OLAP-ТЕХНОЛОГИИ: ОБЗОР РЕШАЕМЫХ ЗАДАЧ И ИССЛЕДОВАНИЙ

Ю. Кудрявцев,
аспирант факультета Вычислительной математики и кибернетики Московского государственного университета

Рассматриваются технологии OLAP (OnLine Analytical Processing), вклюбчающие в себя дефиницию, исторический обзор исследований, обзор бизнес-приложений и существующих проблем при реализации решений подобного класса.

ОПРЕДЕЛЕНИЕ ТЕРМИНА OLAP

Термин OLAP введён в 1993 г. Эдгаром Коддом [Cod93]. Цель OLAP-систем – облегчение решения задач анализа данных. Кодд сформулировал 12 признаков OLAP-данных, и большинство современных OLAP-средств отвечают этим постулатам. Перечислим их:

12 ПРИЗНАКОВ OLAP-ДАННЫХ

Многомерная концепция данных. OLAP оперирует CUBE-данными (см. далее описание работ Грея [GBLP95]), которые являются многомерными массивами. Число измерений OLAP-кубов не ограничено.

Прозрачность. OLAP-системы должны опираться на открытые системы, поддерживающие гетерогенные источники данных.

Доступность. OLAP-системы должны представлять пользователю единую логическую схему данных.

Постоянная скорость выполнения запросов. Производительность не должна падать при росте числа измерений.

Клиент/сервер архитектура. Системы должны базироваться на открытых интерфейсах и иметь модульную структуру.

Различное число измерений. Системы не должны ограничиваться трехмерной моделью представления данных. Измерения должны быть эквивалентны по применению любых функций.

Динамическое представление разреженных матриц. Под разреженной матрицей понимается матрица, не каждая ячейка которой содержит данные. OLAP-системы должны содержать средства хранении и обработки разреженных матриц больших объёмов.

Многопользовательская поддержка. OLAP-системы должны поддерживать многопользовательский режим работы.

Неограниченные многомерные операции. Аналогично требованию о различном числе измерений: все измерения считаются равными, и многомерные операции не должны накладывать ограничения на отношения между ячейками.

Интуитивно понятные инструменты манипулирования данными. Для формулирования многомерными запросами пользователи не должны работать в усложнённых меню.

Гибкая настройка конечных отчётов. Пользователи должны иметь возможность видеть только необходимые им данные, причём все их изменения должны немедленно отображаться в отчётах.

Отсутствие ограничений. Отсутствие ограничений на количество измерений и уровней агрегации данных.

В дальнейшем Найджел Пендс переформулировал 12 правил Кодда в более ёмкий тест FASMI (Fast Shared Multidimensional Information). По его определению, OLAP-система должна быть:

- ✧ Fast – быстрой, обеспечивать почти мгновенный отклик на большинство запросов;
- ✧ Shared – многопользовательской, должен существовать механизм контроля доступа к данным и возможность одновременной работы многих пользователей;
- ✧ Multidimensional – многомерной. Данные должны представляться в виде многомерных кубов;

✧ Information – данные должны быть полны с точки зрения аналитика, т.е. содержать всю необходимую информацию.

Большинство из существующих OLAP-средств удовлетворяют всем этим признакам. Однако в реализации подобных приложений возникает ряд проблем, прежде всего связанных с увеличением объёма данных, которые необходимо хранить.

В 1995 г. группа исследователей во главе с Джеймсом Греем [GBLP95], проанализировав создаваемые пользовательские приложения баз данных, предложили расширение языка SQL – оператор CUBE. Данный оператор отвечает за создание OLAP-кубов в SQL. Концепция многомерного представления данных намного облегчила жизнь разработчикам и пользователям. В той же работе исследователи указали ряд эвристических рекомендаций по реализации новой структуры данных. CUBE представляет собой обобщение GROUP BY операторов по всем возможным комбинациям измерений с разными уровнями агрегации данных. Оператор, расширяющий SQL, называется CUBE BY (синтаксис такой же, как и у GROUP BY). В стандарт SQL'99 включён набор операторов для работы с OLAP-данными.

