Организация баз данных
2002. №2(4)

УДК 681.326

© 2002 г. **Н.А. Гребенников,**
В.М. Постников, канд. техн. наук

(Московский государственный технический университет им. Н.Э. Баумана)

РАЗРАБОТКА МЕТОДА И МОДЕЛИ ОЦЕНКИ ВРЕМЕНИ ВЫПОЛНЕНИЯ ЗАПРОСОВ ПОЛЬЗОВАТЕЛЕЙ СЕРВЕРАМИ СОВРЕМЕННЫХ СУБД

Рассмотрен процесс оптимизации выполнения запросов пользователей серверами СУБД. Проведен детальный анализ процесса обработки запросов, выявлены основные подходы к оптимизации запросов, реализуемые в современных СУБД. Предложены модель процесса оптимизации запросов и метод оценки времени выполнения запросов пользователей серверами СУБД.

Введение

Системы управления базами данных (СУБД) являются основным компонентом современных систем обработки информации. Все современные модели систем обработки информации (СОИ) – такие как модель файлового сервера, модель сервера баз данных, модель сервера приложений, модель распределенных вычислений и другие – имеют в своей основе выполнение запросов к базам данных, которые обрабатываются СУБД. При разработке и модернизации таких систем необходимо оценивать различные характеристики производительности, основной из которых является время выполнения запросов пользователей сервером СУБД.

Анализ существующих методов оценки времени выполнения запросов пользователей серверами СУБД

В настоящее время предложен ряд методов для оценки времени выполнения запросов пользователей серверами СУБД, однако все они имеют существенные недостатки, не позволяющие эффективно их использовать при проектировании и перепроектировании СОИ. Основным недостатком

существующих методов является то, что большинство позволяет оценивать только частные параметры процесса обработки (чаще всего это стоимость выполнения ряда низкоуровневых операций, реализуемых модулем выполнения запросов), но не предоставляет механизма для оценки всего процесса обработки запросов. Причиной этого служит сложность определения плана выполнения запроса по исходному запросу, что, в свою очередь, определяется декларативным, высокоуровневым характером языков запросов.

Наглядным примером указанной проблемы служит работа [1], в которой проводится детальный анализ методов оценки времени выполнения запросов к СУБД. Аналитическая модель Тиюва [2] не является, как утверждается в работе [1], "аналитической моделью оценки времени выполнения запросов к ООСУБД", а представляет всего лишь четыре модели хранения объектов на диске. Имитационная модель Дэлиса [3] вообще не позволяет учитывать параметры запросов и вычислять время обработки запросов в СУБД. Алгоритмы выполнения запросов Грэфа [4] и алгоритмы соединений Хэрриса [5], как показывает обзор, могут рассматриваться только как хорошие стоимостные функции для части физических операций, реализуемых модулем выполнения СУБД, но не как "методы оценки времени выполнения запросов". Предложенный в работе [1] метод оценки времени выполнения запросов к ООСУБД, к сожалению, не позволяет определить время выполнения конкретного запроса на конкретной СУБД, а предлагает формулы для оценки стоимости важной, но не единственной операции объединения, используемой для выполнения запросов с выражениями пути в ООСУБД.

По нашему мнению, для оценки времени выполнения запросов пользователей могут быть применены два подхода: калибровочные методы [6, 7] и эвристические методы [4]. Недостаток первого подхода – в том, что модели требуют калибровки (настройки) для каждой программно-аппаратной платформы, что приближает их по трудоемкости к эксперименту. Недостатком второго подхода является некорректность и отсутствие достаточного обоснования предположений, которые делаются о процессе выполнения запросов. Эти недостатки порождают необходимость разработки нового метода, в которых их не будет.

Концепция метода оценки времени выполнения запросов серверами современных СУБД

Критерии качества нового метода. Анализ существующих методов оценки времени выполнения запросов пользователей серверами СУБД показал, что существует необходимость разработки нового метода оценки данного параметра. Предлагается использовать следующие критерии качества для нового метода:

– полнота охвата множества запросов, которые могут быть сформулированы пользователем на языке SQL;

– точность и обоснованность получаемых результатов;

– гибкость – возможность настройки на различное аппаратно-программное окружение.

Формулирование метода оценки времени выполнения запросов пользователей, удовлетворяющего указанным критериям качества, невозможно без анализа процесса обработки запросов пользователей серверами СУБД.

Анализ процесса обработки запросов. Обработка запросов пользователей является фундаментальной функцией СУБД. Это процесс извлечения данных из базы данных, соответствующих требованиям, определенным пользователем. Процесс обработки запросов пользователей серверами современных СУБД представлен на рис. 1.

