УДК 004.5

Георгій Асєєв,
завідувач кафедри інформаційних технологій ХДАК,
доктор технічних наук, професор

Вітрини даних — необхідна ланка в концепціях побудови сховищ даних

Розглянуті вітрини даних, що створюються для інформаційного забезпечення аналітичних завдань конкретних управлінських підрозділів компанії чи організації. Висвітлені питання застосування багаторівневих рішень у СУБД сховищ даних. Проаналізована ідея поєднання концепції сховища і вітрини даних в одній реалізації та можливості використання сховища даних як єдиного джерела інтегрованих даних усіх вітрин даних.

Ключові слова: *вітрини даних, інформаційне забезпечення, багаторівневі рішення у СУБД сховищ даних, єдине джерело інтегрованих даних.*

У статті [1] розглянута архітектура електронних сховищ даних, при цьому вітрина даних згадана лише побіжно. У пропонованій публікації спробуємо заповнити цю прогалину.

Функціонально орієнтовані вітрини даних (Data Mart) є структурами даних, що забезпечують рішення аналітичних завдань у конкретній функціональній сфері або підрозділі компанії, наприклад, управлінні прибутковістю, аналізах ринків, ресурсів тощо. Іноді ці структури зберігання даних називають також кіосками даних. Вітрини даних можна розглядати як невеликі сховища, що створюються для інформаційного забезпечення аналітичних завдань конкретних управлінських підрозділів компанії чи організації.

Як правило, вітрина даних містить значно менше інформації, ніж електронне сховище даних, охоплює лише кілька предметних областей і має коротшу історію. Вітрини даних можна представити у вигляді логічно чи фізично розділених підмножин сховищ даних. Зазвичай вони будуються для обслуговування потреб певної групи користувачів.

Джерелом даних для вітрин є дані сховища, що, як правило, агрегуються та консолідуються за різними рівнями ієрархії. Детальні дані можуть також міститися у вітрині або відбиватися в ній у вигляді посилань на дані сховища.

Різні вітрини даних містять різноманітні комбінації та вибирання одних і тих самих деталізованих даних сховища. У сховищах даних є великі області для збирання/зберігання/переміщення існуючих даних, звідки ті можуть бути перерозподілені по вітринах даних, що якраз і призначені для доступу користувачів, які приймають рішення. Тоді як сховище даних призначене для управління ними, а вони надходять великими партіями від їхніх постачальників (наприклад, операційних систем), а також для організації і зберігання цих даних, вітрини даних можуть бути зосереджені на упаковці та пропозиції наборів даних кінцевим користувачам відповідно до їхніх потреб. Вітрина даних за своїм вихідним визначенням — це набір тематично пов'язаних баз даних, які містять інформацію, що стосується окремих аспектів діяльності корпорації. По суті, вітрина даних — це полегшений варіант сховища даних, що містить тільки тематично об'єднані дані. Цільова база даних максимально наближена до кінцевого користувача і може зберігати тематично орієнтовані агрегатні дані. Вітрина даних суттєво менша за обсягом, ніж корпоративне сховище даних, і для її реалізації не потрібна занадто потужна обчислювальна техніка.

Концепція вітрин даних була запропонована Forrester Research ще в 1991 році. На думку авторів, вона, як величезна кількість тематичних БД, де міститься інформація, що стосується окремих аспектів діяльності організації, має стати реальною альтернативою сховищам даних [2].

Концепція вітрин даних має низку безсумнівних переваг:

- аналітики працюють лише з тими даними, що реально їм потрібні;
- цільова БД вітрини даних максимально наближена до кінцевого користувача;
- вітрини даних зазвичай містять тематичні підмножини заздалегідь агрегованих даних, що спрощує проектування і налаштовування;
- для реалізації вітрин даних не потрібна високопотужна обчислювальна техніка, отже, знижуються витрати.

І саме вітрини даних (або щось дуже близьке до них) мав на увазі Е. Кодд, коли використовував термін "OLAP Server" [3].

Концепція вітрин даних має не лише незаперечні переваги, а й дуже серйозні проблеми. По суті, тут передбачається реалізація територіально розподіленої інформаційної системи з малоконтрольованою надмірністю, однак відсутні засоби для забезпечення цілісності та несуперечності збережених у ній даних.

