Компьютерная Математика

УДК 004.652

Б.Е. ПАНЧЕНКО, Е.В. КРЮЧКО

ИССЛЕДОВАНИЕ ЭЛЕМЕНТОВ OLAP-РЕШЕНИЙ НА РЕЛЯЦИОННОМ КАРКАСЕ

Экспериментально исследован алгоритм фонового формирования OLAP итогов в БД, построенной на реляционном каркасе. Делается вывод о возможности использования подхода в единой каркасной БД и как оперативном, и как архивном хранилище. Приводятся результаты численного эксперимента доступа к данным.

Вступление. Системы автоматизации современных предприятий содержат базы данных (БД), позволяющие проводить комплексный анализ информации, что, в конечном итоге, дает возможность принимать оптимизированные решения. Такие приложения получили название систем поддержки принятия решений.

Для принятия корректного управленческого решения необходима комплексная информация. Однако подходы, развивающиеся для ее формирования последнее время, основаны на избыточности: на первом этапе создается оперативная БД, которой позволено быть «неочищенной». А на втором – создается хранилище данных (ХД) [1], в котором осуществляется отсеивание и рафинирование данных для их использования для аналитической работы и создания соответствующих отчетов.

Ральф Кимбалл описал ХД как «место, где имеется возможность получения доступа к своим данным» [2]. Там же сформулированы и основные требования к ХД: поддержка высокой скорости получения данных, поддержка внутренней непротиворечивости, возможность получения и сравнения срезов данных, наличие удобных утилит просмотра данных, полнота и достоверность хранимых данных и поддержка качественного процесса пополнения данных.

Очевидно, что этому перечню требований может соответствовать единая и единст-

венная БД, предоставляющая пользователю
и оперативные, и исторические данные.

Более того, потребность пользователей в таком объединении является общеизвестной. В работе [3] эта концепция описана в разделах «исчезновение отдельных хранилищ данных» и «хранилища данных в режиме реального времени». А ранее аналогичная потребность исследована и предложена в [4]. Дальнейшие внедрения данного подхода разными пользователями подтвердили такой вывод.

Постановка задачи. Большинство авторов утверждает, что удовлетворять всем перечисленным требованиям в рамках одного и того же продукта зачастую не удается [5, 6]. Поэтому для реализации ХД обычно используется несколько продуктов, одни из которых представляют собой собственно средства хранения данных, другие – средства их извлечения и просмотра, третьи – средства их пополнения и т. д.

Во-первых, оперативные БД предназначены для того, чтобы помочь пользователям выполнять повседневную работу, тогда как ХД предназначены для принятия решений. Например, продажа товара и выписка счета производятся с использованием БД, предназначенной для обработки транзакций, а анализ динамики продаж за несколько лет, позволяющий спланировать работу с поставщиками, – с помощью ХД.

Во-вторых, оперативные БД подвержены постоянным изменениям в процессе работы пользователей, а ХД относительно стабильно: данные в нем обычно обновляются согласно расписанию (например, еженедельно, ежедневно или ежечасно – в зависимости от потребностей). В идеале процесс пополнения представляет собой просто добавление новых данных за определенный период времени без изменения прежней информации, уже находящейся в хранилище.

В-третьих, оперативные БД чаще всего являются источником данных, попадающих в хранилище. Кроме того, хранилище может пополняться за счет внешних источников, например статистических отчетов.

Однако, приложения, построенные на каркасной модели данных, позволяют решить все указанные задачи на единой БД. Цель настоящей работы – исследовать основные OLAP-элементы в фоновом он-лайн режиме. И тем самым подтвердить правильность выводов о том, что объединение свойств оперативной и архивной БД в едином приложении не только возможно, но и более перспективно.

OLAP-подход на каркасе и механизм on-line-транзакций. При обслуживании современных БД основным вопросом является скорость реакции системы. Такие бизнес-приложения, как ретроспективный анализ деятельности компании, анализ рынка, стратегическое планирование и прогнозирование и т. п. характеризуются необходимостью извлекать большое число записей из очень больших наборов данных и вычислять на их основе с максимально возможной скоростью разнообразные итоговые показатели. Предоставление таких сервисов является основным назначением всех OLAP-инструментов [6], работающих с ХД.