Расширения стандартов SQL-1999 и SQL-2003, восполняющие OLAP-функциональность языка SQL:

✧ Grouping Set запросы;
✧ Cube By запросы;
✧ Rollup By запросы;
✧ Window By запросы;
✧ Model By запросы в Oracle 10g и выше.

Исследования в области многомерных массивов данных велись задолго до работ Грея, но его работа стала основополагающей в создании промышленных продуктов и расширении языка SQL.

БИЗНЕС-ПРИЛОЖЕНИЯ НА ОСНОВЕ OLAP-ТЕХНОЛОГИЙ, ПРИМЕРЫ ПРОДУКТОВ

Наиболее часто встречаются следующие применения OLAP-технологий:

Анализ данных. Задача, для которой изначально использовались и до сих пор остаются самыми популярными OLAP-средства. Многомерная модель данных, возможность анализировать значительные объёмы данных и быстрый отклик на запросы делают подобные системы незаменимыми для анализа продаж, маркетинговых мероприятий, дистрибуции и других задач с большим объёмом исходных данных.

Примеры продуктов: Microsoft Excel Pivot Tables, Microsoft Analysis Services, SAP BW, Oracle Essbase, Oracle OLAP, Cognos PowerPlay, MicroStrategy, Business Objects.

Финансовое планирование\бюджетирование. Многомерная модель позволяет одновременно вводить данные и легко анализировать их (например, план-факт анализ). Поэтому ряд современных продуктов класса CPM (Corporate Performance Management) используют OLAP-модели. Важная задача – многомерный обратный расчёт (backsolve, breakback, writeback), позволяющий рассчитать требуемые изменения детальных ячеек при изменении агрегированного значения. Это инструмент для анализа «что-если» (what-if), т.е. для проигрывания различных вариантов событий при планировании.

Примеры продуктов: Microsoft PerformancePint, Oracle EPB, Oracle OFA, Oracle Hyperion Planning, SAP SEM, Cognos Enterprise Planning, Geac.

Финансовая консолидация. Консолидация данных согласно международным стандартам учёта, принимая во внимание доли владения, различные валюты и внутренние обороты – актуальная задача в связи с ужесточающимися требованиями проверяющих органов (SOX, Basel II) и выходом компаний на IPO. OLAP-технологии позволяют ускорить расчёт консолидированных отчётов и повысить прозрачность всего процесса.

Примеры продуктов: Oracle FCH, Oracle Hyperion FM, Cognos Controller.

МНОГОМЕРНЫЕ КУБЫ, ОПРЕДЕЛЕНИЕ И СВОЙСТВА

Рассмотрим базовую (фактическую) таблицу, на основе которой мы будем строить OLAP-куб. Множество атрибутов условно делят на 2 группы:

1. Набор измерений (категорий, локаторов), служащие критериями для анализа и определяющие многомерное пространство OlAP-куба. За счёт фиксации значений измерений получаются срезы (гиперплоскости) куба. Каждый срез представляет собой некий запрос к данным, включающий агрегации.

2. Набор мер – функции, ставящие данные в соответствие каждой точке пространства.

Из атрибутов создаются измерения, содержащие проекцию по атрибуту, с введённой иерархией (например, для таблицы, содержащей фактические данные по продажам магазина, возможно измерение «Время», содержащее иерархию вида «Год-Месяц-Неделя-День»). Куб представляет собой декартово произведение измерений, где для каждого элемента произведения проставлен набор мер.

В кубе существуют отношения обобщения и специализации (roll-up/drill-down) по иерархиям измерений. Ячейка высокого уровня иерархии может «спускаться» (drill-down) к ячейке низкого уровня. Например, (R1, ALL, весна) может «спуститься» к ячейке (R1, книги, весна) и наоборот, «подниматься» (roll-up) от (R1, книги, весна) к (R1, ALL, весна) по измерению «продукты».

