УДК 004.65:004.051

© 2006 г. **Ю.А. Григорьев,** д-р техн. наук,
С.П. Остриков
(Московский государственный технический университет им. Н.Э. Баумана)

ВЫБОР ОПТИМАЛЬНОГО ПЛАНА ВЫПОЛНЕНИЯ ЗАПРОСА С ВЛОЖЕННЫМИ КОРРЕЛИРОВАННЫМИ ПОДЗАПРОСАМИ

В статье предлагается подход к построению альтернативных планов выполнения запроса с многоуровневыми вложенными коррелированными подзапросами и операциями агрегирования. Выводятся аналитические выражения, позволяющие оценивать временные характеристики этих планов, и приводятся рекомендации по выбору того или иного плана.

Введение

В настоящее время в различных информационных системах (ИС) используются системы управления базами данных, поддерживающие реляционную модель данных. Здесь для описания и манипулирования данными применяется стандартный язык SQL, который относится к классу непроцедурных языков. SQL-запрос, поступивший на сервер базы, подвергается оптимизации с целью уменьшить время его выполнения. Суть оптимизации состоит в генерации нескольких альтернативных планов реализации запроса и выборе плана с наименьшей стоимостью, которая пропорциональна времени выполнения запроса. В теории реляционных баз данных наибольшее внимание уделяется построению оптимальных планов для класса запросов, включающих операции соединения таблиц или имеющих конструкции, которые можно свести к соединению [1, 2]. В теории по существу не рассматривается важный класс запросов с многоуровневыми вложенными коррелированными подзапросами и операциями агрегирования. В публикациях [3, 4] в основном анализируются запросы с двумя коррелированными подзапросами (внешним и внутренним) и необоснованно утверждается, что для них оптимальным является план с группированием значений атрибута связи. При этом расчет стоимости и сравнение альтернативных планов не выполняются. Это привело к тому, что во многих СУБД (Oracle, DB2 и др.) для реализации запросов из указанного класса используются планы на основе операции группирования. Но анализ показывает, что эти планы являются оптимальными только в некоторой облас-

ти изменения параметров запросов и наполнения базы данных. То есть часто для реализации запроса в СУБД выбирается неоптимальный план.

Поэтому важной является задача генерации альтернативных планов выполнения запросов с многоуровневыми вложенными коррелированными подзапросами и операциями агрегирования, оценки стоимости планов и построения областей их предпочтительного использования.

Преобразование SQL-запроса с вложенными коррелированными подзапросами и операциями агрегирования

На рис. 1 приведено описание исследуемого класса запросов.

```
SELECT  A10        FROM  R1
WHERE   A11 τ1
        (SELECT    f2(A22)          FROM  R2
        WHERE      R1.A13=R2.A24    AND  A21 τ2
                   (SELECT          f3(A32)          FROM  R3
                   WHERE            R2.A23=R3.A34    AND A31
                                                     τ3
                                    (SELECT          f4(A42)      FROM  R4
                                    …
                                                                  AND An-1,1 τn-1
                                                     (SELECT      fn(An2)  FROM  Rn
                                                     WHERE        Rn-1.An-1,3=Rn.An4
                                                     ) …)))
```

Рис. 1. Описание исследуемого класса запросов.

Опишем введенные обозначения и дадим ряд пояснений. Ri ($i=\overline{1,n}$) – таблица *i*-го подзапроса, из которой читаются записи в процессе выполнения запроса. Некоторые или все таблицы могут совпадать. A10 – атрибут таблицы R1, который является результатом выполнения запроса (A10 может быть подмножеством атрибутов или операций агрегирования). Ai1 ($i=\overline{1,n-1}$) – атрибут *i*-й таблицы, для которого установлено подусловие поиска Ai1 τi fi+1(Ai+1,2). τi ($i=\overline{1,n-1}$) – арифметический оператор сравнения (=, >, >=, < и т.д.). fi ($i=\overline{2,n}$) – одна из операций агрегирования (MIN, MAX, SUM, COUNT, AVG и др.). Это может быть и арифметическое выражение, куда входят несколько операций агрегирования. Ai2 ($i=\overline{2,n}$) – атрибут *i*-й таблицы, являющийся аргументом функции агрегирования fi. Агрегирование значений Ai2 выполняется для тех записей таблицы Ri, которые удовлетворяют условию WHERE *i*-го вложенного подзапроса. Будем считать, что 1-й вложенный подзапрос – это самый внешний запрос SELECT. Ai3 ($i=\overline{1,n-1}$) – атрибут *i*-й таблицы, по которому устанавливается соединение с (*i*+1)-й таблицей (Ri+1) в (*i*+1)-м подзапросе (поэтому подзапрос является коррелированным). Ai4 ($i=\overline{2,n}$) – атрибут *i*-й

таблицы, по которому устанавливается соединение с (*i*–1)-й таблицей (Ri – 1) в *i*-м подзапросе.