Международный комитет по стандартизации и международная электротехническая комиссия ISO/IEC в 2001 году расширили известный стандарт SQL-подобных языков. Был добавлен пакет специфических OLAP-функций. Как основные решения стандарт [7] задекларировал функции GROUPING SETS, CUBE BY и ROLLUP BY. Каждая из них позволяет получать произвольные выборки данных для ответов на запросы пользователей.

Однако основной недостаток пост-запросного подхода – это плохое распараллеливание вычислений на больших объемах данных и, как следствие, низкая производительность системы в целом. Подавляющее большинство запросов не является случайными, а предопределено спецификой ПрО. Но механизмы учета данного факта используются не в полной мере.

Альтернативной является методика использования реального времени, предложенная в [4] (в 1994 году), где в фоновых отношениях все необходимые базовые агрегаты данных, такие как, например, итоги и подитоги, формируются по факту ввода данных или редактирования. При этом процесс распараллеливания осуществляется автоматически специализированной буферизацией транзакций. А за счет механизма индексирования скорость формирования фоновых отношений максимально возможная в рамках используемой операционной среды.

Важное свойство каркасной OLAP-методики – это более естественный механизм отслеживания целостности данных, когда единицей атомарности является каскад данных, сформированный по всей совокупности отношений. Протоколирование таких транзакций значительно упрощается. И механизмы восстановления потерянных групп при сбоях в системе сводятся к простому копированию.

Несложно показать, что при агрегировании значений шунтирующих атрибутов «нижних» каркасных отношений и фиксации единственного агрегированного значения в качестве шунтирующего атрибута в одном кортеже «верхнего» каркасного отношения, появление в этом кортеже дополнительных, необусловленных ограничениями на домены и ключи, функциональных зависимостей (ФЗ) между атрибутами невозможно.

В работе [8] для монотонного агрегирования данное утверждение доказано в виде леммы. Атрибут S каркасного отношения со схемой $R(X, S)$, где X – ключ, полученный произвольной монотонной функцией $F(R_i(X_i, A_{ij}))$ от произвольной совокупности $A_{i,j}$ неключевых атрибутов иных каркасных отношений R_i, каждое из которых удовлетворяет только особым ограничениям [9], не является детерминантом отношения R.

Необходимо отметить, что использование в одном кортеже нескольких агрегированных атрибутов от разных аргументов агрегирования не приводит отношения к денормализации, так как никаких «паразитных» ФЗ между агрегатами не возникает. Но следует также и то, что использование в одном кортеже нескольких агрегированных атрибутов от одного аргумента агрегирования может приводить к денормализации. В зависимости от специфики функций агрегирования в таком отношении могут появиться транзитивные ФЗ в неключевых шунтирующих агрегированных атрибутах.

С одной стороны, транзитивные ФЗ, появляющиеся между несколькими агрегированными шунтирующими атрибутами от общего аргумента агрегирования, не являются критическими, так как устраняются обычной декомпозицией на несколько каркасных отношений, каждое из которых может хранить необходимый единственный агрегат. Но с другой стороны, хранение значительного числа результатов агрегирования от громоздких функций может привести к значительным временным затратам на операции по отслеживанию целостности этих данных. Поэтому решение проектировщик принимает исходя из статистики запросов.

Для выполнения основных OLAP-функций в каркасной модели использован механизм пошаговых параллельных транзакций, построенный на каскадных функциях, аналогичных перечню стандартизованных [7].

Благодаря интеграции рабочих станций в распределенную среду становится возможным более эффективное распределение функций в ней, когда прикладные программы выполняются на рабочих станциях, называемых серверами приложений, а БД обслуживаются выделенными компьютерами, называемыми их серверами. Это служит источником развития таких распределенных архитектур, где в роли узлов выступают не просто компьютеры общего назначения, а специализированные серверы.

Параллельный компьютер (мультипроцессор) сам по себе является распределенной системой, составленной из узлов (процессоров, компонентов памяти), соединенных быстрой сетью внутри общего корпуса. Технология распределенных БД может быть естественным образом пересмотрена и распространена на системы параллельных компьютеров [10].

Данная методика позволяет значительно усовершенствовать известный механизм унификации часто повторяющихся запросов. И «он-лайн»-формирование фоновых OLAP-итогов осуществлять на автоматизировано расширяющемся поле фоновых таблиц, учитывающих появляющиеся в системе запросы.

