Параллелизм в дедуктивных базах данных

Дацун Н.Н., Пушкаренко С.А.

Кафедра ПМИ ДонНТУ, S.Pushkarenko@it.stirol.net

Abstract

Datsun N. N., Pushkarenko S. A., Parallelism in deductive databases. *Datalog language translator has been developed. Executed database testing has been performed with Datalog language translator.Datalog-request analysis has been performed and variants of their multisequencing has been offered. Computation system architecture has been designed for deductive database realization. The relational database architecture modification has been suggested for the maintenance of the requests multisequencing.*

Введение

Дедуктивные базы данных представляют собой реляционную базу данных (РБД), использующую в качестве языка запросов логические языки программирования [1].

Преимуществом реляционной модели является параллелизм. “Реляционные запросы просто созданы для параллелизма, так как в действительности они являются реляционными операторами, применяемыми к очень большим наборам данных. Поскольку запросы представляются на непроцедурном языке, имеется существенная свобода при выборе способа выполнения запросов” – так оценивают сущность параллелизма запросов к РБД авторы работы [2].

В настоящее время существуют такие параллельные реализации систем баз данных: SQL Server 2005, Oracle8...9, DB2 Parallel Edition, NonStop SQL, NCR Teradata [3].

Постановка задачи: разработка программных средств для реализации, моделирования и оценки эффективности работы дедуктивных баз данных, оценка времени конструирования и выполнения запросов в дедуктивных базах данных, анализ Дейталог-запросов различной структуры и варианты их распараллеливания с целью повышения производительности дедуктивных баз данных.

1 Транслятор языка Дейталог и оценка эффективности работы дедуктивных баз данных на его основе

В 2005 году на кафедре Прикладной математики и информатики ДонНТУ был разработан транслятор языка Дейталог, используемый в настоящее время в учебных целях в курсе “Функциональное и логическое

программирование". Реализация транслятора выполнена в среде Visual Basic 6.0 с использованием технологии DAO. Транслятор представляет собой программную модель дедуктивных баз данных, обеспечивая работу с РБД в формате MS Access. Опытная эксплуатация транслятора на студенческих лабораторных работах в течение учебного семестра показала его устойчивую работоспособность.

Для оценки характеристик транслятора языка Дейталог на больших дедуктивных базах данных было выполнено тестирование на подмножестве таблиц базы данных АСУ-деканат ДонНТУ, предоставленной вычислительным центром университета. Предоставленный фрагмент базы данных содержит 12 таблиц, из которых тестированию подверглись две основные:

- STUDENT, содержащая 125 полей и 38957 записей;
- GRUPPA, содержащая 20 полей и 2176 записей.

Тестирование проводилось на компьютере с процессором Celeron 1300 MHz и размером оперативной памяти 128 Мбайт.

Особенность транслятора заключается в том, что транслятор преобразует Дейталог-запрос в эквивалентный SQL-запрос. Язык Дейталог является подмножеством реляционного логического языка, а SQL – структурированным языком запросов к реляционным базам данных. Поэтому отображение Дейталог-запроса в SQL-запрос является наиболее естественным.

Был проведен статистический анализ времени, затрачиваемого транслятором на конструирование запроса к базе данных, и на непосредственно выполнение запроса. Для определения этого времени в транслятор были внесены соответствующие дополнения: определялось системное время перед выполнением операции и после её выполнения и вычислялась разница между ними. Среднее время конструирования запроса составило 0,0000765 сек. Время выполнения запросов, соответствующих различным операциям реляционной алгебры, оказалось значительно больше (см. табл. 1).

В результате проведенных статистических исследований выявлено, что наиболее ресурсоёмкой операцией является операция выполнения SQL-запроса на выборку из базы данных (табл.1).

2 Анализ Дейталог-запросов различной структуры и варианты их распараллеливания

Основываясь на результатах проведенных исследований, предлагается производить распараллеливание запроса к базе данных. "Реляционная модель наилучшим образом подходит для "распараллеливания" запросов" [3].