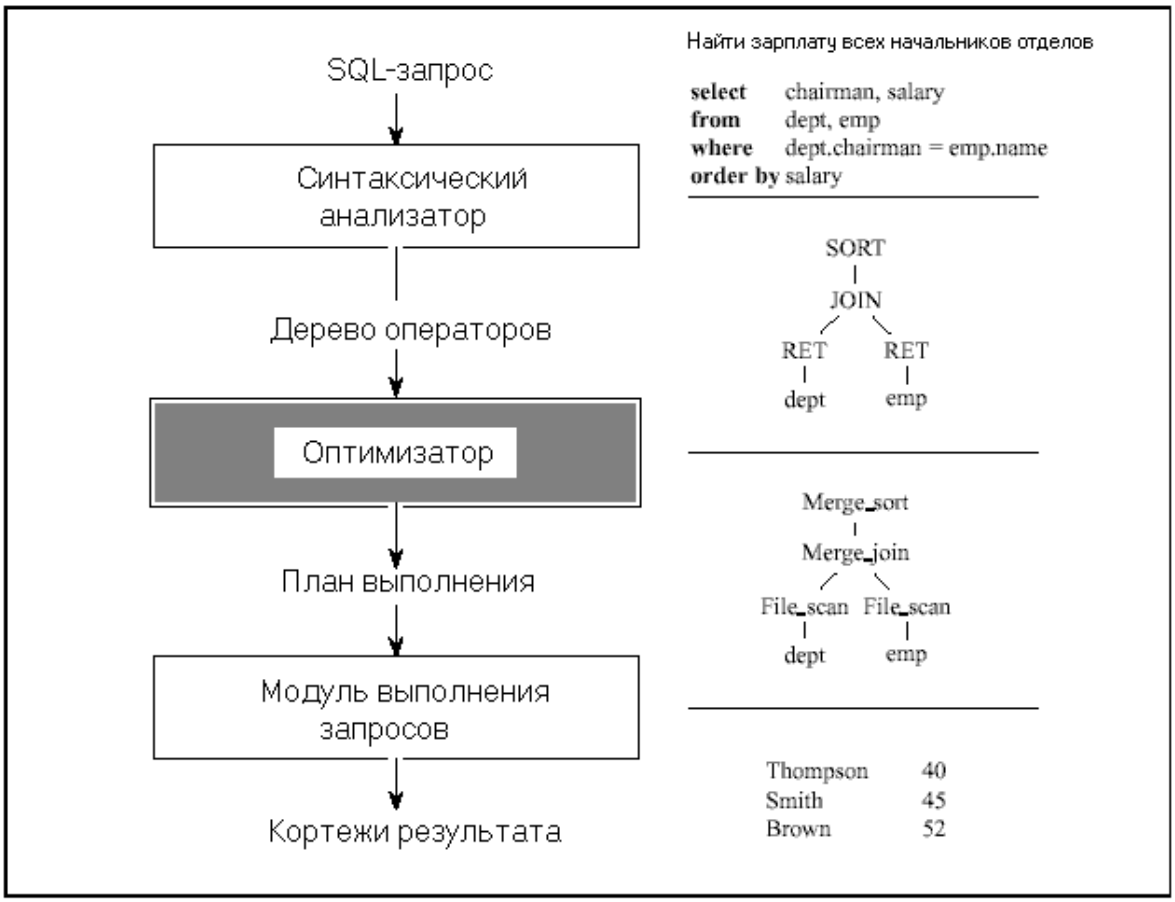

Рис. 1. Процесс обработки запроса пользователя сервером СУБД.

Процесс начинается с разбора поступившего запроса (выраженного на высокоуровневом языке формирования запросов, – например, SQL) и перевода его в дерево запроса, состоящего из операций реляционной алгебры. Затем оптимизатор запросов трансформирует это дерево запроса в план выполнения запроса. И, наконец, модуль выполнения запроса исполняет план выполнения, после чего результаты запроса возвращаются пользователю. Более детально процесс обработки запросов описан в работе [8].