В приведенных выше спецификациях (см. рис. 1) связь между таблицами соседних подзапросов устанавливается посредством равенства атрибутов Ai-1,3 и Ai4. Хотя возможны и более сложные условия. Приведем пример запроса из рассмотренного класса (рис. 2).

```
SELECT   Номер сотрудника
FROM     сотрудник R1
WHERE    зарплата >=
         (SELECT          AVG(зарплата)
         FROM             сотрудник R2
         WHERE            R1.номер отдела= R2.номер от-
                          дела  AND число заказов =
                          (SELECT          MAX(число заказов)
                          FROM             сотрудник R3
                          WHERE            R2.код управления=
                                           R3.код управления ))
```

Рис. 2. Пример запроса с вложенными коррелированными подзапросами и операциями агрегирования.

Этот запрос можно интерпретировать следующим образом: найти сотрудников организации, зарплата которых не меньше средней зарплаты сотрудников соответствующего отдела, в свою очередь, выполнявших максимальное число заказов среди сотрудников управления, куда входит данный отдел.

На рис. 3 показан результат частичной трансляции описания запроса (см. рис. 1) в дерево выражений реляционной алгебры. В дальнейшем префиксы Ri и Ri+1 перед атрибутами Ai3 и Ai+1,4 будем опускать, поскольку ясно, что эти атрибуты принадлежат соответственно таблицам Ri и Ri+1. На логическом уровне операции на рис. 3 выполняются при перемещении по дереву снизу вверх. Здесь на каждом *i*-м уровне вложенности ($i = \overline{1, n-1}$) введена операция σ без индекса, которая определяет отношение, получаемое после выполнения операции селекции с условием Ai1 τi fi+1(Ai+1,2) над отношением R1 или $\sigma_{Ai-1,3=Ai4}$. Операция $\sigma_{Ai-1,3=Ai4}$ – операция селекции с условием, указанным в качестве индекса, над отношением Ri ($i = \overline{2, n}$). Операция π_{A10} – это проекция предыдущего в дереве разбора отношения на атрибут A10, которая определяет результат выполнения исходного запроса.

Из-за коррелированности подзапросов физическая реализация плана (см. рис. 3) отличается от логической интерпретации. Действительно, при движении по дереву снизу вверх (логическая интерпретация) невозможно определить результат селекции $\sigma_{Ai-1,3=Ai4}$, так как отношение Ri–1 определяется по дереву выше. Поэтому в процессе физической реализации дерево просматривается сверху вниз. Сначала читаются записи из R1. Для каждой

такой записи определяется отношение $\sigma_{A13=A24}$ (значение A13 уже определено). Далее для каждой записи из этого отношения определяется отношение $\sigma_{A23=A34}$ (значение A23 уже определено), и так далее до самого нижнего уровня вложенности. Затем для каждой записи из рассматриваемых отношений $\sigma_{Ai-1,3=Ai4}$ выполняется перемещение снизу вверх, связанное с проверкой условий σ (без индекса) и вычислением функций агрегирования fi.

