УДК 681.3

ПІДВИЩЕННЯ АДАПТИВНОСТІ ПРОГРАМНОЇ СИСТЕМИ ШЛЯХОМ ВДОСКОНАЛЕННЯ ОРГАНІЗАЦІЇ ДАНИХ В УНІВЕРСАЛЬНИХ СХОВИЩАХ ДАНИХ

І.В. Федоров
Інститут програмних систем НАНУ
e-mail: fedorov@rgdata.com.ua

Інформація являється однією з найважливіших цінностей сучасної організації. Об'єми даних, які зберігають і оброблюють навіть не найбільш великі організації, вимірюється сотнями гігабайт і продовжують збільшуватися. Однак дані мають реальну цінність тільки в тому випадку, якщо вони були отримані в потрібні строки. Від того, як швидко користувачі отримають потрібну інформацію, необхідну для прийняття стратегічних рішень, залежить ефективність їх роботи. Тому для повноцінного функціонування систем підтримки прийняття рішень досить актуальною є задача забезпечення ефективної організації даних. При цьому потрібно врахувати, що для прийняття стратегічних рішень потрібно мати доступ до всіх робочих даних організації. Зведена інформація про бізнес разом з інтелектуальними засобами аналізу даних дозволить виконувати нерегламентовані запити, що в свою чергу надасть змогу мати потужний інструмент для прийняття рішень. Саме цим можна пояснити готовність організацій вкладати чималі кошти на створення та підтримку сховищ даних.

В процесі створення системи прийняття рішень виникає необхідність забезпечення адаптивності цієї системи до зміни зовнішніх та внутрішніх умов. Адаптивність означає, що система, яка проектується, має здатність легко самозмінюватися для збереження своїх функціональних показників у заданих межах. Адаптивність в архітектурі системи підтримується за допомогою створення універсальної структури сховища даних, яка буде підходити для багатьох проектів, дані яких вписуються у деяку схему. Організація адаптивного сценарію роботи користувачів для показників, отриманих при використанні такої системи та їх аналіз надасть можливість удосконалити технологію збору даних.

В сховище поміщаються дані, які витягнуті з робочих даних і перетворені для прийняття рішень кінцевими користувачами. Дані являються самостійним ресурсом, який потребує надійного зберігання і централізованого управління, до того ж розділяються різнорідними застосуваннями і нерідко мають життєвий цикл, який по тривалості може перевищувати комп'ютерні платформи. У великій організації вирішення такої проблеми при збереженні витрат на прийнятному рівні - досить складна проблема, яка потребує грамотного підходу до проектування сховищ даних. Невдалі проектні рішення можуть призвести до значного подовження і ускладнення циклу розробки і в підсумку часто є причиною низької якості готової системи. Проблеми продуктивності в ній дуже часто не можна вирішити простим нарощуванням потужності апаратних засобів. Головною задачею стає задача вдалого проектування, яка полягає в забезпеченні на момент запуску системи і на всьому протязі її експлуатації:

- функціональності і адаптованості;
- пропускної здатності;
- часу реакції;
- готовності;
- простоти експлуатації;
- безпеки.

Проектування моделі сховища даних

Ретельне проектування – запорука успішної реалізації проекту створення сховища даних. Багато традиційних методів проектування непридатні до сховищ даних, оскільки припускають, що проектувальник хоче добитися оптимальної роботи для системи чи поновлення (дані в сховищі ніколи не поновлюються безпосередньо, а тільки побічно –

шляхом прийому в завантажуючу секцію нових даних). Такі традиційні методи, як нормалізація даних, основані в першу чергу на тому, щоб запобігати надмірності даних, а також збалансувати вимоги до виконання запитів і вимоги до виконання DML-операцій. В процесі нормалізації породжується велика кількість таблиць. Однак велика кількість маленьких таблиць, як правило, є повною протилежністю тому, що найкраще усього обслуговує сховище даних. По цій причині для сховищ даних непридатні деякі ключові принципи традиційного реляційного проектування. Традиційні методи моделювання даних і проектування баз даних, як правило, не дозволяють отримати структури даних, оптимізовані для масових запитів, які можливо використовувати не тільки для підтримки адаптивності організації даних, але й для забезпечення потрібного трафіку повідомлень і зменшення навантаження на сервер, а також часу відповіді. Тому замість традиційних ідей проектування розроблюються більш ефективні методики.

Все більше уваги сучасні організації приділяють забезпеченню здатності сховища даних до нарощування функціональності із збереженням всіх існуючих можливостей. Забезпечення адаптивної організації даних являється однією з ключових бізнес-вимог при оцінці ефективності проектування сховища даних. При виникненні даної проблеми було вирішено забезпечити адаптивність за допомогою створення шаблону, який буде являтися універсальним рішенням для цілого ряду проектів. Це дозволить уникнути багаторазового проектування, на що витрачається велика кількість часу та ресурсів.