Из рассмотренного процесса обработки запроса пользователя сервером СУБД видно, что важнейшую роль играет процесс оптимизации запро-

сов. Это процесс генерации эффективного плана (стратегии) выполнения заданного запроса пользователя. Оптимизатор в процессе своей работы решает задачу *оценки времени выполнения запроса*. Оптимизатор оценивает все возможные планы выполнения запроса, – в частности, тот план, по которому в результате и будет происходить выполнение. Этот факт позволяет предложить новый метод оценки времени выполнения запросов пользователей серверами СУБД, основанный на применении настраиваемого расширяемого оптимизатора СУБД. Потенциально предлагаемый метод позволит получать достоверные оценки времени выполнения запросов для принятия рациональных решений на этапе реорганизации СОИ. Однако, несмотря на кажущуюся простоту метода, для получения адекватных результатов необходимо решить задачу построения оптимизатора, который будет удовлетворять указанным критериям качества. Для этого предлагается построить детальную модель процесса оптимизации запросов пользователей серверами современных СУБД, что, в свою очередь, требует анализа подходов к оптимизации запросов, реализуемых в современных СУБД и их предшественниках.

Анализ существующих подходов к оптимизации запросов, реализуемых в современных СУБД. Существуют различные подходы к проблеме оптимизации запросов пользователей в СУБД [9]. Это одна из наиболее сложных проблем компьютерной обработки информации, которая является NP-полной [10].

Первые оптимизаторы реляционных запросов были описаны в работах [11] и [12] для систем Sytem R и Ingres соответственно. Эти оптимизаторы были разработаны для частных вариантов реляционной модели, особенно для их физических реализаций. Позднее в реляционную модель было добавлено много различных расширений. Так как первые оптимизаторы запросов были разработаны для одной модели данных, возникла необходимость в разработке нового расширяемого оптимизатора. Генератор оптимизаторов EXODUS [13] был специально разработан, чтобы вписаться в структуру расширяемой системы баз данных EXODUS. Позднее Грефом и другими были разработаны среды расширяемых оптимизаторов Volcano [14] и затем Cascades [15].

На рис. 2 показаны подходы к оптимизации запросов, реализуемые в современных СУБД. Все существующие подходы можно классифицировать по следующим признакам:

– тип оптимизации (эвристическая или стоимостная) оптимизация,

– алгоритм стратегии поиска (детерминированные в нисходящем или восходящем стиле, генетические, гибридные и эвристические) алгоритмы,

– способ генерации пространства поиска (жесткий – по заранее определенным правилам; гибкий – с использованием набора изменяемых правил).

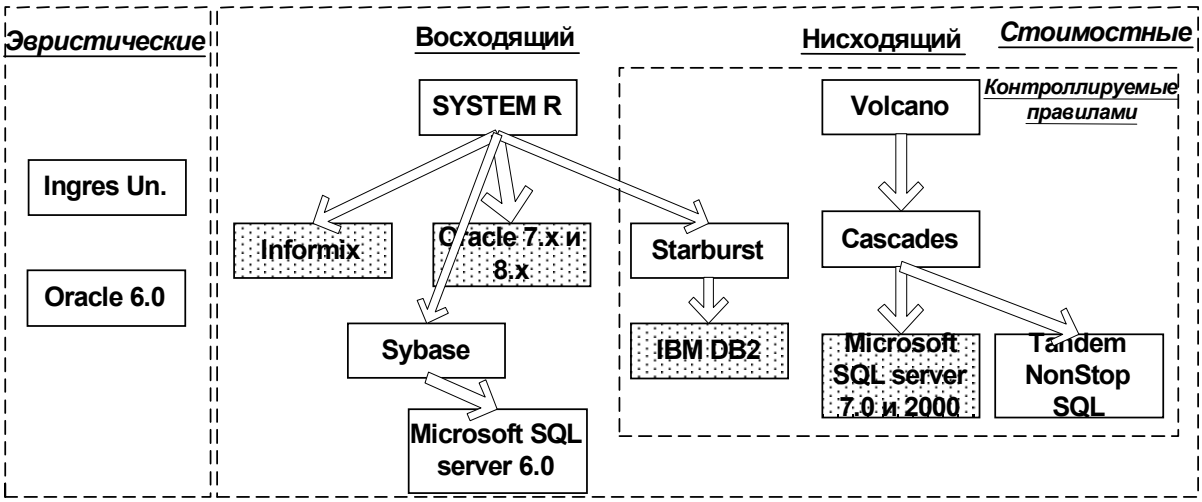

Рис. 2. Подходы к оптимизации запросов, реализуемые в современных СУБД.

Рассмотрим указанные признаки классификации более подробно.