Таблица 1 – Статистический анализ времени, затрачиваемого транслятором языка Дейталог на этапе выполнения запроса

№ п/п	Дейталог-запрос и его эквивалент операциям реляционной алгебры	Среднее время выполнения запроса, сек
1.	STUDENT(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10) - проекция	1,9999998
2.	STUDENT(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10, X11,X12,X13,X14,X15,X16,X17,X18,X19,X20) - проекция	4,9000000
3.	STUDENT(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10, X11,X12,X13,X14,X15,X16,X17,X18,X19,X20, X21,X22,X23,X24,X25,X26,X27,X28,X29,X30, X31,X32,X33,X34,X35,X36,X37,X38,X39,X40, X41,X42,X43,X44,X45,X46,X47,X48,X49,X50) - проекция	14,7999998
4.	STUDENT(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10, X11,X12,X13,X14,X15,X16,X17,X18,X19,X20, X21,X22,X23,X24,X25,X26,X27,X28,X29,X30, X31,X32,X33,X34,X35,X36,X37,X38,X39,X40, X41,X42,X43,X44,X45,X46,X47,X48,X49,X50), GRUPPA(X5,Y2,Y3,Y4,Y5,Y6,Y7,Y8, NAME_GRUP) – соединение по равенству	17,1999999
5.	STUDENT(X1,X2,X3,X4,X5,X6,X7,"Алексей", X9,X10) – проекция и селекция	1,6666664

Например, при выполнении Дейталог-запроса к правилу

p(X):-a(X).

таблица a условно разбивается на p частей, где p – количество процессоров. Каждый процессор при этом будет выполнять один и тот же запрос, но к своей части таблицы. Хост-процессор производит разбивку таблицы на p частей по формуле:

$$n = \frac{count}{p} \qquad (1)$$

где n – количество записей в одном диапазоне таблицы;

$count$ – количество записей в таблице;

p – количество процессоров.

Затем хост-процессор отправляет всем имеющимся в наличии процессорам задание, включающее в себя сам запрос и диапазон записей,

для которого необходимо выполнить данный запрос (начальный номер записи, конечный номер записи). Результаты запроса формируются каждым процессором в определённую область общей памяти. Хост-процессор, объединяя результаты выборки каждого процессора, формирует итоговый результат. Под хост-процессором подразумевается процессор, распределяющий задания между остальными процессорами многопроцессорной системы и формирующий после получения результатов работы каждого из процессоров окончательный результат.

При сложном запросе к правилу типа декартового произведения

p(X,Y):-a(X),b(Y).

можно разбивать на диапазоны одну из таблиц (a или b). Обе таблицы разбивать на диапазоны нельзя, т.к. результат запроса в этом случае может быть неполным.

3 Архитектура вычислительной системы для реализации дедуктивных баз данных

Установим соответствие между понятиями архитектуры вычислительной системы и понятиями дедуктивных баз данных: командой является запрос к базе данных, данными – записи базы данных.

В соответствии с этим выбираем тип вычислительной системы [4]:

- класс вычислительной системы – SIMD;
- память - разделяемая (общая);
- схема коммуникации процессоров (топология) – звезда (рис. 1).