При классическом OLAP-подходе может быть выбрана, например, схема «снежинка» [2], где отношения-«факты» содержат оперативные данные, а множественные отношения с «измерениями», присоединенные к «фактам», показывают, как оперативные данные могут агрегироваться. Такой OLAP-механизм вынуждает пользователя тратить достаточно большой промежуток времени на ожидание ответа на запрос – от нескольких часов до нескольких суток. Но специфика работы современной ПрО чаще всего требует мгновенного анализа.

Иной результат дает OLAP-подход, основанный на режиме реального времени в каркасной схеме БД, при котором скорость получения данных сравнима со скоростью индексной выборки из одной таблицы по составному ключевому полю – менее 1 секунды.

Как показала практика внедрений CASE-оболочки SWS [11], значительное число пользователей расчетные данные формирует посредством заполнения списка суммирующих полей.

Алгоритм этой унифицированной процедуры такой.

1. Агрегирование может быть произведено только в поля либо текущего отношения, для которого описывается обращение, либо в поля иерархически «верхних» отношений, объявленных как фоновые главные. Очевидно, что в поля кортежей иерархически подчиненных отношений производить суммирование бессмысленно.

2. При каждом вводе данных в любое поле (или редактировании любого поля) текущего кортежа активизируется описываемая процедура инициализации суммирующих полей.

3. Специфика накопления следующая – если данные в суммирующих полях существуют, то на первом шаге производится арифметическое вычитание указанных в этих колонках выражений с их аргументами до момента возбуждения (ввода в поле, редактирования поля). А на втором шаге производится арифметическое прибавление указанных в этих колонках выражений, но уже с участием измененных аргументов (после момента возбуждения). Такой алгоритм позволяет динамически поддерживать целостность вычисляемых данных как при вводе, так и при редактировании полей с существующими данными.

4. Суммирование производится по порядку очередности списка сверху вниз – каждая последующая строка использует выражения с участием суммирующих полей с уже просчитанными данными из строк выше.

Результаты экспериментальных исследований. Рассмотрим подробнее элементы синхронного формирования OLAP-данных на примере каркасной схемы БД, моделирующей ведение журналов начислений по абонентам из ПрО «Горгаз» [12].

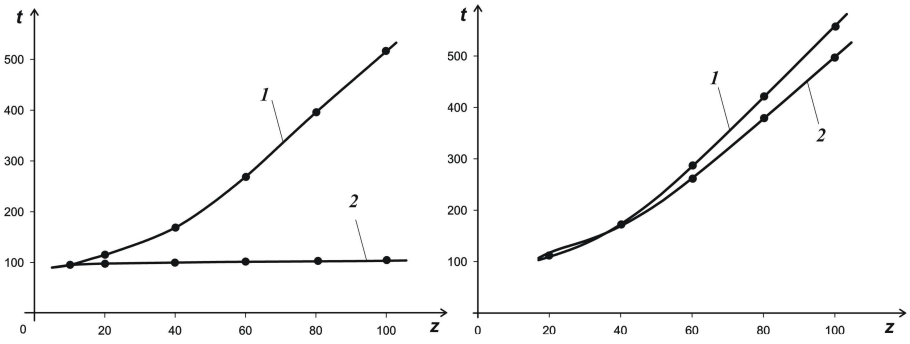
Проведен численный анализ скорости доступа к данным при пост-обработке и в on-line-режиме. Заметное уменьшение времени получения итогов в режиме реального времени (более 10 %) наблюдается для БД, состоящей из 100 отношений, уже при значении числа кортежей около 10^5 в половине отношений. При этом время задержки выполнения каждой транзакции при 10 параллельных процессах и более 100 дополнительных теневых таблиц, в которых по факту появления каждой новой атомарной группы данных в режиме реального времени формируются всевозможные статистические итоги, составляющие не более 0,1 сек. на каждом рабочем месте.

При увеличении основных характеристик системы – количества хранимых данных, количества пользователей системы, количества формирующихся в реальном времени фоновых таблиц, эффективность параллельной обработки значительно повышается. Хотя при этом может наблюдаться незначительное увеличение времени задержки ввода данных на каждом рабочем месте, такие задержки не являются критичными.

Процедура фонового формирования отношений следующая: при введении оператором в поле *СУММА* данных об оплате некоторым абонентом суммы за определенный счет, срабатывает унифицированная каскадная функция, которая пересчитывает итоговые данные во всех фоновых главных отношениях.