Тип оптимизации. Основным отличием стоимостного подхода по сравнению с эвристическим является то, что оптимизатор генерирует множество планов выполнения, основываясь на возможных порядках объединения, операциях объединения и доступных методах доступа. Затем оптимизатор оценивает стоимость каждого плана и выбирает план с наименьшей стоимостью. Оптимизатор оценивает стоимости в соответствии с заданной стоимостной моделью, которая определяет стоимость выполнения плана, основываясь на оценках ресурсов, необходимых для выполнения низкоуровневых операций «физической» алгебры, которые умеет выполнять модуль выполнения запросов.

Алгоритм стратегии поиска. Стратегия поиска – это алгоритм, определяющий, какие планы в пространстве поиска следует рассматривать и оценивать. Подавляющее большинство промышленных СУБД использует детерминированные алгоритмы поиска в пространстве планов и более конкретно – алгоритм динамического программирования [16] или его нисходящую версию – алгоритм «запоминания» [17].

Способ генерации пространства поиска. Одним из главных достижений в области разработки СУБД явился переход от жесткой структуры построения оптимизатора к оптимизаторам, построенным по гибкой, базирующейся на правилах, структуре. Примерами таких систем являются EXODUS [13], Starburst [18], Volcano [14], Cascades [15], MS SQL Server 2000. Идея расширяемости состоит в модульном построении оптимизатора, что облегчает добавление логических и физических операторов. Это позволяет специализировать оптимизатор на особенную модель внедрением в него особого набора правил. Правила рассматриваются как способ определить альтернативные планы «высокоуровневым, декларативным способом». Правила при данном подходе выглядят как грамматика. Оптимизатор действует подобно компилятору языка программирования, для которого грамматика

описывает, как трансформировать входной формат (язык, представленный токенами) в выходной формат (например, машинный язык). Различие состоит в том, что компиляторы используют грамматики для нахождения одной последовательности терминалов, которая удовлетворяет входному потоку токенов, тогда как оптимизатор генерирует все такие последовательности.

Структура настраиваемого расширяемого оптимизатора

Анализ процесса обработки запросов и анализ подходов к оптимизации запросов пользователей, реализуемых в современных СУБД, позволяют построить модель процесса оптимизации запросов и определить структуру настраиваемого расширяемого оптимизатора. Предлагаемая структура оптимизатора запросов показана на рис. 3.