Досить ефективним підходом для створення потрібного шаблону являється його проектування на основі моделі даних, яка представляє собою зіркообразну схему. Основним критерієм вибору даної схеми є зручність її використання у застосуваннях для сховищ даних у випадку, коли кожний вимір поміщається в одній таблиці. Це дасть змогу оптимізувати виконання запитів. Для забезпечення оптимізації в запит до зіркообразної схеми включаються детальні відомості про точки цієї схеми (вимір), а потім дані, які відповідають цим точкам узагальнюються. Проводиться декілька таких ітерацій з метою виявлення тенденцій зміни даних. Для цього декілька вимірів тримаються постійними, а над іншими робляться зміни або з'єднання таблиці фактів з іншими таблицями для того, щоб побачити кореляцію між таблицею фактів та вимірами, не вказаними в умові запита. Використання зіркообразної схеми спрощує потрібні для неї запити. В запитах таблиці вимірів не будуть з'єднуватися між собою – це ознака хорошого проектування сховища.

Створений шаблон для побудови сховища даних дозволяє йому містити будь-яку кількість таблиць фактів і вимірів, необхідну для реалізації конкретного проекту, але вони повинні мати однакову внутрішню структуру згідно технічних вимог. Центральна таблиця фактів з'єднується з декількома довідковими таблицями, кожна з яких з'єднується з таблицею фактів, але ніде не з'єднується з іншими довідковими таблицями. З'єднання по схемі "зірка" націлене на використання саме такої відсутності взаємозв'язків – спочатку формується декартовий добуток всіх потрібних строк з довідкових таблиць, а потім виконується його з'єднання с таблицею фактів.

Зіркообразна модель складається із центральної таблиці фактів, яка оточена декількома таблицями вимірів. В таку схему в цілях економії часу на виконання запитів записуються як робочі дані із операційних баз даних, так і аналітичні, які отримані шляхом обчислень, дані. Посилальні і довідкові дані представляються в декодованому виді.

Таблиця фактів являється основною таблицею сховища даних. Вона містить відомості про об'єкти чи події, сукупність яких буде в подальшому аналізуватися. Факти представляють собою основні види бізнес-діяльності організації і фактори, які впливають на даний бізнес чи його сектор. Існує чотири типа фактів, які найбільш часто зустрічаються. До них відносяться:

- факти, які зв'язані с транзакціями. Вони засновані на окремих подіях (типовими прикладами яких є телефонний дзвінок чи зняття грошей с рахунку за допомогою банкомату);

- факти, які зв'язані з <моментальними знімками>. Вони базуються на стані об'єкта (наприклад, банківського рахунку) в певні моменти часу, наприклад на кінець дня або місяця. Типовими прикладами таких фактів є об'єм продаж за день або денний виторг;
- факти, які зв'язані с елементами документа. Базуються на тому чи іншому документі (наприклад, рахунку за товар або послуги) і містить докладну інформацію про елементи цього документа (наприклад, про кількість, ціну, відсоток знижки);
- факти, які зв'язані с подіями або станом об'єкта. Представляють виникнення події без подробиць про неї (наприклад, просто факт продажу або факт відсутності такової без інших подробиць).

В таблиці фактів не міститься ніяких відомостей про те, як групувати записи при обчисленні агрегатних даних. Ці відомості, в подальшому використовуються для побудови ієрархій у вимірах і містяться в таблицях вимірів. В таблиці фактів заносяться детальні дані, які відповідають членам нижніх рівнів ієрархії відповідних вимірів.
Фізично таблиця фактів представляє собою декілька секційованих таблиць. Термін "секціонування" дуже широко використовується спеціалістами по базам даних і в загальному випадку означає формування підмножин даних. Таблиці фактів в сховищі мають джерело даних, яке піддається інтенсивним транзакціям. Секціонування було проведено з метою уникнути завантаження даних в одну таблицю, що дозволило уникнути ряду недоліків, а саме:

- Дуже скоро таблиця стане дуже великою, що ускладнить її адміністрування і управління.
- В процесі завантаження даних ця таблиця може виявитися недоступною для користувачів, які виконують запити. Це залежить від того, які методи будуть застосовуватися для завантаження.
- Якщо не створити спеціальну утиліту очистки і не запускати її періодично, то відділити і викинути дані, які застаріли і втратили свою актуальність, буде складно.

Щоб вирішити ці проблеми, вирішено секціонувати таблицю фактів на менші таблиці. Секціонування не тільки дає позитивний вплив на процес завантаження, але й впливає на запити даних. На перший погляд, при цьому повинні існувати проблеми. Наприклад, щоб запросити чи узагальнити дані потрібно об'єднати багато таблиць. Але при цьому може виявитися, що таке об'єднання важке для програмування і неефективне в роботі. З його громіздкістю можна справитися, якщо визначити представлення, в яких і будуть відбуватися великі об'єднання таблиць (при цьому обов'язково потрібно застосовувати операцію UNION ALL, а не UNION).