Ідея поєднати обидві концепції — сховищ даних і вітрин даних — та використовувати сховище даних як єдине інтегроване джерело даних для вітрин даних, вочевидь, належить М. Демаресту (M. Demarest), який у 1994 році виклав її в праці "Побудова вітрин даних" [4]. І сьогодні застосовується саме таке багаторівневе рішення:

— *перший рівень:* загальнокорпоративна БД на основі реляційних СУБД (РСУБД) з нормалізованою або слабоденормалізованою схемою (деталізовані дані);

— *другий рівень:* БД рівня підрозділу (або кінцевого користувача), що реалізуються на основі багатовимірних СУБД (МСУБД) (агреговані дані);

— *третій рівень:* робочі місця кінцевих користувачів, де безпосередньо встановлений аналітичний інструментарій;

© Асєєв Г., 2012

поступово стає стандартом де-факто, даючи можливість найповніше реалізовувати і використовувати переваги кожного з підходів; компактне зберігання деталізованих даних і підтримання великих БД, що забезпечуються РСУБД; простота настроювання і задовільний час відгуку при роботі з агрегованими даними, що забезпечуються МСУБД.

Перші спроби впровадження вітрин даних виявилися настільки успішними, що навколо нової технології розпочався справжній бум. Пропонувалося взагалі відмовитися від реалізації корпоративного сховища і замінити його на сукупність вітрин даних. Однак незабаром з'ясувалося, що зі зростанням кількості вітрин збільшується складність їхньої взаємодії, оскільки зробити вітрини повністю незалежними не вдається. Зараз затвердилась точка зору, відповідно до якої розроблення корпоративного сховища має відбуватися паралельно з розробленням і впровадженням вітрин даних.

Після того, як дані потрапили в сховище, вони можуть бути поширені по будь-якій кількості вітрин даних для забезпечення доступу до них користувачів. Ці вітрини даних можуть мати довільну форму (від бази даних клієнт-сервера до настільних баз даних, OLAP кубів і навіть електронних таблиць). Можливий широкий вибір засобів створення користувацьких запитів, він має відповідати різноманітним перевагам і досвіду користувачів. Забезпечення широкого доступу до таких інструментів і легкість роботи з ними — це найдешевша для виконання частина середовища зберігання даних. Якщо дані в сховищі якісні та добре структуровані, то подальший їхній експорт у нові вітрини даних не буде дорогим.

Фактичним стандартом структури даних при розробленні вітрини є схеми "зірка" або "сніжинка". Схема "зірка" — популярний тип моделі даних для вітрин. Для цієї моделі характерна наявність таблиці фактів, оточеної пов'язаними з нею таблицями розмірностей. Запити до такої структури включають прості об'єднання таблиці фактів з кожною таблицею розмірностей. Проектується для виконання аналітичних запитів. Характеризується високою продуктивністю виконання запитів та невеликою надмірністю даних, високою, порівняно з нормалізованими структурами, продуктивністю. Для скорочення часу виконання запитів у деяких промислових СУБД та інструментах класу OLAP/Reporting вміло використані переваги схеми "зірка".

Оскільки у вітринах даних поряд з детальними мають зберігатися й агреговані дані, у випадку застосування схем "сніжинка" або "зірка" з'являються таблиці агрегованих фактів (агрегатів). Подібно до звичайних фактів агрегати можуть мати виміри. Крім того, вони повинні бути пов'язані з детальними фактами для забезпечення можливої деталізації. На практиці вітрини даних часто включають декілька таблиць фактів, пов'язаних між собою вимірами, які таким чином розділяються між декількома таблицями фактів. Така схема носить назву "розширена сніжинка".

При побудові схеми взаємодії корпоративного сховища і вітрин даних у межах створення СППР рекомендується визначити деяку спеціальну структуру для зберігання історичних даних і додатково розгорнути низку вітрин, заповнених даними з цієї структури. Тим самим вдається розділити два процеси: нагромадження історичних даних та їхній аналіз.

Як уже зазначалося, вітрини, побудовані на основі сховища даних або на базі первинних джерел, проектуються для задоволення потреб певної групи користувачів, орієнтованих на вирішення конкретних аналітичних завдань, і дають можливість порівняно легко забезпечити прийнятну продуктивність, оскільки містять менший обсяг даних, завчасно їх агрегують і затребувані обмеженим колом користувачів. Для побудови такої вітрини можна використовувати як реляційні, так і багатовимірні СУБД.