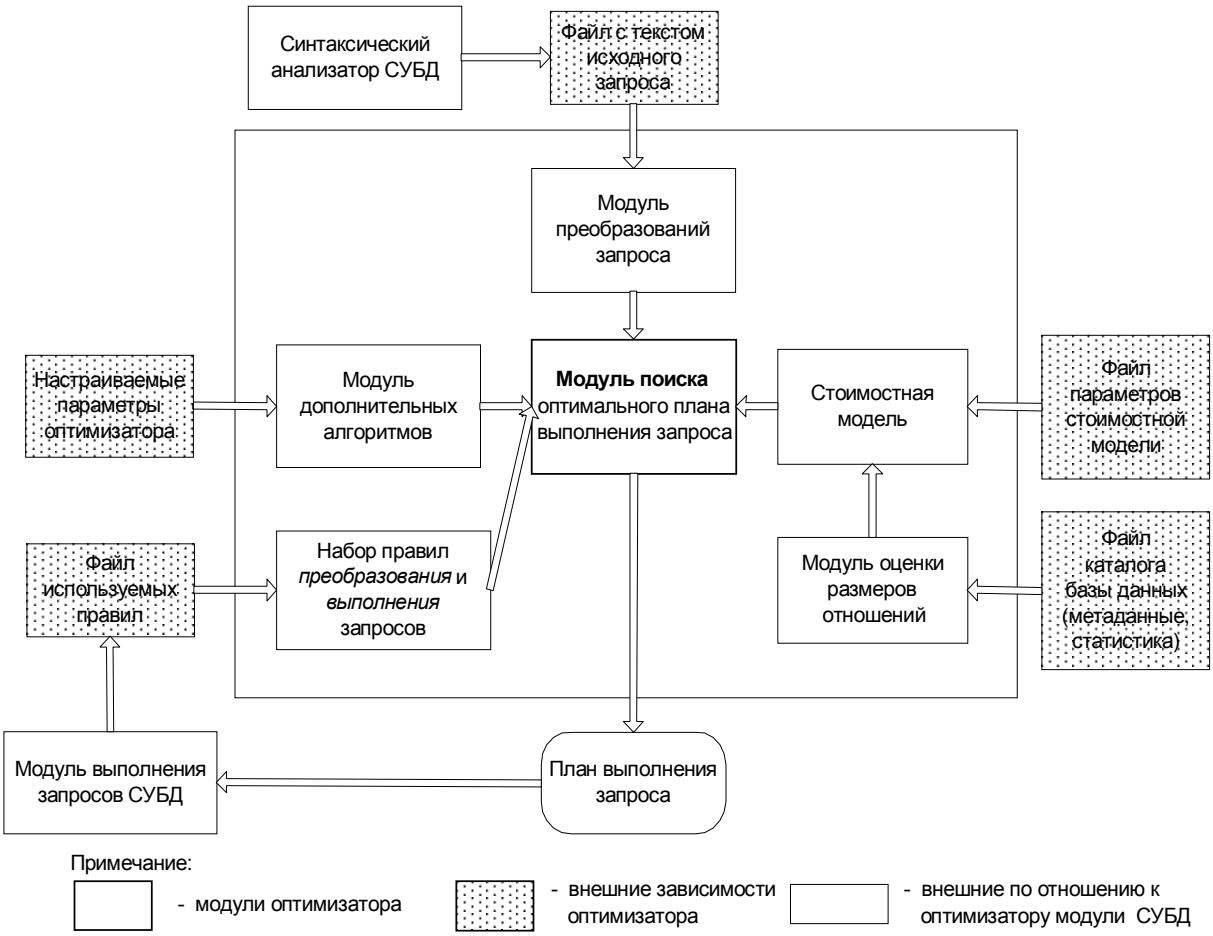

Рис. 3. Структура настраиваемого расширяемого оптимизатора запросов.

При построении оптимизатора предлагается использовать следующие подходы: стоимостная оптимизация, гибкая модульная структура, контролируемое правилами пространство поиска и детерминированные алгоритмы поиска в пространстве планов (динамическое программирование в нисходящем и восходящем стиле).

Основными компонентами оптимизатора являются следующие модули: модуль поиска оптимального плана выполнения запроса, модуль преобразований запроса, модуль дополнительных алгоритмов, модуль работы с набором правил, модуль стоимостной модели и модуль оценки размеров отношений.

Входными данными для оптимизатора являются следующие файлы: файл с текстом исходного запроса, файл каталога базы данных, файл параметров стоимостной модели, файл используемого набора правил и файл параметров оптимизатора. На выходе оптимизатора получается файл с оптимальным планом выполнения запроса, состоящий из операторов «физической» алгебры с оцененной стоимостью выполнения.

Модель процесса оптимизации запросов

Для построения оптимизатора в соответствии с описанной структурой необходимо определить модель процесса оптимизации запросов. Предлагаемая модель процесса оптимизации запросов включает в себя модель логических операций, модель физических операций, стоимостную модель, модель правил, модель ограничений пространства поиска, а также применение некоторой стратегии поиска для выбора оптимального плана в пространстве планов. Детальное описание предлагаемой модели приведено в работе [19].

Модель логических операций. Модель логических операций определяет семантику модели базы данных и операторов, используемых в логической алгебре для описания всех возможных запросов пользователя. Модель логических операций включает основные и элементарные логические операторы, а также логические свойства. Список логических операторов и логических свойств для модели логических операций показан в табл. 1.

Таблица 1

№	Логические операторы	Логические свойства
1	Чтение отношения (Get)	Схема таблицы
2	Селекция (Select)	Альтернативные ключи
3	Эквисоединение (EqJoin)	Функциональные зависимости
4	Проекция (Project)	Мощность таблицы
5	Группировка (Agg-List)	Уникальная мощность таблицы
6	Сортировка (Order-By)	Уникальность котрежей
7	Удаление дубликатов (RM-Duplicates)	Уникальная мощность атрибута
8	Взятие функции (Func-Op)	

Многие логические операторы соответствуют операциям реляционной алгебры, – например, операторы селекции (SELECT) и проекции (PROJECT). Оператор эквисоединения (EQJOIN) является оператором соединения таблиц. Логические операторы ORDER_BY и AGG_LIST являются расширениями реляционной алгебры для поддержки семантики SQL.

Элементарный логический оператор FUNC_OP предназначен для построения блоков предикатов для входа основного логического оператора SELECT.

Модель физических операций. Модель физических операций определяет набор элементарных операций, которые «умеет» выполнять модуль исполнения запросов СУБД. В общем случае она должна учитывать различные аспекты работы СУБД, такие как буферизацию ввода/вывода и модели памяти, параллельные алгоритмы, распределение данных по множеству серверов, учет материализованных представлений и т.д. Список физических операторов и физических свойств для модели физических операций показан в табл. 2.