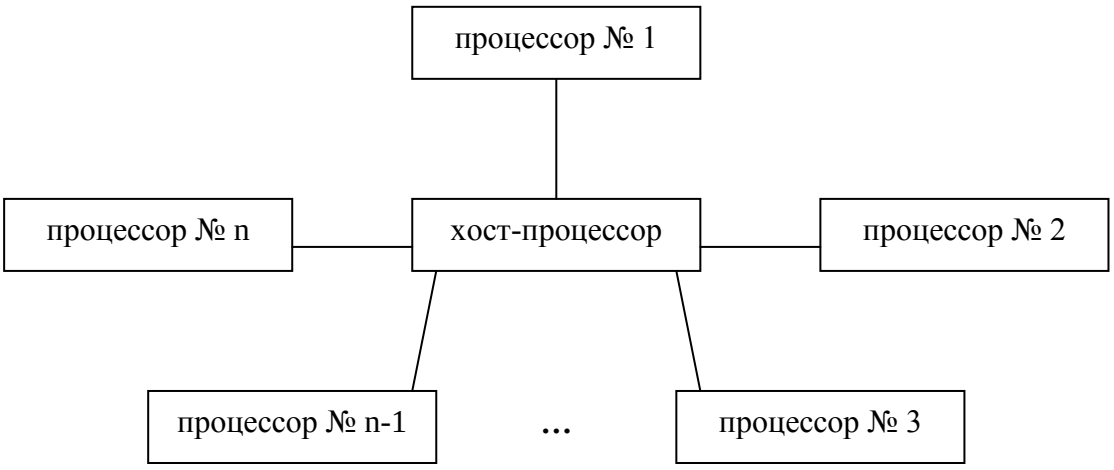

Рисунок 1. – Схема коммуникации процессоров

Расчёт начального номера записи ($begin$) и конечного номера записи (end) диапазона записей таблицы для каждого процессора:

$$begin_n = \frac{(n-1)*count}{p} + 1; \qquad (2)$$

$$end_n = \frac{n * count}{p}. \qquad (3)$$

где n – порядковый номер процессора;

p – количество процессоров.

Все операции деления в данных формулах являются целочисленными, т.к. количество записей в соответствующей таблице может не быть кратно количеству процессоров.

При составлении SQL-запроса для каждого процессора, необходимо задать диапазон поиска (начальный и конечный номера записей). При этом возникают некоторые сложности. В реляционной модели данных не существует понятия – “физический порядковый номер записи”. Физический порядок расположения записей имеет смысл только в базах данных формата dBase.

В FoxPro имеется функция recno( ), возвращающая порядковый номер текущей записи (физическое расположение). На FoxPro можно выполнить SQL-запрос следующего вида:

*select * from table where recno( )<11*

Этот запрос выбирает первые 10 записей из таблицы с именем table.

Предлагается произвести следующие изменения в стандартной архитектуре реляционных баз данных и СУБД для обеспечения распараллеливания запроса:

- модифицировать структуру файла базы данных, добавив к каждой записи таблицы скрытое "системное" поле, хранящее физический порядковый номер записи в таблице;
- при перепаковке (т.е. физическом удалении помеченных записей) базы данных, пересчитывать значение поля, хранящего физический порядковый номер записи, для избежания "пропусков" в номерах записей. При удалении записи из таблицы, она физически не удаляется, а помечается как удалённая (логическое удаление). Физическое удаление происходит при перепаковке базы данных (в MS Access: Сервис->Служебные программы->Сжать и восстановить базу данных; в MS Access 2000 есть возможность автоматического сжатия и восстановления при каждом закрытии БД);
- ввести системную переменную, возвращающую физический порядковый номер записи в таблице.

Например, если системная переменная, возвращающая физический порядковый номер записи в таблице имеет имя @@RecordNumber, то запрос, передаваемый на каждый из процессоров будет иметь вид:

SELECT * FROM a

WHERE @@RecordNumber>=@begin AND @@RecordNumber<@end;
Вместо переменных @begin и @end главный процессор подставляет конкретные значения перед отправкой запроса каждому из процессоров.

Заключение

Таким образом, в представляемой статье рассмотрен транслятор Дейталога как инструмент моделирования для реализации и оценки эффективности работы дедуктивных баз данных. Выполнена оценка времени конструирования и выполнения запросов в дедуктивных базах данных, которая показала, что время выполнения запросов значительно превышает время конструирования запроса. Проведен анализ Дейталог-запросов различной структуры, из которого видно, что наибольшее время выполнения имеют запросы типа декартового произведения (соединения). Рассмотрены варианты распараллеливания Дейталог-запросов с целью повышения производительности дедуктивных БД. Предложены изменения в стандартной архитектуре реляционных баз данных и СУБД для обеспечения распараллеливания запроса. Полученные результаты позволяют построить модель вычислительной системы предложенной архитектуры для повышения эффективности дедуктивных баз данных.

Литература

1.Чери С., Готлоб Г., Танка Л. Логическое программирование и базы данных – М.: Мир, 1992. – 352 с.

2. Девитт Д., Грэй Д. Параллельные системы баз данных: будущее высоко эффективных систем баз данных// Системы Управления Базами Данных. - 1995, №2. - с. 8-31.

3. Соколинский Л.Б. Параллельные машины баз данных// Природа. - 2001, №8.

4. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с.