Реляційна форма подання даних, яка використовується в центральній загально-корпоративній БД, забезпечує найкомпактніший спосіб зберігання даних. А сучасні РСУБД уже "вміють" працювати навіть з терабайтними базами. І хоча така центральна система зазвичай не зможе забезпечити оперативного режиму опрацювання аналітичних запитів, при використанні нових способів індексації та зберігання даних, а так само часткової денормалізації таблиць, час опрацювання заздалегідь регламентованих запитів (а як такі, можна розглядати і регламентовані процедури вивантаження даних у багатовимірні БД) виявляється цілком прийнятним.

У свою чергу, використання МСУБД у вузлах нижнього рівня забезпечує мінімальний час опрацювання та відповіді на нерегламентовані запити користувача. Крім того, у деяких МСУБД є можливість зберігати дані як на постійній основі (безпосередньо в багатовимірній БД), так і динамічно (на час сеансу) завантажити дані з реляційних БД (на основі регламентованих запитів).

Таким чином, є можливість зберігати на постійній основі лише ті дані, які найчастіше запитуються в певному вузлі. Для всіх інших зберігаються лише описи їхньої структури і програми вивантаження з центральної БД. І хоча при первинному зверненні до таких віртуальних даних час відгуку може виявитися досить тривалим, таке рішення забезпечує високу гнучкість і потребує дешевших апаратних засобів.

РСУБД [2]. Сьогодні РСУБД стали домінуючим промисловим рішенням при реалізації найрізноманітніших СЕД. Вони забезпечують прийнятні часи відгуку при довільному вибиранні окремих записів та невеликих груп записів. А реальні обсяги БД, з якими вони вміють працювати, перевищують сотні гігабайтів. Більше того, сьогодні відомі, правда, поки на рівні демонстраційних версій, БД з обсягом у декілька терабайтів. Однак початково орієнтовані на реалізацію систем операційного оброблення даних РСУБД виявилися менш ефективними у завданнях аналітичного опрацювання. З чим це пов'язано?

По-перше, через наявність досить жорстких обмежень, накладених існуючою реалізацією мови SQL, аналітичні запити виходять досить громіздкими, а багато функцій опрацювання доводиться виносити в заздалегідь написані користувальницькі додатки. І якщо питання про громіздкість конструкцій як серйозний недолік є спірним (сьогодні практично ніхто не пише програми безпосередньо на SQL, а відповідні конструкції автоматично генеруються засобами клієнтського інструментарію), то обмеження SQL реально існують і їх так просто не уникнути.

Прикладом такого реально існуючого обмеження є припущення про те, що дані в реляційній базі невпорядковані, точніше, впорядковані випадковим чином. Але виконання більшості аналітичних функцій (наприклад, побудова "Прогнозу"), навпаки, неможливе без припущення про впорядкованість даних. Природно, при використанні реляційної бази є можливість після вибирання даних з бази виконати їхнє сортування і потім побудувати "Прогноз". Однак це потребуватиме додаткових витрат часу на сортування, яке потрібно проводити щоразу при зверненні до цієї функції, і, найголовніше, така функція може бути визначена і використана тільки в користувальницькому додатку, але не може бути вбудованою функцією мови SQL.

По-друге, для забезпечення прийнятного часу відповіді при використанні РСУБД необхідно вже на етапі проектування знати про всі можливі типи запитів, необхідні зрізи і рівні агрегації даних.

Основою традиційного реляційного підходу є нормалізація (декомпозиція) таблиць, що припускає усунення над-

мірності в їхніх основних ключах та транзитивних залежностей між реквізитами, що утворюють таблицю. Це дає змогу не тільки мінімізувати сумарний обсяг даних у БД, а й вирішує проблеми, пов'язані з різним видом аномалій, що виникають у процесі видалення та модифікації даних у ненормалізованих таблицях. І хоча у процесі нормалізації втрачаються і семантичні зв'язки, що існують між реквізитами, це не особливо критично для традиційних СЕД. Та невелика кількість зв'язків, що необхідна для реалізації певної програми, відома і легко реалізується за допомогою механізму зовнішніх ключів. Актуальніша ця проблема для аналітичних систем, де зазвичай навіть не можна заздалегідь визначити, які зв'язки між різними реквізитами будуть використовуватися частіше, а які не будуть застосовуватись взагалі. Все залежить від неординарності мислення конкретного аналітика, ситуації на ринку, у фірмі та багатьох інших факторів.