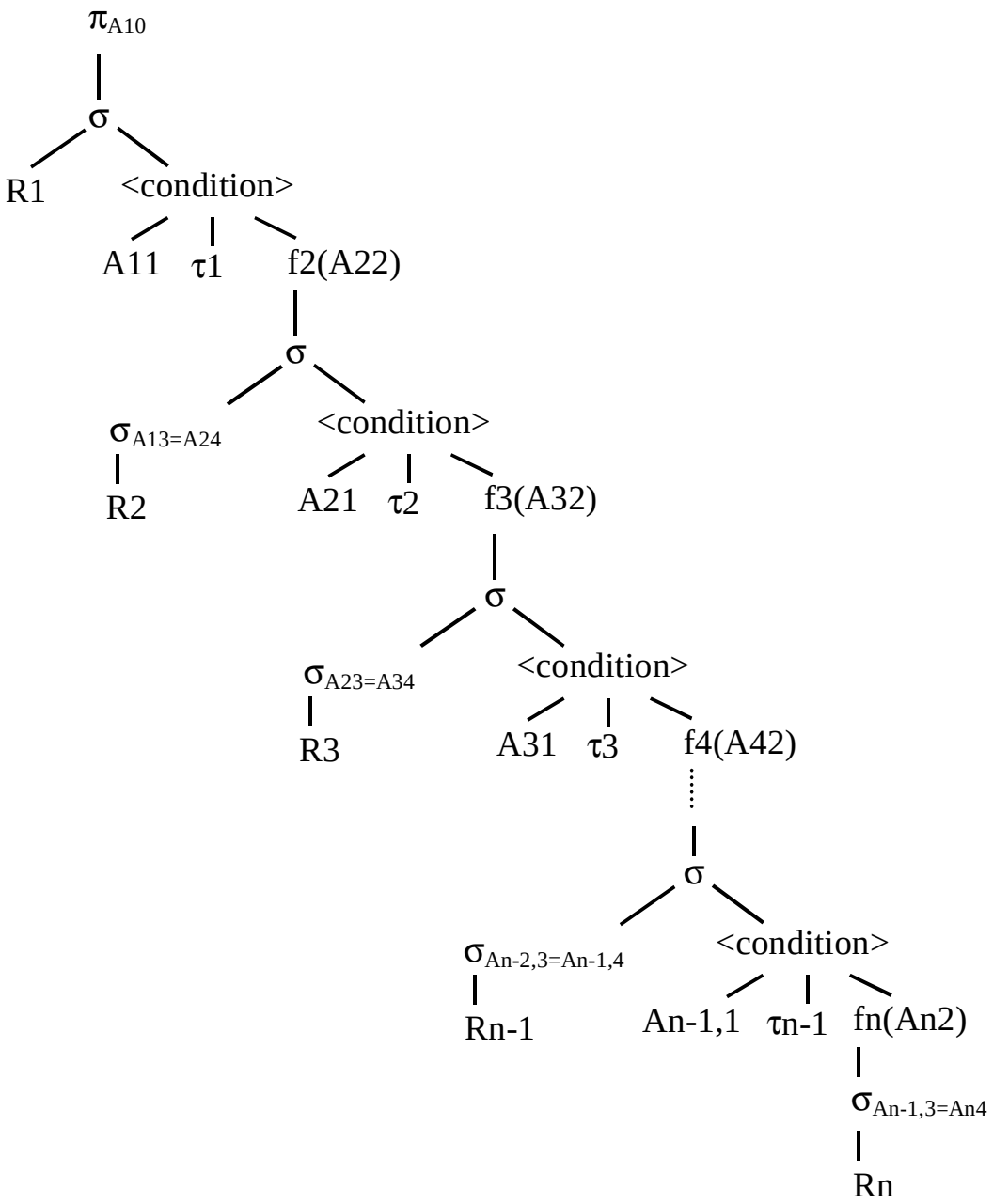

Рис. 3. Результат трансляции исходного запроса.

Далее будет показано, что при некоторых значениях параметров запроса и наполнения базы данных такой процесс выполнения запроса (см. рис. 3) может потребовать достаточно много вычислительных ресурсов. Ниже разрабатывается процедура преобразования плана, представленного на рис. 3.

Для изменения плана необходимо выполнить следующие шаги на каждом уровне вложенности:

1. Селекцию $\sigma_{Ai-1,3=Ai4}$ переместить вверх по дереву за операцию селекции σ без индекса, т.е. после ее выполнения (рис. 4).

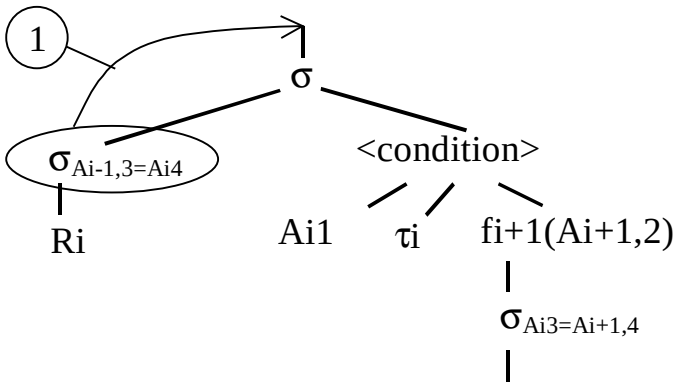

Рис. 4. Перемещение селекции $\sigma_{Ai-1,3=Ai4}$.

2. Операцию селекции $\sigma_{Ai3=Ai+1,4}$ переместить за операцию селекции σ без индекса. На место перемещенной селекции вставить операцию группирования γ(Ai+1,4 ri+1, fi+1(Ai+1,2) qi+1) (рис. 5). Здесь Ai+1,4 – это атрибут, по которому выполняется группирование. Группируемое отношение поступает на вход операции γ после выполнения операций, расположенных по дереву ниже. fi+1(Ai+1,2) – операция агрегирования атрибута Ai+1,2, которая выполняется для каждой группы. Идентификаторы ri+1 и qi+1 – псевдонимы значений, указанных слева от них, т.е. Ai+1,4 и fi+1(Ai+1,2) соответственно.