Таблица 2

№	**Физические операторы**
1	Файловое сканирование (File Scan)
2	Индексное сканирование (Indexed Scan)
3	Фильтрация (Filter)
4	Физическое проецирование (Physical Project)
5	Соединение методом вложенных циклов (Nested Loops Join)
6	Индексное соединение методом вложенных циклов
7	Соединение методом сортировки и слияния (Merge Join),
8	Хеш-соединение (Hash Join)
9	Быстрая сортировка (Qsort)
10	Хеш-упорядочивание (Hash Order)
11	Хеш-группировка (Hash Group List)
12	Хеш-удаление дубликатов (Hash Dublicates)
	Физические свойства
1	Упорядоченность кортежей (order)
2	Стоимость выполнения (cost)

Предлагаемая физическая модель содержит физические операторы для реализации всех логических операторов, представленных в предыдущем подразделе. Для каждой из физических операций в модели определяются число входов, входные и выходные параметры, условия применения, алгоритм выполнения и метод расчета стоимости.

Модель правил. Модель правил определяет набор правил преобразований в пространстве поиска. Правила делятся на две группы: правила преобразования, которые определяют преобразования в логическом пространстве поиска, и правила выполнения, которые определяют правила замены логических операторов на реализующие их физические операторы.

Задача выбора набора правил – одна из наиболее сложных при построении оптимизатора запросов, так как именно набор правил определяет генерируемое пространство поиска и от него напрямую зависит качество работы всего оптимизатора. Предлагаемый набор правил представлен в табл. 3. Для каждого правила в модели определяются диаграмма преобразования, условия применимости, алгоритмы изменения параметров операндов.

Таблица 3

№	Правила преобразования	Правила выполнения
1	Агрегация <---> Эквисоединение (1)	Чтение отношения ---> Файловое сканирование
2	Агрегация <---> Эквисоединение (2)	Селекция ---> Фильтрация
3	Проекция <---> Эквисоединение	Селекция ---> Индексное сканирование
4	Проекция <---> Селекция	Проекция ---> Физическое проецирование
5	Проекция <---> Агрегация	Эквисоединение --> Метод вложенных циклов
6	Проекция <---> Взятие функции	Эквисоединение --> Метод сортировки и слияния
7	Коммутативность эквисоединения	Эквисоединение --> Хеш-соединение
8	Ассоциативность эквисоединения	Эквисоединение --> Индексный метод вложенных циклов
9	Идемпотентность проекции	Удаление дубликатов --> хеш-удаление дубликатов
10		Группировка -->хеш-группировка

Стоимостная модель. Стоимостная модель представляет собой набор аналитических формул для расчета стоимости физических операций. Для каждой физической операции вычисляется как стоимость операций ввода-вывода с внешними устройствами, так и стоимость выполнения процессорных инструкций (именно учет последнего фактора позволяет использовать получаемую стоимость выполнения оптимального плана в качестве оценки времени выполнения запроса). Чтобы стоимости планов можно было сравнивать, вводится коэффициент, позволяющий перевести стоимость операций ввода-вывода в стоимость выполнения инструкций процессора. Сумма стоимостей всех физических операций оптимального плана представляет собой общую стоимость выполнения запроса. Формулы для оценки стоимостей получены на основе анализа алгоритмов выполнения каждой физической операции и используют два типа параметров – мощности входных и выходных таблиц физической операции и параметры стоимостной модели, определяющие время выполнения элементарных операций. Именно через эти параметры происходит привязка стоимостной модели к параметром аппаратного обеспечения компьютера, на котором работает СУБД.

Стратегии поиска и модель ограничений пространства поиска. В качестве стратегий поиска в пространстве планов выбирается нисходящая (в стиле Cascades) или восходящая (в стиле System R) детерминированные стратегии поиска с использованием динамического программирования. Алгоритмы указанных стратегий поиска приведены в работе [19]. В качестве ограничений пространства поиска могут использоваться ограничения, представленные в табл. 4.

Таблица 4

№	Ограничения пространства поиска
1	Не использовать ограничения пространства поиска
2	Ограничение генерации только левоглубоких деревьев
3	Ограничение задержки выполнения декартова произведения
4	Стоимостное ограничение по верхней границе стоимости плана
5	Стоимостное ограничение по заданному ξ-значению.