Измерения. Измерения куба – набор доменов, по которым создаётся многомерное пространство. Важная особенность OLAP-моделей – разделение измерений на локаторы (задающие точки) и меры (задающие значение). Как отмечено в [Tho02], данное разделение может носить как условный, так и жесткий характер. В случае условного разделения возможно «разворачивать» измерения как данные и как аналитики, создавая новое значение куба по продажам – «количество продаж». Таким образом, чрезвычайно возрастает гибкость моделей и уровень абстракции. Однако данный подход, несмотря на свою привлекательность, чрезвычайно сложен в реализации (в частности, необходимость создания оптимальных алгоритмов хранения абстрактных типов данных) и, насколько известно, нигде промышленно не реализован. Теоретически, вкупе с моделированием структур логикой предикатов первого порядка, абстрагирование понятия «измерение» даёт очень интересные результаты.

Локаторы куба отличаются иерархической структурой, и для получения значений мер на каждом уровне агрегирования вводятся агрегирующие функции.

Иерархии и агрегирование. Иерархичность данных – одно из важнейших свойств многомерных кубов. Иерархии призваны добавлять новые уровни в аналитическое пространство пользователя. Самый распространённый пример иерархии – «месяц–неделя–год». Соответственно, для уровней иерархии работают отношения обобщения и специализации (rollup/drilldown). Как правило, в работах рассматриваются простые примеры иерархий «детальное значение – ALL», однако подобного уровня детализации может быть недостаточно.

Все иерархии можно разбить на 2 типа. Основой разбиения будет служить расстояние от корня до листов. Если расстояние одинаково до всех листов – иерархии *уровневые* (leveled), в другом случае – *несбалансированные* (ragged).

Примеры типов иерархий:

- ✧ **уровневые:** день–месяц–год; улица – город – страна;
- ✧ **несбалансированные:** Организационная диаграмма, различная группировка продуктов.

Важное свойство уровневых иерархий – возможное наличие частичного порядка внутри каждого уровня иерархии. Например, возможность сравнения месяцев по старшинству или городов по географическому положению. Большинство современных средств (алгоритмов) пренебрегают данным свойством, удаляя тем самым потенциально полезные связи модели.

Агрегирующие функции, меры и формулы. Неотъемлемая часть OLAP-модели – задание функций агрегирования. Поскольку цель OLAP – создание многоуровневой модели анализа, данные на уровнях, отличных от фактического, должны быть соответствующим образом агрегированы. По каждому измерению возможно задавать собственную (и не одну) функцию агрегации.

В [GC97] приведена следующая классификация агрегирующих функций с точки зрения сложности распараллеливания:

- ✧ дистрибутивные функции позволяют разбивать входные данные и вычислять отдельные итоги, которые потом возможно объединять;
- ✧ алгебраические функции возможно представить комбинацией из дистрибутивных функций (например, Average() можно представить как сумму, разделённую на count);
- ✧ холистические функции невозможно вычислять на частичных данных или представлять каким-либо образом.

ХРАНЕНИЕ И ЭФФЕКТИВНЫЙ РАСЧЁТ OLAP-КУБОВ

Материализация представлений. Главным свойством OLAP-систем считается возможность эффективно отвечать на запросы. Одна из мер повышения этой эффективности – материализация кубов, а не вычисление их «на лету» (вычисление агрегаций непосредственно во время обработки запроса).

Считается стандартным представление куба в виде т.н. решетки – графа, в котором узлы определяют представления (view) для ответа на запрос. Для каждого узла пометка обозначает измерения, по которым есть фактические данные в представлении; по пропущенным измерениям производится агрегация. Для трехмерного куба (регионы, продукты, время года) структура будет выглядеть следующим образом (*рис. 1*):

Таким образом, в получившейся структуре куба материализуется набор представлений, содержащий агрегированные данные. Подобные представления называются *подкубами* (англ cuboids). Ячейки базового подкуба называются *базовыми ячейками,*

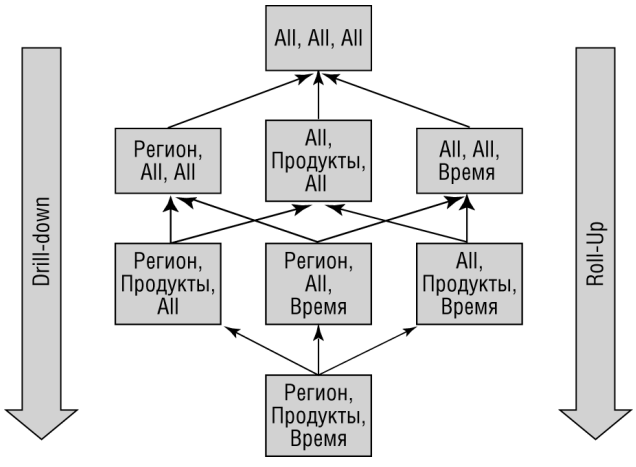

Рис. 1. Структура куба в виде набора представлений для агрегации

ячейки других подкубов называются *агрегированными ячейками*.