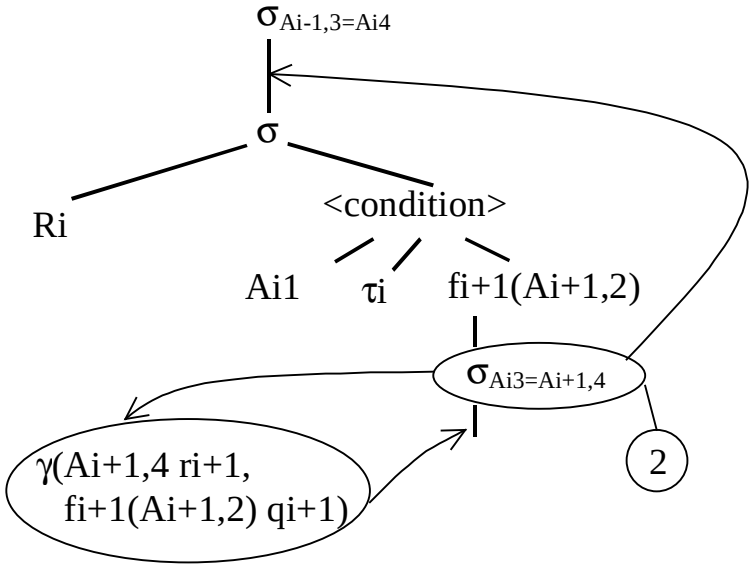

Рис. 5. Перемещение селекции $\sigma_{Ai3=Ai+1,4}$.

3. Операцию селекции σ без индекса и условие <condition> заменить на селекцию $\sigma_{Ai1\ \tau i\ qi+1}$ над декартовым произведением Ri и γ (рис. 6).

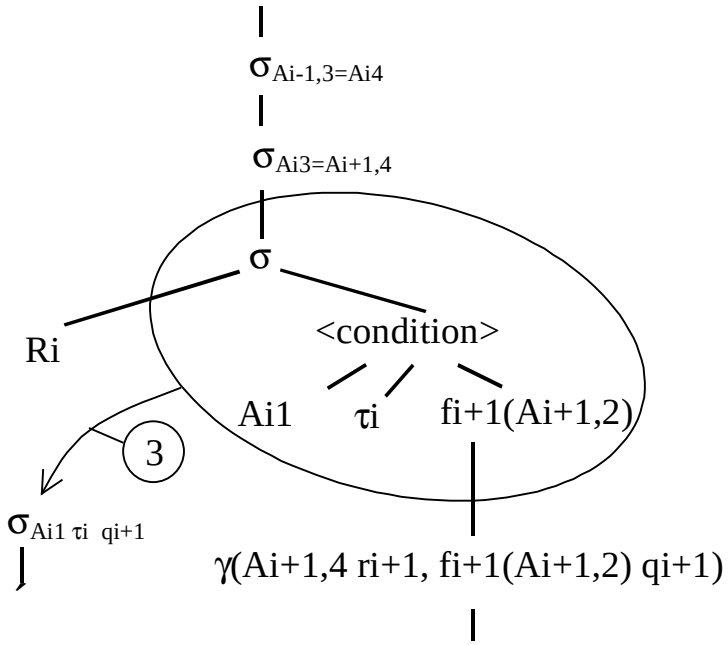

Рис. 6. Замена селекции σ без индекса.

4. Каскад селекций $\sigma_{Ai3=Ai+1,4}$ и $\sigma_{Ai1\ \tau i\ qi+1}$ заменить на одиночную селекцию $\sigma_{Ai3=Ai+1,4\ \cap\ Ai1\ \tau i\ qi+1}$ (рис. 7).

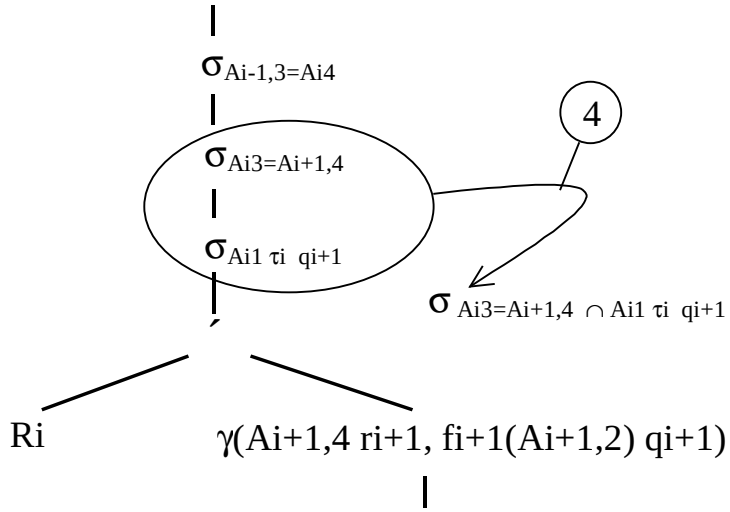

Рис. 7. Замена каскада селекций.

Приведенные выше четыре шага (будем в дальнейшем называть их процедурой 1-4) соответствуют *i*-му уровню вложенности подзапроса (см. рис. 3).