Однак основний недолік традиційних РСУБД полягав у тому, що в них як основний і часто єдиний механізм, що забезпечує швидкий пошук і вибирання окремих рядків у таблиці (або у пов'язаних через її зовнішні ключі), зазвичай використовуються різні модифікації індексів, заснованих на B-деревах. Але таке рішення виявляється ефективним лише при опрацюванні невеликих груп записів і високої інтенсивності модифікації даних у БД. В аналітичних системах введення і вибирання даних здійснюється великими порціями. А дані після того, як вони потрапляють до БД, залишаються незмінними протягом тривалого часу. В цьому випадку ефективнішим виявляється зберігання даних у формі частково денормалізованих таблиць, у яких для збільшення продуктивності можуть зберігатися не тільки деталізовані, а й попередньо обчислені агреговані значення. Для навігації та вибирання використовуються спеціалізовані, засновані на припущенні про малу змінюваність та незначну рухливість даних у БД, методи адресації та індексації. Такий спосіб їхньої організації іноді називають передобчисленням, підкреслюючи тим самим його відмінність від нормалізованого реляційного підходу, що передбачає динамічне обчислення різного виду підсумків (агрегація) і встановлення зв'язків між реквізитами з різних таблиць (операції з'єднання).

І все ж, на думку Е. Кодда [3], реляційні бази даних стали та залишаться і в майбутньому найбільш придатною технологією для реалізації інформаційних систем рівня підприємства. Головною ж причиною їхньої неефективності в аналітичних додатках є не стільки, власне, недоліки реляційного підходу, скільки те, що виробники РСУБД донедавна просто не звертали уваги на ринок аналітичних систем.

Але вже нині ситуація змінилася. І, мабуть, головною новацією тут є те, що сьогодні офіційно визнано необхідність і право на існування в реляційній БД таблиць з денормалізованою формою — різні модифікації схеми організації даних типу "зірка". У своєму класичному варіанті ця схема передбачає наявність однієї денормалізованої фактологічної таблиці (кількість рядків у цій таблиці зазвичай становить десятки і сотні мільйонів), з якою пов'язані кілька десятків відносно невеликих довідкових таблиць.

Іншим напрямом розвитку РСУБД є пошук і реалізація нових способів індексації та організації зберігання даних. Виникло завдання — відсіяти максимальну кількість даних, що не задовольняють умови запиту, ще до їхнього зчитування із зовнішнього накопичувача й одночасно мати індекс такого розміру, щоб він легко уміщався в оперативній пам'яті (чи мав порівнянний з нею розмір). Прикладом таких індексів є Bitmap-індекси, що виявляються досить ефективними при роботі з реквізитами, кількість значень яких відносно невелика. Не менший, якщо не більший, виграш може принести використання різних варіантів горизонтального або вертикального поділу таблиць (даних); наприклад, поділ однієї великої фактологічної таблиці на кілька окремих фрагментів (горизонтальна фрагментація). Такий розподіл може здійснюватися, приміром, відповідно до тимчасових інтервалів, до яких відносяться дані. Кожна з таких підтаблиць (фрагментів) має одну і ту саму структуру, формат реквізитів, індекси. І кожен з цих фрагментів може опрацьовуватись незалежно (наприклад, будувати і перебудовувати індекси). У запиті всю таблицю можна подавати як єдине ціле, причому фрагменти, що не задовольняють умови вибирання, легко відсікаються ще до моменту їхнього зчитування із зовнішнього накопичувача.

Іншим підходом до підвищення продуктивності є вертикальна фрагментація даних. Припустимо, що у нас є таблиця з 10 мільйонів рядків, кожен з яких складається з 30 полів (колонок) по 10 символів (байтів). Абстрагуючись від питань ефективності чи неефективності зберігання даних у конкретній реалізації РСУБД, припустимо, що обсяг результувальної БД дорівнює обсягу вихідних даних. У цьому випадку ми отримаємо БД розміром в 3 гігабайти.