Складні засоби запитів дозволяють створити метамодель даних, яка описує відношення між секційними таблицями, які визначають таблицю фактів. При цьому необхідно дотримуватися деяких умов – представлення повинно будуватися на "чистих" секціях, тобто логічні визначення секцій повинні бути повністю ідентичні. Можливо проводити секціонування і більш ніж по одному атрибуту, але цей метод іде всупереч ідеї спрощення структури і породжує складності при комбінуванні таблиць.

В таблицю фактів поміщається унікальний складовий ключ, який об'єднує первинні ключі таблиць вимірів. Це цілочислові значення, оскільки таблиця фактів може містити сотні тисяч записів і зберігати в ній текстові описи, які повторюються, невигідно. Надмірність інформації негативно позначиться на продуктивності системи в цілому і ускладнить розуміння схеми даних, тим самим знижуючи ефективність управління.
Набагато краще помістити записи в менші по об'єму таблиці вимірів. При цьому, як ключові, так і деякі неключові поля відповідають вимірам.

Кожний вимір заповнюється значеннями з відповідних таблиць вимірів сховища даних. Таблиці вимірів містять дані, які не змінюються або змінюються рідко. Ці дані представляють собою по одному запису для кожного члена нижнього рівня ієрархії у вимірі. В таблицях вимірів поміщається одне описове поле з ім'ям члена виміру і цілочисельні ключові поля, які являються сурогатними ключами для однозначної ідентифікації члена виміру. В схемі

створюються виміри, які базуються на певній таблиці вимірів і містять ієрархію. В таблицю вимірів поміщаються поля, які вказують на "батька" даного члена в цій ієрархії, а також додаткові атрибути членів вимірів, які знаходяться в вихідній оперативній базі даних. Кожна таблиця вимірів знаходиться у відношенні "один до багатьох" з таблицею фактів. Додаткові таблиці вимірів в створеній схемі відповідають верхнім рівням ієрархії виміру і знаходяться вони у відношенні "один до багатьох" в головній таблиці вимірів, яка відповідає нижньому рівню ієрархії.

В описаній моделі сховища даних організована підтримка рівнів вимірів. Для нормального функціонування моделі створюється таблиця для кожного рівня узагальнення кожного атрибута, який призначений для класифікації показників. Будемо називати такі таблиці довідниками вимірів. Рівень виміру являється зовнішнім ключем довідника рівнів, в якому зберігаються всі його можливі значення. Кожний вимір має свій власний словник рівня. Це дозволить забезпечити можливість перегляду показників с різними степенями агрегації, що надає можливість більш ефективно організувати представлення деталізованих даних для користувача системи при вирішенні ним різноманітних бізнес-проблем.

Висновки

В результаті створено модель даних, що може використовуватися при проектуванні сховищ даних, які призначені для оперативного вирішення таких задач, як збір даних, планування, отримання звітності і аналіз інформації. Такі сховища даних здатні консолідувати інформацію із різноманітних корпоративних систем і надавати швидкий доступ до всіх ділових даних для ефективного управління і контролю діяльності багатофіліальної організації та для оперативного обгрунтування прийняття управлінських рішень. Наведена методика реалізації призначена для полегшення роботи, зв'язаної з внесенням змін в структуру бази даних. Це надасть можливість рідше повертатися до питання перепроектування, ніж це буває при традиційному підході, коли ситуація змінюється і потрібно створювати нові таблиці, поля, обмеження і прив'язувати це до вже експлуатованих даних, тобто вирішувати задачу знову - а, як правило, її вирішувати не дуже просто. Забезпечуються набагато менші втрати часу на ретельне проектування сховищ даних. Описана модель може забезпечити ефективну роботу сховищ даних, тобто забезпечити якомога менше використання ресурсів на ведення сховищ даних, які, як правило, представляють собою часто дуже великі бази даних, а також підвищити в них швидкість обробки даних та досягти більш оптимального виконання запитів.

Список літератури

1. Ігнатенко П.П. Проблеми забезпечення життєздатності програмних систем та підходи до їх вирішення// Пробл. програм.- 2002. - № 3-4. – с. 58 – 73.

2. Калашник В.І. Правила відображення візуальної моделі предметної області у схему реляційної бази даних // Вісн. Черніг. технол. і-ту, 1997. - №2, С. 67-73.

3. W.H. Inmon, J.D. Welch, Katherine L.Glassey, Managing the Data Warehouse. Wiley Computing Publishing, 1997

4. Oracle Method.Custom Development. Data Warehouse Method Handbook, Release 1.0.0, 1996 Oracle Corporation.

5. Сахаров А.А. Принципи проектування і використання багатомірних баз даних (на прикладі Oracle Express Server). СУБД, №3/1996.

6. Кирилов В.В. Основи проектування реляційних баз даних. http://www.citforum.ru/database/index.shtml.

7. Ульман Дж. Основи систем баз даних. – М.: Фінанси і статистика, 1983. – 334 с.