Процедура 1 – 4 является рекурсивной и ее следует выполнить для всех *i*, начиная с (*n* – 1) до 1 (селекцию $\sigma_{A03=A14}$ будем считать пустой). Последовательно применяя процедуру 1 – 4 для каждого уровня вложенности (*i* меняется от *n* –1 до 1), план, представленный на рис. 3 (исходный план), преобразуем к дереву, которое изображено на рис. 8 (альтернативный план). Теперь покажем, что преобразования 1 – 4 являются корректными, т.е. после выполнения процесса, представленного на рис. 8, будет получен

правильный результат поиска данных.

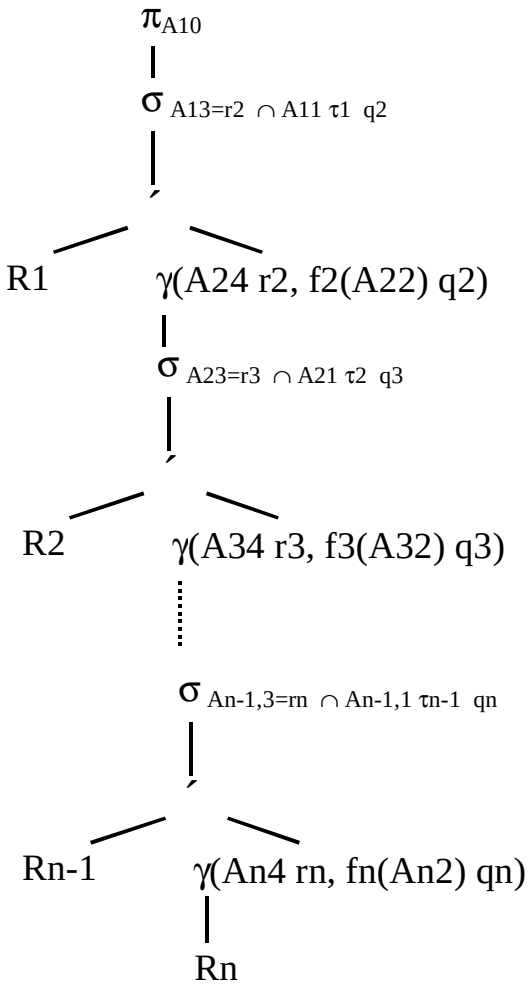

Рис. 8. Альтернативный план выполнения исходного запроса.

Перестановка операций $\sigma_{Ai-1,3=Ai4}$ и σ без индекса (см. рис. 4) является корректной в силу закона реляционной алгебры о коммутативности операции селекции. То же можно сказать и об операциях $\sigma_{Ai3=Ai+1,4}$ и σ без индекса (см. рис. 5). Только здесь селекция $\sigma_{Ai3=Ai+1,4}$ переносится вверх по дереву за пределы (i+1)-го подзапроса. То есть имеет место отложенная селекция. Чтобы это стало возможным, необходимо в (i+1)-м вложенном подзапросе определить все значения атрибута Ai+1,4, обеспечивающего связь с подзапросом более высокого уровня, и выполнить агрегирование Ai+1,2 для каждого значения Ai+1,4. Это можно реализовать за счет операции группирования γ (GROUP BY). Получаемое отношение γ (см. рис. 5) позволяет в дальнейшем связать два соседних подзапроса с учетом требуемых условий: Ai3=Ai+1,4 и Ai1 τi fi+1(Ai+1,2). Операция σ без индекса с аргументами Ri и <condition>(Ai1 τi fi+1(Ai+1,2)) эквивалентна операции селекции $\sigma_{Ai1\ \tau i\ qi+1}$, выполняемой над декартовым произведением отношений Ri и γ (см. рис. 6). Далее каскад селекций $\sigma_{Ai3=Ai+1,4}$ и $\sigma_{Ai1\ \tau i\ qi+1}$ можно объединить в одиночную селекцию (см. рис. 7) в силу соответствующего

закона реляционной алгебры. Итак, доказательство корректности преобразования дерева на рис. 3 в план, представленный на рис. 8, завершено.

В работах [3, 4] необоснованно утверждается, что только план на рис. 8 является оптимальным. Ряд СУБД (Oracle, DB2 и др.) этим планом и ограничиваются. В следующих разделах статьи будут определены условия, при которых целесообразно использовать тот или иной план выполнения исходного запроса (рис. 3 или рис. 8).

В некоторых СУБД, например Oracle, план на рис. 8 выполняется неэффективно на физическом уровне (здесь соединение $\sigma_{Ai3=ri+1\ \cap\ Ai1\ \tau i\ qi+1}$ реализуется путем фильтрации всех записей уровня i). Ниже приведены "*n*" SQL-операторов (рис. 9), совокупность которых соответствует плану на рис. 8.