У традиційних реалізаціях РСУБД дані зберігаються по рядках та при послідовному перегляді, незалежно від того, які значення із скількох колонок таблиці потрібні для формування відповіді на запит, будуть зчитані всі 3 гігабайти даних. Але в аналітичних запитах украй рідко виникає необхідність роботи одночасно з усіма колонками таблиці. Саме на цьому припущенні й ґрунтується механізм вертикальної фрагментації, при використанні якого дані зберігаються не по рядках, а по стовпцях. Таким чином, кожен стовпець є незалежним розділом даних і при запитах на читання може опрацьовуватись незалежно. І якщо в нашому прикладі для формування відповіді на запит будуть потрібні значення з трьох стовпців таблиці, то необхідно буде враховувати тільки 300 мегабайтів, а не 3 гігабайти даних.

До цього часу ми, в основному, говорили про переваги різних способів підвищення ефективності опрацювання аналітичних запитів, але жоден з цих підходів не універсальний і однаково ефективний в усіх ситуаціях. Сьогодні виробники РСУБД перебувають на етапі пошуку, і жоден з описаних раніше механізмів не є загальновизнаним, безперечним і універсальним.

МСУБД [2]. Простіше та ефективніше аналітичні системи реалізуються засобами спеціалізованих баз даних на основі багатовимірного подання даних. У цих системах дані організовані у вигляді не плоских таблиць (як у реляційних системах), а впорядкованих багатовимірних масивів гіперкубів (або полікубів).

Очевидно, що таке рішення вимагає більшої сумарної пам'яті для зберігання даних, є менш гнучким за необхідності модифікації структур даних і вимагає значних витрат часу при їхньому завантаженні. Але, як вже було сказано, в аналітичних завданнях все це компенсується за рахунок швидшого пошуку і вибирання даних, відсутності необхідності в багаторазовому поєднанні різних таблиць і багаторазового обчислення агрегованих значень. І, як правило, середній час відповіді на нерегламентований аналітичний запит при використанні багатовимірної СУБД зазвичай менший на один/два порядки, ніж у випадку реляційної СУБД із нормалізованою схемою даних.

Здавалося б, що все очевидно і вибір однозначний — багатовимірні БД. Однак не все так просто. Багатовимірні бази, через чисто історичні причини, "не вміють" працювати з великими обсягами даних. На сьогодні їхня реальна межа — база обсягом у 10—20 гігабайтів. І хоча це обмеження не пов'язане з якими-небудь об'єктивними внутрішніми недолі-

ками багатовимірного підходу і, швидше за все, є тимчасовим, однак це так. І на це не можна не зважати. До того ж за рахунок денормалізації та попередньо виконаної агрегації 20 гігабайтів у багатовимірній базі, в кращому випадку, еквівалентні не більше ніж 1 гігабайту вихідних даних. За оцінками Е. Кодда, для систем, заснованих на багатовимірному поданні даних, це співвідношення лежить у діапазоні від 2,5 до 100. І тут необхідно зупинитися на основному недоліку багатовимірних БД — неефективне, порівняно з реляційними БД, використання зовнішньої пам'яті. І це вже об'єктивний чинник.

В основі багатовимірного підходу лежить подання даних у вигляді багатовимірних гіперкубів, при цьому передбачається, що всередині такого гіперкуба немає пустот, тобто всі чарунки куба завжди заповнені. Це пов'язане з тим, що дані в них, як правило, зберігаються у вигляді безлічі логічно упорядкованих блоків (масивів), мають фіксовану довжину, причому саме блок є мінімальною індексованою одиницею. І хоча в МСУБД зазвичай передбачається, що блоки, повністю заповнені невизначеними значеннями, не зберігаються, це забезпечує лише часткове вирішення проблеми. Дані в таких системах зберігаються в упорядкованому вигляді, невизначені значення усуваються, і то частково, лише в тому випадку, коли ми за рахунок вибору порядку сортування згрупуємо їх у максимально великі безперервні групи.

Однак таке рішення може безпосередньо вступити в конфлікт з тим, що МСУБД працюють дуже швидко тільки тоді, коли дані в них заздалегідь розсортовані в тому порядку, в якому вони мають бути у відповіді на запит. Але порядок сортування, найчастіше використовуваний у запитах, може не збігатися з тим, який потрібен для максимального усунення невизначених (неіснуючих) значень. У результаті розробник може зіткнутися з важко вирішуваною дилемою:

- пожертвувати швидкодією, але це одна з головних переваг і часто одна з основних причин вибору саме МСУБД;
- пожертвувати зовнішньою пам'яттю, але збільшення обсягу даних також не збільшує швидкодію. Крім того, як уже говорилося, МСУБД нині не пристосовані для роботи з великими обсягами даних.

