УДК 004.657

К.т.н., доцент Орлова М.М., магістрант Валієва К.В.

**Національний технічний університет України
«Київський політехнічний інститут»**

ОПТИМІЗАЦІЯ *SQL*-ЗАПИТІВ ЗА ДОПОМОГОЮ «МАТЕРІАЛІЗОВАНИХ ПОДАНЬ»

Вступ

Сьогодні бази даних містять дуже великі обсяги інформації (фактів), яка необхідна для прийняття своєчасних та обґрунтованих рішень. Однак, для перетворення фактів у бізнес-знання часто необхідно виконувати складні запити, які підсумовують дані з декількох підлеглих (*detail*) таблиць, що потребує виконання великого обсягу обчислень. До нещодавно єдиним рішенням цієї проблеми було використання зведених таблиць у сховищі даних, але цей підхід має істотні недоліки: по-перше, складність синхронізації змін в підлеглих таблицях, і по-друге, необхідність у поінформованості всіх користувачів щодо їх коректного використання.

Для вирішення цих проблем запропоновано нове рішення - об’єкт схеми даних «матеріалізоване подання». Основні аспекти реалізації та використання цього рішення розкриті у працях [1], [3] та [4].

Постановка задачі

Основною метою статті є аналіз і наступна оптимізація використання «матеріалізованих подань» за критерієм підвищення швидкості обробки *SQL*-запитів. Для досягнення цієї мети вирішуються наступні задачі:

- аналіз використання зведених таблиць,
- аналіз використання «матеріалізованих подань»,
- дослідження отриманих результатів.

Запити для отримання підсумкових даних

Одними з найбільш використовуваних і корисних типів запитів у сховищах даних є запити на отримання підсумкових (зведених) даних, таких як сума, середнє арифметичне значення або загальна кількість. Ці запити є дуже важливими, оскільки поєднують детальну інформацію підлеглих таблиць і представляють їх у зручній формі, яка допомагає користувачам у прийнятті обґрунтованих рішень.

На жаль, для отримання підсумкових даних можуть знадобитися істотні накладні витрати при обробці даних, оскільки при цьому опрацьовуються

величезні підлеглі таблиці в сховищі даних. При виконанні цієї операції витрачається багато часу.

Розглянемо процес отримання підсумкових даних на наступному прикладі. Нехай існує база даних для ведення замовлень на закупівлю товарів. Вона містить таблиці *PARTS, ORDERS* та *ITEMS, у* яких зберігається інформація про товари, замовлення і пункти замовлення відповідно. Створимо запит 1, який обчислює загальну суму кожного замовлення:

```
SELECT i.ord_ord_id AS order_id,
MAX(TO_CHAR(orderdate,'MonthYYYY')) AS orderdate,
SUM(i.quantity * p.unitprice) AS total
FROM orders o, items i, parts p
WHERE o.ord_id = i.ord_ord_id
AND p.part_id = i.part_part_id
GROUP BY i.ord_ord_id;
```

Результатом виконання даного запиту є таблиця, яка містить номер замовлення, його дату та загальну суму (табл. 1).

Таблиця 1. Результат виконання запиту 1

ORDER_ID	***ORDERDATE***	***TOTAL***
1	Лютий 2009	1234.22
2	Лютий 2009	5678.33
3	Лютий 2009	9012.44
4	Лютий 2009	3456.55
. . .		

Скористаємося командою *EXPLAIN PLAN для* отримання плану виконання запиту. За її допомогою отримаємо загальну вартість (*COST*, відносне значення, яке оцінюється оптимізатором) запиту 1 – 1655.

Використання зведених таблиць

Практика показує, що коли підлеглі таблиці містять багато записів, вартість запитів для отримання підсумкових даних є досить істотною. Для уникнення витрат на обчислення підсумкової інформації при багаторазових запитах до неї, у сховищах даних створюються зведені таблиці. Ці таблиці зберігають результати одного або декількох заздалегідь обчислених запитів, які найчастіше виконуються у сховищі даних. Наприклад, для нашого випадку можна створити таблицю *SALES_SUMMARY*, що зберігатиме заздалегідь обчислений результат запиту 1. Після створення цієї таблиці загальна сума кожного замовлення може бути обчислена за допомогою наступного запиту.

Запит 2:

```
SELECT order_id, orderdate, total
FROM sales_summary;
```

Результат виконання цього запиту буде аналогічним результату виконання запиту 1, але при цьому загальна вартість буде істотно меншою – 31 проти 1655 у першому випадку.