Выбор материализованных элементов этого набора определяет будущую производительность системы. Мы можем получить набор представлений, при использовании которого для выполнения запросов не будет производиться более 1–2 агрегаций (по одному измерению), что означает очень быстрый ответ на запрос. И наоборот, возможна ситуация, в которой для ответа на запрос необходимо будет создавать все агрегации от фактических данных (базового куба). Однако количество подкубов экспоненциально зависит от количества измерений, поэтому полная материализация может требовать огромного объёма памяти и места на жестком диске.

Изучение алгоритмов полной материализации помогает в расчётах индивидуальных подкубов. В дальнейшем их можно хранить на второстепенных носителях, или же создавать полные кубы на основе подмножества измерений и осуществлять детализацию (drill-down) в исходные данные. Подобные алгоритмы должны быть масштабируемыми, принимать во внимание ограниченное количество оперативной памяти, время вычисления и общий размер рассчитанных данных.

Многие ячейки кубов могут не представлять интереса для аналитиков, так как данные в них пренебрежимо малы. Подобная ситуация возникает часто, так как данные разреженно распределены в многомерном пространстве куба. Например, клиент покупает каждый раз лишь несколько товаров в одном магазине. Подобное событие будет отражено в виде набора ячеек с малыми показателями мер (объём покупки, количество предметов). В таких случаях полезно вычислять лишь ячейки со значением меры, большим определённой границы. Например, нас будут интересовать лишь покупки на сумму свыше 500 рублей. Такой подход позволяет точнее сфокусировать анализ, сократить время вычисления и отклика. Подобные частично вычисленные кубы называются кубами-айсбергами (англ. iceberg cubes), так большая часть их данных «скрыта под водой». Простой подход к вычислению кубов-айсбергов заключается в следующем: сначала вычислить весь куб, затем применить условие и отрезать неудовлетворяющие ячейки. Но это слишком расточительно. Нужно вычислять куб-айсберг, не вычисляя полный куб. Использование ограничений на значение меры может приводить к ситуациям, требующим бессмысленных повторных вычислений. Например, в стомерном кубе существуют всего 2 ячейки, отличающиеся по значениям одного измерения. Если меры в этих ячейках больше заданной границы, появится множество дублирующих эти суммы ячеек (по всем 99 измерениям), а на деле в кубе разнятся всего три из них.

Множество работ посвящено выбору комплекса представлений (подкубов) для материализации. Из них необходимо выделить [HRU96], где впервые приведена модель оценки стоимости материализации представлений и создания агрегаций. На базе подобных оценок создан алгоритм материализации представлений, оптимизирующий запросы по стоимости. Но применимость алгоритма [HRU96] и многих последующих сильно ограничена типами запросов, распределением данных, структурой куба и пр. Ограничения накладываются самими авторами, поэтому часто создаются эффективные алгоритмы. Доказано, что общая проблема выбора представлений для материализации NP-полна [KM99], поэтому интересной темой представляется создание алгоритмов полной материализации (materialize-all) куба, поскольку их применение не имеет подобных ограничений и не зависит от выбора начальных параметров. Необходимо заметить, что большинство промышленных OLAP-систем строятся на фактических данных, хранимых в реляционных таблицах, поэтому обоснованный выбор и последующая настройка набора материализованных представлений даёт результаты выше, чем существующие промышленные алгоритмы полной материализации.

Главная проблема подхода полной материализации (materialize-all) – это «взрыв данных», при котором объём данных и время вычисления куба растут экспоненциально. Например, десятимерный куб без иерархии внутри измерений, с размерностью 100 для каждого измерения, приводит к структуре с ячейками. Даже если мы положим разреженность 1 (только одна из миллиона ячеек содержит данные), куб всё

равно будет иметь 1,1^14 непустых ячеек. Таким образом, сокращение размера куба – насущная задача создателей OLAP-приложений.