Исследовано также и время задержки на выполнение арифметической операции на каскаде отношений. При тестировании схемы моделировалось следующее: на протяжении года 200 операторов ежедневно вносят по 1000 кортежей в общее отношение, что в итоге составляет $73 \cdot 10^6$ записей. При этом формируется до 100 фоновых итоговых отношений. И на начальной, и в завершающей фазе общее усредненное время выполнения функции (время задержки системы) с учетом Интернет-трафика не превышает 0,1 сек. на каждом рабочем месте, что не является существенным. Результат полностью согласуется с методикой индексного поиска.

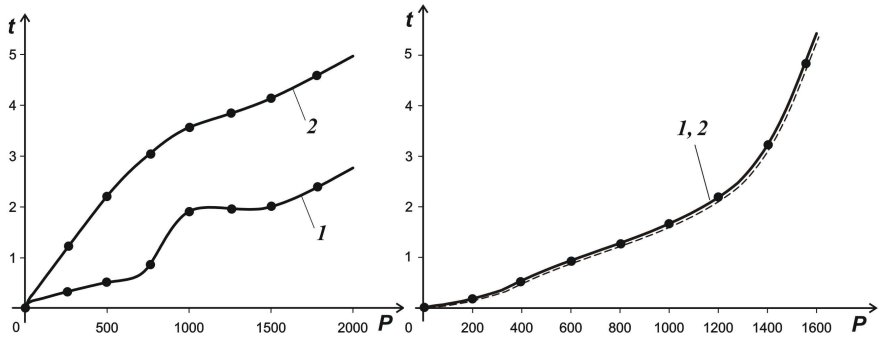
На рис. 1, 2 показаны графики роста времени задержки (в нано- и миллисекундах соответственно) на выполнение процедуры фонового суммирования от числа формируемых отношений для 1 ядра (кривая 1) и 12 ядер (кривая 2) от 100 пользователей. Каждое отношение формируется до 10^7 кортежей. Видно, что распараллеливание процесса формирования агрегированных отношений заметно ускоряет выполнение фоновых процедур. Однако даже без распараллеливания сервер AMD OPTERON X12 с процессором 3,26 GHz на ядро (на 12 ядер), с сервером данных PostgreSQL 8.4 и операционной системой UBUNTU SERVER 12.04, выполнил в описанном эксперименте транзакцию для 100 таблиц за 400 нСек. – более чем удовлетворительное время. Все остальные затраты времени приходятся на работу глобальной сети. Использована SIMD-архитектура с общей памятью. Для MIMD-архитектуры эксперименты не проводились.



а б

РИС. 1. Зависимость времени от числа фоновых отношений

На рис. 1, а – показана зависимость времени выполнения параллельных операций, когда результат следующей операции не зависит от результата предыдущей. Видно, что сокращение времени транзакций прямо пропорционально числу ядер. И общий прирост времени с увеличение нагрузки на БД в многоядерной конфигурации незначителен. Каркасная схема БД позволяет применять алгоритмы агрегирования с независимыми операциями; б – та же зависимость, но для схемы БД, где результат следующей операции зависит от результата предыдущей. Очевидно, что последовательные вычисления плохо подаются распараллеливанию. И время выполнения операции зависит только от быстродействия ядер, а не от их числа. Незначительное ускорение многоядерной архитектуры объясняется использованием параллельных процессов для выполнения вспомогательных операций – переноса данных с жесткого диска в ОЗУ, формирование прерываний пользователя, накладные расходы самой СУБД и т. п.



а б

РИС. 2. Зависимость времени от числа фоновых отношений

На рис. 2 показаны графики роста времени задержки (в миллисекундах) на формирование фоновых таблиц от числа пользователей: а – показывает зависимость времени выполнения последовательных операций со 100 (кривая 1) и 500 (кривая 2) фоновыми отношениями от числа пользователей подключенных к БД. Видно, что рост времени выполнения операций зависит не столько от числа формируемых запросом таблиц, сколько от числа пользователей, что объясняется спецификой СУБД. Необходимость обработки прерываний инициатора транзакций и возврат ему данных вызывает дополнительные вычислительные расходы; б – демонстрирует выполнение той же процедуры, но с учетом сетевых издержек на передачу данных. Кривая 1 показывает время на выполнение операции и на отсылку данных клиенту, а кривая 2 – в том числе и время на получение данных и декодирование сетевого запроса на транзакцию. Из графика видно, что время собственно обработки данных очень мало по сравнению со временем передачи-приема запроса. Это говорит о том, что основной причиной задержек в работе с БД является работа сети, а не недостатки алгоритма или вычислительная мощность ПК.