Як видно з розглянутого прикладу, зведені таблиці можуть допомогти прискорити виконання запитів для отримання підсумкових даних. Але, натомість, вони спричиняють наступні проблеми:

- Адміністраторам баз даних доводиться вручну оновлювати кожну зведену таблицю й пов'язані з нею індекси після завантаження нових даних у відповідні підлеглі таблиці, - у такий спосіб зведені таблиці синхронізуються з підлеглими таблицями, і забезпечуються правильні результати запиту.
- Розробники повинні бути поінформовані про всі існуючі зведені таблиці, і розуміти, коли й для яких запитів потрібно їх використовувати.

Використання «матеріалізованих подань»

Для запобігання проблемам, пов’язаних з використанням зведених таблиць запропоновано нове рішення – об’єкт схеми «матеріалізоване подання». Це спеціальний вид подання, що фізично існує в базі даних. У це подання можуть бути включені з'єднання й/або складені значення (агрегати). Можна сказати, що воно існує для зменшення часу виконання запиту шляхом попереднього виконання дорогих з'єднань і операцій агрегування до їхнього використання в реальному запиті. Цей підхід реалізовано в *Oracle8i* і для ефективної роботи з ним було додано наступний функціонал:

- Розширення можливостей оптимізатора, що дозволяє йому автоматично переписувати запит таким чином, щоб він використовував «матеріалізовані подання»;
- Вбудований пакет *DBMS_MVIEW*, що включає процедури відновлення й інші процедури для управління «матеріалізованими поданнями»;
- Засіб для аналізу ефективності існуючих і потенційних «матеріалізованих подань»(*SQL Access Advisor*).

До того ж, для *Oracle* надає наступні методи оновлення та синхронізації «матеріалізованих подань»:

- повне;
- швидке (застосовуються тільки зміни);
- примусове: якщо це можливо, виконується швидке відновлення, інакше - повне відновлення ;
- з використанням механізму відстеження змін розділів (швидке відновлення рядків тільки в змінених розділах)

Наведемо приклад створення «матеріалізованого подання», що зберігатиме результат запиту 1:

```
CREATE MATERIALIZED VIEW mv_sales_summary
BUILD IMMEDIATE
REFRESHCOMPLETE
ON DEMAND
ENABLE QUERY REWRITE
AS
SELECT i.ord_ord_id AS order_id,
```

MAX(TO_CHAR(orderdate,'MonthYYYY')) AS orderdate,
*SUM(i.quantity * p.unitprice) AS total*
FROM orders o, items i, parts p
WHERE o.ord_id = i.ord_ord_id
AND p.part_id = i.part_part_id
GROUP BY i.ord_ord_id;

Після створення «матеріалізованого подання» результат виконання запиту 1 не зміниться, але загальна вартість буде такою ж як і при використанні зведеної таблиці – 31.

Табл. 2 містить загальні вартості запитів для всіх розглянутих вище випадків.

Таблиця 2. Загальні вартості запитів

Випадок	**Вартість**
1. Запит 1	354
2. Запит 2 (Запит 1 попередньо збережений у зведеній таблиці)	31
3. Запит 1 без змін, зі створенням «матеріалізованого подання»	31

Як видно з табл. 2, за допомогою «матеріальних подань» можна зменшити час виконання запиту.

Висновки

«Матеріалізовані подання» дозволяють підвищити продуктивність у сховищах даних або в базах даних. Вони допомагають зменшити час відповіді на запит, причому для цього не потрібно його змінювати – механізм перезапису запитів *Oracle* автоматично перезапише *SQL*-запити таким чином, щоб у них використовувалися «матеріалізовані подання», усуваючи потребу кінцевого користувача знати про існування підсумкових/зведених даних. До того ж, завдяки механізму оновлення/синхронізації «матеріалізовані подання» не потребують додаткових операцій для їх актуалізації.

Література

1. . *Nanda.* Room with a Better View / Arup Nanda. // Oracle Magazine. - 2003. – №2. – P. 62-68.
2. *A. B. Danchenkov, D. Burleson.* Oracle Tuning: The Definitive Reference / Alexey B. Danchenkov, Don Burleson. – New York : Library of Congress. – 2006. – P. 980.
3. *L. Hobbs*. Oracle Materialized Views & Query Rewrite. An Oracle White Paper / Lilian Hobbs. - Redwood Shores : Oracle Press. – 2005. – P. 24.
4. *С. Бобровски*. Использование материализованных представлений для ускорения запросов / Стив Бобровски. // Oracle Magazine RE. – 1999. – №5. - С. 43-51.