SELECT	An4 as rn, fn(An2) as qn INTO TEMPn
FROM	Rn
GROUP BY	An4
Далее необходимо закодировать n-2 SQL-операторов, здесь i изменяется от n-1 до 2:	
SELECT	Ai4 as ri, fi(Ai2) as qi INTO TEMPi
FROM	Ri, TEMPi+1
WHERE	Ri.Ai3=TEMPi+1.ri+1 AND Ri.Ai1 τi TEMPi+1.qi+1
GROUP BY	Ai4
Затем должен быть выполнен n-ый запрос, который имеет номер 1:	
SELECT	A10
FROM	R1, TEMP2
WHERE	R1.A13=TEMP2.r2 AND R1.A11 τ1 TEMP2.q2

Рис. 9. Запросы SQL, моделирующие план на рис. 8.

В каждом запросе выполняется соединение 2 таблиц, кроме самого первого. Практически во всех СУБД предусмотрены эффективные методы соединения. Поэтому эти "n" SQL-операторов часто выполняются быстрее, чем один исходный запрос (см. рис. 1) в соответствии с планом на рис. 8.

Оценка времени выполнения запроса для исходного и альтернативного планов

Ниже приведены 4 теоремы о производящих функциях (ПФ) числа обработанных записей и преобразованиях Лапласа-Стилтьеса (ПЛС) времени выполнения запроса для исходного и альтернативного планов. Это важно, так как на этапе проектирования информационных систем конкретные значения полей записей базы данных и некоторые параметры запросов неизвестны и потому можно говорить об их случайном характере.

1. *Исходный план* (см. рис. 3).

Теорема 1. Производящая функция $H_i(z)$ числа записей, обработанных на i-м уровне вложенности и ниже для каждой записи $(i-1)$-го уровня, определяется следующей рекуррентной формулой:

$$H_i(z) = G_i(1 - p_i(1 - zH_{i+1}(z))), \quad i = \overline{1,n}, \tag{1}$$

$$G_i(z) = z^{V_i}, \tag{2}$$

где $G_i(z)$ – производящая функция числа записей в таблице R_i; V_i – прогнозируемое число записей в этой таблице

$$p_i = \begin{cases} 1, & для\ i = 1, \\ \dfrac{1}{I_{i4}}, & для\ i = \overline{2,n}; \end{cases} \tag{3}$$

I_{i4} – прогнозируемая мощность атрибута Ai4 (число разных значений); $H_{n+1}(z) \equiv 1$.

Следствие 1. Производящая функция числа записей базы данных, обработанных при выполнении запроса в соответствии с планом на рис. 3, равна

$$H(z) = H_1(z). \tag{4}$$

Теорема 2. ПЛС $T_i(s)$ времени обработки записей i-го уровня вложенности и ниже для каждой записи $(i-1)$-го уровня равна

$$T_i(s) = G_i(1 - p_i(1 - d_i(s)T_{i+1}(s)x_i(s))), \ i = \overline{1,n}, \tag{5}$$

где $G_i(z)$ и p_i определяются выражениями (2) и (3); $T_{n+1}(s) \equiv 1$, $d_i(s)$ – ПЛС времени поиска/чтения одной записи i-го уровня вложенности из БД; $x_i(s)$ – ПЛС времени сравнения в соответствии с условием Ai1 τi fi+1(Ai+1,2) для $i = \overline{1,n-1}$ и/или определения текущего значения функции агрегирования fi(Ai2) при успешном сравнении для $i = \overline{2,n}$ (см. рис. 3).

Следствие 2. ПЛС времени выполнения запроса в соответствии с планом на рис. 3 равно

$$T(s) = T_1(s). \tag{6}$$

2. *Альтернативный план* (в виде SQL-запросов, см. рис. 9).

Теорема 3. Производящая функция $W_i(z)$ числа записей, обработанных при выполнении i-го SQL-запроса определяется следующей формулой:

$$W_i(z) = Q_{i+1}(z)G_i(z)Q_i(z), \ i = \overline{n,1}, \tag{7}$$

$$Q_{n+1}(z) \equiv 1, \ Q_1(z) \equiv 1, \ Q_i(z) = z^{I_{i4}}, \ G_i(z) = z^{V_i}. \tag{8}$$

Следствие 3. ПФ $W(z)$ числа записей, обработанных в процессе выполнения всех n SQL-запросов, представленных на рис. 9, равна

$$W(z) = \prod_{i=1}^{n} W_i(z). \tag{9}$$

Теорема 4. ПЛС $Y_i(s)$ времени выполнения i-го SQL-запроса определяется следующим выражением:

$$Y_i(s) = W_i(d_i^*(s)) \cdot G_i(Q_{i+1}(x_i^*(s))), \ i = \overline{n,1}, \tag{10}$$

где ПФ $W_i(z)$ определяется выражением (7); $Q_{n+1}(x_n^*(s)) \equiv x_n^*(s)$ ПФ $G_i(z)$, $Q_i(z)$ определяются выражениями (8); $d_i^*(s)$ – ПЛС времени чтения/записи одной записи базы данных в таблицу при выполнении i-го запроса; $x_i^*(s)$ – ПЛС времени сравнения записей в соответствии с условием Ai3=TEMPi+1.ri+1 AND Ai1 τi TEMPi+1.qi+1 для $i = \overline{n-1,1}$ и/или определения текущего значения функции агрегирования fi(Ai2) при успешном сравнении для $i = \overline{n,2}$.