Таким чином, МСУБД однозначно ефективні тільки за умов виконання двох вимог:

- рівень агрегації даних у БД досить високий і, відповідно, обсяг БД не дуже великий (не більше декількох гігабайтів);
- гранями багатовимірного куба вибрані досить стабільні в часі реквізити (з точки зору незмінності їхніх взаємозв'язків) і, відповідно, число неіснуючих значень відносно невелике.

На жаль, сьогодні відсутні офіційні порівняльні результати тестування продуктивності систем, реалізованих на основі багатовимірних і реляційних баз даних, але при цьому загальна оцінка така: "продуктивність якісно налаштованих реляційних систем при використанні схеми "зірка" цілком порівнянна з продуктивністю систем, реалізованих на основі багатовимірних баз даних".

Глобальне сховище даних. Останнім часом все популярнішою стає ідея поєднати концепції сховища і вітрини даних в одній реалізації та використовувати сховище даних як єдине джерело інтегрованих даних для всіх вітрин даних. Тоді природною стає така трирівнева архітектура системи:

• на *першому рівні* реалізується корпоративне сховище даних на основі однієї з розвинених сучасних реляційних СУБД. Це сховище інтегрованих, в основному, деталізованих даних. Реляційні СУБД забезпечують ефективне збереження та управління даними дуже великого обсягу, але не повністю відповідають потребам OLAP-систем, зокрема у зв'язку з вимогою багатовимірного подання даних;

• *на другому рівні* підтримуються вітрини даних на основі багатовимірної системи управління базами даних (прикладом такої системи є Oracle Express Server). Такі СУБД майже ідеально підходять для розроблення OLAP-систем, але поки що не дають можливість зберігати надвеликі обсяги даних (граничний розмір багатовимірної бази даних становить 10—40 Гбайтів). У такому випадку це і не потрібно, оскільки мова йде про вітрини даних. Зауважимо, що вітрину даних не обов'язково необхідно повністю сформувати. Вона може містити посилання на сховище даних і добирати звідти інформацію в міру надходження запитів. Звичайно, це дещо збільшує час відгуку, але зате знімає проблему обмеженого обсягу багатовимірної бази даних;

• на *третьому рівні* знаходяться клієнтські робочі місця кінцевих користувачів, на яких встановлюються засоби оперативного аналізу даних.

Список використаної літератури

1. *Асєєв Г. Г.* Архітектура корпоративного сховища даних / Георгій Асєєв // Вісник Книжкової палати. — 2010. — № 10 — С. 20—25.
2. *Сахаров А. А.* Принципы проектирования и использования многомерных баз данных (на примере Oracle Express Server) / А. А. Сахаров // СУБД. — 2006. — № 3. — С. 44—59.
3. *Codd E. F.* Providing OLAP (On—Line Analytical Processing) to User—Analysts: An IT Mandate / S. B. Codd., C. T. Salley. — E. F. Codd & Associates, 1993. — 24 p.
4. *Demarest M.* Building the Data Mart / M. Demarest // DBMS. — 1994. — № 7. — P. 44—50.
5. *Кузнецов С.* Обзор возможностей применения ведущих СУБД для построения хранилищ данных / С. Кузнецов (DataWarehouse). — Режим доступа: http://www.olap.ru/basic/dbms.asp. — Загл. с экрана.

Рассмотрены витрины данных, которые созданы для информационного обеспечения аналитических заданий конкретных управленческих подразделений компаний и организаций. Освещены вопросы использования многоуровневых решений в хранилищах данных. Проанализирована идея объединения концепции хранилища и витрины данных в одной реализации, а также возможности использования хранилища данных как единого источника интегрированных данных всех витрин данных.

Data marts, which are developed for information support of analytical tasks of specific administrative subdivisions of companies and organizations, are examined. The problems of application of multi level solutions in the database management system of data warehouses are elucidated. The idea of integration of the conception of data warehouse and data mart in one realization and possibilities of using data warehouse as a single source of integrated data for all data marts are analyzed.

Надійшла в редакцію 18 жовтня 2011 року