Метод оценки времени выполнения запросов

Он состоит в выполнении следующих шагов:

1. Выбрать используемые модели:

1.1. выбрать модель логических операций на основе операций реляционной алгебры, расширений реляционной алгебры и дополнительных алгоритмов преобразований;

1.2. выбрать модель физических операций на основе низкоуровневых операций модуля выполнения запросов СУБД;

1.3. выбрать модель правил:

– правила преобразования – на основе законов реляционной алгебры и дополнительных алгоритмов,

– правила выполнения – на основе взаимосвязи операций логической и физической моделей;

1.4. выбрать стоимостную модель:

– формулы стоимости выполнения физических операций – на основе их алгоритмов выполнения,

– параметры стоимостной модели – на основе параметров аппаратной платформы сервера СУБД.

2. Выбрать стратегию поиска и определить ограничения пространства поиска:

2.1. динамическое программирование в стиле System R (восходящий подход) или Cascades (нисходящий подход);

2.2. варианты ограничений пространства поиска (см. табл. 4).

3. Описать параметры прикладной программной среды: системный каталог базы данных и запросы пользователей, выполняемые к базе данных.

4. Применить стратегию поиска для нахождения оптимального плана выполнения запроса.

5. Использовать оцененную стоимость оптимального плана в качестве оценки времени выполнения запроса.

Анализ адекватности получаемых результатов

Адекватность результатов, получаемых с помощью предлагаемого метода, проверялась с помощью тестов TPC-H международной ассоциации Transaction Processing Performance Council (TPC), которая включает более 50 фирм-разработчиков аппаратного и программного обеспечения. Тесты TPC-H разработаны для анализа характеристик производительности систем поддержки принятия решений (DSS) и состоят из 22 сложных запросов к тестовой базе данных. Выбор тестов TPC-H для анализа обусловлен рядом преимуществ, которыми они обладают перед другими тестами (например, TPC-C) в смысле оценки характеристик работы оптимизатора запросов в однопользовательской среде при большом числе строк таблиц, к которым произ-

водится доступ.

Результаты анализа адекватности метода оценки времени выполнения 6 запросов теста TPC-H для системы HP900 Superdome Enterprise Server с СУБД Oracle 9i при использовании масштабного коэффициента теста (SF), равного 1000, представлены в табл. 5.

Таблица 5

Номер запроса	**Число таблиц**	**Результаты теста, сек.**	**Результаты метода, сек.**	**Погрешность, %**
2	4	936.1	750.5	19.8
3	2	351.2	262.1	25,3
5	5	1126.9	856.8	28,4
7	5	1402.8	1163.4	17,1
9	5	4641.3	3102.1	33,2
16	2	1106.4	740.4	33,1

Данные, показанные в столбце «Результаты метода», получены при проведении экспериментов с использованием расширяемого оптимизатора запросов, структура которого была показана выше. В соответствии с предлагаемым методом оценки времени выполнения запросов пользователей сначала был определен файл «Используемый набор правил», затем файл «Параметры стоимостной модели» и файл с определением «Параметры оптимизатора». После этого в соответствии со спецификацией теста TPC-H были определены файл «Каталог базы данных» и 22 файла «Тексты исходных запросов». Далее были проведены эксперименты (применена выбранная стратегия поиска), в которых в качестве результатов были получены оптимальные планы выполнения запросов и оцененные стоимости их выполнения.

Сравнение полученных результатов и результатов тестов TPC-H дает возможность говорить о том, что погрешность предлагаемого метода для данных тестов составляет не более 35%, это позволяет использовать его для принятия рациональных решений на стадии реорганизации СОИ. Однако точность предложенного метода существенно зависит от точности задания параметров стоимостной модели, определение которых является сложной задачей.

Сравнение с существующими методами

Предлагаемый метод имеет ряд преимуществ в сравнении с существующими методами оценки времени выполнения запросов пользователей:

– в отличие от *калибровочных методов* обладает достаточной гибкостью, так как параметры аппаратно-программной среды могут быть заданы в качестве одного из наборов входных параметров,

– в отличие от *эвристических методов* обладает обоснованностью полу-

чаемых результатов, которая гарантируется тем, что оптимизатор выбирает именно тот план, по которому будет произведено выполнение запроса; кроме того, точность получаемых результатов определяется только точностью задания наборов входных данных (и не затрагивает «ядро» метода),

– в отличие от *частных методов* (позволяющих оценивать отдельные характеристики функционирования) предлагаемый метод обладает потенциальной полнотой охвата множества возможных типов запросов. Это множество определяется лишь набором реализуемых в оптимизаторе алгоритмов, который легко дополнять (свойство расширяемости).

Заключение

Анализ существующих методов оценки времени выполнения запросов пользователей серверами СУБД показал, что у существующих методов отмечается ряд недостатков, они требуют настройки для каждой программно-аппаратной платформы, не имеют достаточного обоснования и т.д.