Еще одно важное ограничение – требование сохранения семантики отношений обобщения/специализации (roll-up/drill-down). Отбрасывая это требование, многие алгоритмы достигают хороших результатов, но восстановление этих отношений в дальнейшем либо невозможно, либо трудно вычисляется, что ограничивает возможность применения подобных алгоритмов.

СПОСОБЫ ХРАНЕНИЯ

ROLAP (Relational OLAP) – данные, включающие в себя все возможные агрегации, хранятся в реляционных таблицах. Все запросы пользователя транслируются в SQL-операторы выборки из данной таблицы.

Плюсы данного подхода: все данные хранятся внутри одной СУБД в одном формате.

Минусы данного подхода: чрезмерное увеличение объёма таблицы данных для куба и сложности пересчёта агрегированных значений при изменениях начальных данных.

MOLAP (Multidimensional OLAP) – все данные хранятся в многомерной базе данных.

Все запросы пользователя транслируются в запросы многомерной выборки (MDX, Express 4GL и др).

Плюсы данного подхода: все данные хранятся в многомерных структурах, что существенно повышает скорость обработки запросов.

Минусы данного подхода: данные куба «оторваны» от базовой таблицы, необходимы специальные инструменты для формирования кубов и их пересчёта в случае изменения базовых значений.

HOLAP (Hybrid OLAP) – базовые данные хранятся в реляционной таблице, агрегированные – в многомерной структуре.

Данный метод пытается совмещать достоинства предыдущих, не имея их недостатков, но по скорости он проигрывает MOLAP. Кроме того, с его помощью невозможно целостное хранение данных и возрастают затраты на поддержку и определение типа хранения для подкубов.

ВЫВОДЫ

OLAP-технологии – актуальная и востребованная тема исследований, её практические результаты имеют широкое применение.

Несмотря на достаточно долгую историю исследований, до сих не существует единых терминологических стандартов, стандартов передачи данных, языка запросов и формирования кубов. Растущие объёмы корпоративных данных повышают значимость средств анализа, большая часть которых построена на OLAP-принципах, в связи с чем актуальны проблемы выбора оптимальных схем хранения и обработки OLAP-кубов. Интеграция различных источников информации для анализа порождает новые применения аналитических технологий, например, для анализа данных геопозиционирования (GPS) в привязке к финансовой информации используются spatioOLAP-технологии. Задачи бюджетирования, требующие совмещения скорости ввода транзакционных систем и аналитических возможностей OLAP, представляют собой особый класс систем, алгоритмическая база которых только создается. ■

Литература

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, Proc. 20th Int. Conf. Very Large Data Bases, VLDB, pages 487–499. Morgan Kaufmann, 12–15 1994.

[BR99] Kevin Beyer and Raghu Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In SIGMOD, 1999.

[Cod93] E.F. Codd. Providing OLAP for end-user analysys: An IT mandate. 1993.

[GBLP95] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Microsoft Lab, 1995.

[GC97] Sanjay Goil and Alok Choudhary. High performance olap and data mining on parallel computers. Center of Parallel and Distributed Computing Technical Report TR-97-05, 1997.

[HRU96] Venky Harinarayan, Anand Rajaraman, and Jeffrey Ulman. Implementing data cubes efficiently. SIGMOD, 1996.

[KM99] H.J. Karloff and M. Mihail. On the complexity of view-selection problem. In PODS, 1999.

[SG99] Gerd Stumme and Bernhard Ganter. Formal Concept Analysis - Mathematical Foundations. Springer, 1999.

[SR04] Yannis Sismanis and Nick Roussopoulos. The polynomial complexity of fully materialized coalesced cubes. In VLDB, 2004.

[SS01] Sunita Sarawagi and Gayatri Sathe. Intelligent rollups in multidimensional olap data. In VLDB, 2001.

[Tho02] Erik Thomsen. OLAP Solutions: Building Multidimensional Information Systems Second Edition. Wiley Computer Publishing John Wiley & Sons, Inc., 2002.