Следствие 4. ПЛС времени выполнения всех n SQL-запросов, т.е. времени выполнения исходного запроса (см. рис. 1), в соответствии с альтернативным планом (см. рис. 9) равно

$$Y(s) = \prod_{i=1}^{n} Y_i(s). \qquad (11)$$

Дифференцируя выражения (4) и (9), а также (6) и (11) несколько раз соответственно в точке $z = 1$ и $s = 0$, можно получать моменты высоких порядков числа обработанных записей и времени выполнения запроса. Что важно, так как эти моменты являются исходными данными при моделировании вычислительных систем на макроуровне, т.е. на уровне сети [2].

Продифференцировав как сложную функцию правую часть выражения (5) по s в нуле, получим

$$\overline{T}_i = -T_i^{'}(0) = \overline{G_i} \cdot p_i \cdot (\overline{d}_i + \overline{T}_{i+1} + \overline{x}_i),\ i = \overline{1,n}, \qquad (12)$$

где $\overline{G}_i = V_i$ – число записей в таблице R_i; p_i – вероятность, определяемая формулой (3); $\overline{d}_i$ – математическое ожидание времени поиска/чтения одной записи i-го уровня вложенности из БД; $\overline{x}_i$ – математическое ожидание времени сравнения и определения текущего значения функции агрегирования для i-го уровня, $\overline{T}_{n+1} = 0$.

Из (6) следует, что математическое ожидание времени выполнения запроса, реализуемого в соответствии с исходным планом (см. рис. 3), равно

$$\overline{T} = \overline{T}_1, \qquad (13)$$

где $\overline{T}_1$ определяется рекуррентной формулой (12).

Из (11) и (10) можно получить выражение для математического ожидания времени выполнения запроса, реализуемого в соответствии с альтернативным планом в виде SQL-запросов (рис. 9)

$$\overline{Y} = -Y^{'}(0) = \\ = V_n(\overline{d}_n^* + \overline{x}_n^*) + \sum_{i=1}^{n-1} I_{i+1,4}(\overline{d}_i^* + \overline{d}_{i+1}^*) + \sum_{i=1}^{n-1} V_i(\overline{d}_i^* + I_{i+1,4}\overline{x}_i^*). \qquad (14)$$

Используя (13) и (14), можно сравнивать планы по времени реализа-

ции и выбирать наилучший из них.

Сравнение планов выполнения запроса

Ниже определен подкласс запросов, задаваемый следующими выражениями:

$$V_i = V,\ \bar{d}_i = \bar{d}_i^* = \bar{d},\ \bar{x}_i = \bar{x}_i^* = \bar{x},\ i = \overline{1,n}, \qquad (15)$$

$$I_{i+1,4} = \frac{I_{i4}}{k},\ i = \overline{2,n-1},\ 1 \le I_{i4} \le V,\ I_{24} = I,$$

где k – коэффициент изменения мощности атрибута связи при переходе с одного уровня вложенности подзапросов на другой уровень, расположенный ниже.

Полагая, что $\bar{d} >> \bar{x}$ и $V\bar{x} >> 2\bar{d}$, из (13) и (14) можно получить выражения для математических ожиданий времени реализации исходного и альтернативного планов для данного подкласса запросов:

$$\bar{T} = \bar{d}\sum_{i=1}^{n}\frac{V^i}{I^{i-1}}k^{\frac{(i-1)(i-2)}{2}},\ \bar{Y} = \bar{d}nV + \bar{x}VI\frac{1-\frac{1}{k^{n-1}}}{1-\frac{1}{k}}. \qquad (16)$$

Ниже рассмотрены различные варианты для n и k и построены области преимущественного использования этих планов в пространстве (k, I) (рис. 10).

В случае $n = 2$ время $\bar{T}$ и $\bar{Y}$ не зависит от k (см. (16)). Поэтому область предпочтительного использования альтернативного плана на рис. 8 (в форме SQL-запросов – рис. 9) соответствует неравенству $1 \le I < I_1^*$, при $I_1^* < I \le V$ более целесообразно применять исходный план на рис. 3.