Предложен новый метод оценки времени выполнения запросов пользователей серверами современных СУБД, основанный на построении настраиваемого расширяемого оптимизатора СУБД.

Проведен анализ существующих подходов к оптимизации запросов, реализуемых в современных СУБД. На основе проведенного анализа предложена модель процесса оптимизации запросов, включающая модели логических и физических операций, стоимостную модель, модель правил, модель ограничений пространства поиска, а также применение некоторой стратегии поиска для выбора оптимального плана в пространстве планов.

Разработана структура настраиваемого расширяемого оптимизатора, реализующая предложенную модель процесса оптимизации запросов.

Выработаны критерии качества для нового метода, проведены сравнение его с существующими методами и анализ адекватности получаемых результатов, показавшие работоспособность предложенного метода.

ЛИТЕРАТУРА

1. *Бурдаков А.В.*: Модели и методы анализа вычислительных систем с архитектурой брокера объектных запросов. Автореф. дис. … канд. техн. наук / МГТУ им. Н.Э. Баумана. М., 2002.
2. *Teeuw W.B., Rich C., Scholl M.H., Blanken H.M.* An Evaluation of Physical Disk I/Os for Complex Object Processing // 9-th International Conf. on Data Engineering, 1993. P.363-371.
3. *Delis A., Roussopoulos N.* Performance and scalability of client-server database architectures // Proceedings of 18th International Conference on VLDB. Vancouver, 1992. P.610-623.
4. *Graefe G.* Query evaluation techniques for large databases // ACM Computing Surveys. N.Y., 1993. Vol. 25. No. 2. P.73-170.

5. *Harris E.P., Ramamohanarao K*. Join algorithm costs revisited // VLDB Journal. Heidelberg, 1996. Vol. 5(1). P.64-84.
6. *Hwang H-Y., Yao-Tin Y*. An Analytical Method for Estimating and Interpreting Query Time // Proceedings of the 13th VLDB Conference. Brighton, 1987.
7. *Gardarin G., Sha F., Tang Z.-H*. Calibrating the query optimizer cost model of IRO-DB an objectoriented federated database system // Proceedings of the 22nd International Conference on VLDB. Bombay, 1996. P.378-389.
8. *Постников В.М., Гребенников Н.А*. Технология обработки запросов пользователей в СУБД ORACLE // Вестник МГТУ им. Н.Э. Баумана. Сер. "Приборостроение". 2001. № 2. С.106–126.
9. *Chaudhuri S*. An Overview of Query Optimization in Relational Systems // PODS-98. Seattle WA, USA, 1998.
10. *Ibaraki T., Kameda T*. On the optimal nesting order for computing N-relational joins. //ACM Transactions on Database Systems. 1984. 9(3). P.482-502.
11. *Selinger Patricia G., Astrahan M.M., Chamberlain R., Lorie R.A., Price T.* Access Path Selection in a Relational Database Management System. Proceedings of 1979 ACM SIGMOD Conference. June, 1979.
12. *Wong, Eugene, Youssefi, K.*, Decomposition - A Strategy for Query Processing. ACM Transactions on Database Systems, 3 (September) 1976.
13. *Graefe Goetz, Dewitt David J.* The EXODUS Optimizer Generator. In Proceedings 1987 ACM SIG-MOD International Conference on Management of Data, San Francisco, 1987. P.387-394.
14. *Graefe Goetz, McKenna William*. The Volcano Optimizer Generator: Extensibility and Efficient Search. In Proceeding of the 12th International Conf. on Data Engineering, 1993. P.209-218.
15. *Graefe G.* The Cascades Framework for Query Optimization, Bulletin of the IEEE Technical Committee on Data Engineering, 18(3), September 1995, P.19-29.
16. *Bellman R.E.* Dynamic Programming, Princeton University Press, Princeton, N. Jersey, 1975.
17. *Michie D.* 'Memo' Functions and Machine Learning, Nature, April 1968. No. 218, P.19-22/
18. *Pirahesh Hamid, Hellerstein Joseph M., Hasan Waqar*. Extensible/Rule Based Query Rewrite Optimization in Starburst. In Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, pages 39-48, San Diego, California, June 1992.
19. *Постников В.М., Гребенников Н.А*. Модель процесса обработки запросов пользователей в СУБД ORACLE // Вестник МГТУ им. Н.Э. Баумана. Сер. "Приборостроение". 2002. №3.

Статья представлена к публикации членом редколлегии Ю.А. Григорьевым.