=1
   and itb.comid=1
   and glpurchasecode like '371.-.%'
   and itt.trndate between '2006/01/01'  and '2006/12/31'
group by m.id,s.id,ltrim(str(month(trndate),2)) + str(year(trndate),4)

--budget
insert into
altex_dwh.dbo.F_SALES(D_COMPANY_CODE_ID,D_HTYPE_ID,D_MATERIAL_ID,D_STORE_ID,D_TIME_ID,VAL_F
ARA_TVA_SI_DISC,COST,CANT)
select 1 as D_COMPANY_CODE_ID,1 as D_HTYPE_ID,m.id as D_MATERIAL_ID,s.id as D_STORE_ID, ltrim(str(fipid,2))
+  str(fyeid,4) as D_TIME_ID,
      sum(budgetvalue1*whsprice) as VAL_FARA_TVA_SI_DISC,
      sum(budgetvalue1*(case when calcdate<'2005/07/01' then isnull(itf.costvalue/10000,0) else isnull(itf.costvalue,0) end)) as
COST,
      sum(budgetvalue1) as CANT
  from altex.dbo.material m,
      altex.dbo.salesman s,
      altex.dbo.itemfindata itf,
      altex.dbo.budgetdata b
 where s.id=dvalue1
   and m.id=dvalue3
   and itf.masterid=m.id
   and fipid between 8 and 12
   and fyeid=2006
   and m.comid=1
   and s.comid=1
   and itf.comid=1
group by m.id, s.id, ltrim(str(fipid,2)) +  str(fyeid,4)


-- set EUR values
```

```
update F_SALES
set COST_EUR = COST / (select xvalue from M_CUR_CONVERSION where PERIOD = D_TIME_ID),
    VAL_FARA_TVA_SI_DISC_EUR = VAL_FARA_TVA_SI_DISC / (select xvalue FROM M_CUR_CONVERSION
WHERE PERIOD = D_TIME_ID)
```

- **dwh_md_flat_init.sql:**

```
/*
ALTEX QLIKVIEW PILOT DWH
INITIAL PREPARATION OF MASTER DATA NOT SUPPOSED TO BE RELOADED
Dimitar Dimitrov
*/

/*Currency conversion to EUR; Period format MMYYYY*/
delete altex_dwh.dbo.M_CUR_CONVERSION;

insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('12006',3.6445);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('22006',3.5404);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('32006',3.5074);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('42006',3.4911);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('52006',3.5071);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('62006',3.5483);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('72006',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('82006',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('92006',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('102006',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('112006',3.5458);
insert into altex_dwh.dbo.M_CUR_CONVERSION
    values ('122006',3.5458);

/*Harmonized type - Realizat/Forecast*/
delete altex_dwh.dbo.D_HTYPE;

insert into altex_dwh.dbo.D_HTYPE
    values (1,'Forecast');
insert into altex_dwh.dbo.D_HTYPE
    values (2,'Actuals');

/*Company code - changes are not expected*/
delete from altex_dwh.dbo.D_COMPANY_CODE;


insert into altex_dwh.dbo.D_COMPANY_CODE
select c.codeid as ID,c.name as DESCR
  from altex.dbo.company c
 where c.codeid = 1;
```

- **dwh_md_full.sql:**

```
/*
ALTEX QLIKVIEW PILOT DWH
MASTER DATA FULL RELOAD
Dimitar Dimitrov
*/

delete altex_dwh.dbo.D_MATERIAL;

insert into altex_dwh.dbo.D_MATERIAL
select m.id as ID,m.description as DESCR,m.whsprice as WHSPRICE,m.rtlprice as RTLPRICE,
      d.codeid as L_DIVIZII_ID,d.descr as L_DIVIZII_DESCR,
      f.codeid as L_FAMILII_ID,f.descr as L_FAMILII_DESCR,
      igp.codeid as L_MATGROUP_ID,igp.descr as L_MATGROUP_DESCR,
```

```
      ict.codeid as L_MATCAT_ID,ict.descr as L_MATCAT_DESCR
  from altex.dbo.material m,
      altex.dbo.r_divizii d,
      altex.dbo.r_familii f,
      altex.dbo.itemcategory ict,
      altex.dbo.itemgroup igp
 where m.igpid = igp.codeid
   and m.ictid = ict.codeid
   and m.r_divid = d.codeid
   and m.r_famid = f.codeid
   and m.comid = 1

delete altex_dwh.dbo.D_STORE;

insert into altex_dwh.dbo.D_STORE
select s.id as ID, s.code as CODE, s.name as NAME, s.city as L_CITY, s.r_retail as L_RETAIL
  from altex.dbo.salesman s
  where s.comid = 1
```

Финалният QlikView файл *Altex_SvsF_v_0_1.qvw*, предоставящ In-Memory OLAP среда на крайните потребители се дистрибутира с настоящата дипломна работа.