Рассмотрим теперь случай $n > 2$. Кривая "A" на рис. 10 может быть получена из уравнения $\bar{T} = \bar{Y}$ (см. (16)). Кривая "A" разделяет область $(0 < k \le \sqrt[n-2]{V}] \times [1 \le I \le V]$ на две части. Нижняя часть соответствует более предпочтительному использованию альтернативного плана (см. рис. 8 и рис. 9), верхняя часть – более предпочтительному использованию исходного плана (см. рис. 3). Для какого-либо I (см. рис. 10) k может изменяться на отрезке $[k_{\min}, k_{\max}]$. На отрезке от точки "a" до точки "b" целесообразно использовать исходный план, а на отрезке от "b" до "c" – альтернативный.

Заключение

1. На основе законов реляционной алгебры разработан метод преобразования исходного плана выполнения запроса с многоуровневыми вложенными коррелированными подзапросами и операциями агрегирования в

альтернативный план, что позволяет выбирать план с наименьшим временем реализации в информационных системах.

2. Получены формулы для производящей функции числа обработанных записей и преобразования Лапласа-Стилтьеса времени выполнения запроса для исходного и альтернативного плана, позволяющие вычислять математические ожидания и моменты более высоких порядков.

3. Для исходного и альтернативного плана выполнено сравнение математических ожиданий времени выполнения запроса из определенного подкласса для различных случаев и построены области преимущественного использования этих планов.

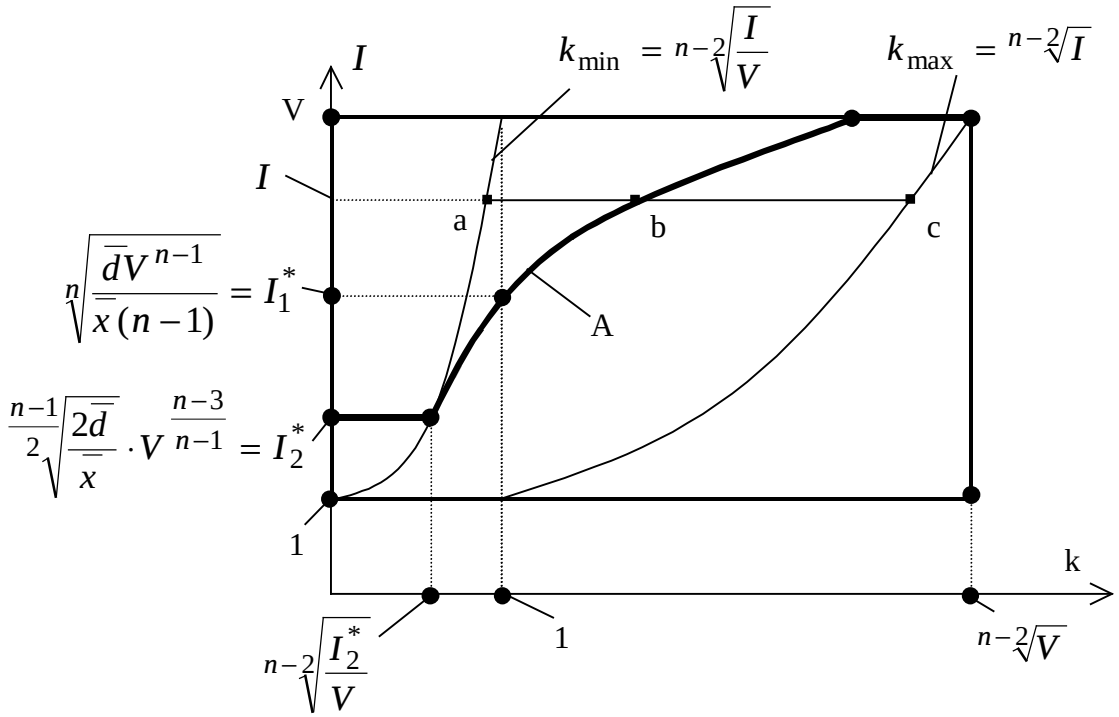

Рис. 10. Области преимущественного использования исходного и альтернативного плана.

ЛИТЕРАТУРА

1. *Graefe G.* Query evaluation techniques for large databases // ACM Computing Surveys. – N.Y., 1993. – Vol. 25, N. 2. – P. 73-170.
2. *Григорьев Ю.А., Плутенко А.Д.* Теоретические основы анализа процессов доступа к распределенным базам данных. – Новосибирск: Наука, 2002.
3. *Гарсиа-Молина Г., Ульман Дж., Уидом Дж.* Системы баз данных. Полный курс. – М.: Издательский дом "Вильямс", 2003.
4. *Шаша Д., Бонне Ф.* Оптимизация баз данных: принципы, практика, решение проблем. – М.: КУДИЦ-ОБРАЗ, 2004.