Заключение. В работе экспериментально исследован алгоритм, позволяющий использовать каркасную БД как единое хранилище и для оперативных, и для архивных данных. Показано, что большинство базовых OLAP-решений, таких, как суммарные итоги множества числовых оперативных данных от множества измерений, на каркасной схеме БД можно формировать в режиме реального времени. А приложение БД проектировать с учетом потребностей оперативного анализа и параллельной архитектуры.

При этом если общее число пользователей не превышает 2000, задержка на выполнение таких операций каждым приложением, обслуживающем в реальном времени до 500 таблиц, не является существенной даже в однопроцессорной архитектуре сервера данных и применения глобальной сети. А при использовании распараллеливания и выделенного трафика возможно значительно большее увеличение мощности системы.

Б.Є. Панченко, Є.В. Крючко

ДОСЛІДЖЕННЯ ЕЛЕМЕНТІВ OLAP-РІШЕНЬ НА РЕЛЯЦІЙНОМУ КАРКАСІ

Експериментально досліджено алгоритм фонового формування OLAP підсумків у БД, побудованій на реляційному каркасі. Робиться висновок про можливість використання підходу в єдиній каркасній БД як оперативного та архівного сховища даних. Наведено результати чисельного експерименту доступу до даних.

B.E. Panchenko, E.V. Kruchko

INVESTIGATION OF ELEMENTS OF OLAP-SOLUTIONS
ON A RELATIONAL FRAMEWORK

An algorithm of background forming of OLAP results in DB built on a relational framework is experimentally investigated. A conclusion is made about the possibility of using the approach in a single framework DB of both OLAP and Data Warehouse. The results of a numerical experiment of data access are given.

1. *Inmon W.H.* Building the Data Warehouse // John Willey & Sons, New York. – 2002. – 412 p.
2. *Kimball R.* The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses // John Willey & Sons, New York. – 1996. – 491 p.
3. *Хоббс Л., Хилсон С., Лоуренд Ш.* Разработка и эксплуатация хранилищ данных (Oracle 9iR2). – М.: Кудиц-образ, 2004. – 586 с.
4. *Панченко Б.Е., Гайдабрус В.Н., Церковицкий С.Л.* Сетевые вычислительные комплексы // Компьютеры плюс программы. – Киев, 1994. – С. 30 – 37.
5. *Кузнецов С.Д., Артемьев В. А.* Обзор возможностей применения ведущих СУБД для построения хранилищ данных (DataWarehouse) // http://citforum.ck.ua/database/kbd98/glava15.shtml
6. *Федоров А., Елманова Н.* Введение в OLAP. – М.: Диалог-МИФИ, 2002. – 268 с.
7. *International* Standart 9075, Database Language SQL, AMENDMENT 1: On-Line Analytical Processing (SQL/OLAP), ISO/IEC, 2001.
8. *Панченко Б.Е.* Хранилища данных на реляционном каркасе // Управляющие системы и машины. – 2013. – № 1. – С. 71 – 84.
9. *Панченко Б.Е.* Исследования доменно-ключевой схемы реляционной базы данных // Кибернетика и системный анализ. – 2012. – № 6. – С. 157–172.
10. *Valduries P.* Parallel Database Systems: Open Problems and New Issues // Distributed and Parallel Databases. – April, 1993. – 1(2). – P. 137 – 165.
11. *Панченко Б.Е., Гайдабрус В.Н.* Реляционный каркас и модель CASE-оболочки нового типа // Кибернетика и системный анализ. – 2013. – № 3. – С. 172 – 186.
12. *Панченко Б.Е.* Численный анализ каркасной схемы реляционной базы данных // Компьютерная математика. – 2012. – № 2. – С. 116 – 126.

Получено 22.01.2013

Об авторах:

Панченко Борис Евгеньевич,
кандидат физико-математических наук, старший научный сотрудник
Института кибернетики имени В.М. Глушкова НАН Украины,

Крючко Евгений Витальевич,
аспирант Сумского государственного университета.