УДК 004.65

© 2004 г. **Ю.А. Григорьев**, д-р техн. наук,
В.Г. Матюхин
(Московский государственный технический университет им. Н.Э.Баумана)

ОЦЕНКА ВРЕМЕНИ ВЫПОЛНЕНИЯ СЛОЖНОГО SQL-ЗАПРОСА В СУБД MS SQL SERVER 2000

В работе анализируются результаты оптимизации сложного SQL-запроса в СУБД MS SQL Server 2000 для сложной схемы базы данных при использовании различных языковых конструкций предложения SELECT и индексов.

Введение

Чем сложнее предметная область, тем более трудоемкие запросы выполняются в соответствующей автоматизированной системе. Разработка таких запросов превращается в непростую задачу, так как при выполнении они потребляют много ресурсов, а при их проектировании затруднительно подобрать параметры, уменьшающие время их выполнения.

Рассмотрим систему расчета железнодорожных тарифов, которая в зависимости от исходных данных (тип груза, вид вагона и т.д.) должна вычислять стоимость перевозки груза между станциями. На первый взгляд, это простая задача: берется расстояние между станциями, задается тип груза и вид вагона и по формуле выполняется расчет стоимости. Но как всегда появляется множество "если". Тарифы негибкие, не проходит и недели как поступает телеграмма, которая устанавливает новые условия. Вот пример такой телеграммы:
"КОЭФФИЦИЕНТ 0,85 К СТАВКАМ ПЛАТ РАЗДЕЛА 3 ПРЕЙСКУРАНТА НР 10-01 «ТАРИФЫ НА ПЕРЕВОЗКИ ГРУЗОВ И УСЛУГИ ИНФРАСТРУКТУРЫ, ВЫПОЛНЯЕМЫЕ РОССИЙСКИМИ ЖЕЛЕЗНЫМИ ДОРОГАМИ» ЗА ПЕРЕВОЗКИ СКОРОПОРТЯЩИХСЯ ГРУЗОВ (ПОЗ. О41-О43, 051-053, 551-555, 561-564, 572-574, 581, 584, 591-592, 595 ЕТСНГ) В РОССИЙСКОМ РЕФРИЖЕРАТОРНОМ ПОДВИЖНОМ СОСТАВЕ, СЛЕДУЮЩЕМ С ПОДДЕРЖАНИЕМ ТЕМПЕРАТУРНОГО РЕЖИМА НА СТАНЦИИ РЖД ЧЕРЕЗ ПОГРАНПЕРЕХОДЫ СО СТРАНАМИ БАЛТИИ (ЛИТВА, ЛАТВИЯ, ЭСТОНИЯ)".

В данной телеграмме можно насчитать, как минимум, четыре параметра, которые явно влияют на применение коэффициента '0,85': интервалы кодов грузов, обязатееееельность перевозки груза в рефрижераторах России через погранпереходы со странами Балтии, применение коэффициента для перевозок по России. И это не предел.

Так как тарифы обновляются часто, необходимо иметь систему, которая бы позволяла оперативно и гибко учитывать новые условия без исправления кода, который реализует логику расчета (хотя даже в данной системе без этого не обходится). Поэтому важно, чтобы схема базы данных системы поддерживала требуемые изменения.

Схема базы данных

На рис. 1 приведен фрагмент схемы базы данных, используемой при выполнении запроса для поиска данных, которые применяются для пересчета тарифа перевозки груза.

Для экономии места здесь показаны не все таблицы. При построении схемы базы данных использовалось следующее правило: каждому полю таблицы ITCommon, начиная с поля ITGruz и заканчивая ITMassa, соответствует таблица (таблицы справа ITGruz, ITClass, ..., ITMassa). Каждая из этих таблиц имеет один и тот же набор атрибутов:

ITCode – поле, содержащее код какого-либо параметра отбора записи из таблицы ITCommon;

ITLow – верхняя граница параметра;

ITHigh – нижняя граница параметра.

В этих таблицах каждому значению ITCode соответствует интервал значений, которые может принимать параметр. Но одному значению ITCode может соответствовать несколько интервалов. Причем для того, чтобы параметр мог не учитываться при отборе записей из таблицы ITCommon для ITCode = 0, устанавливаются самые предельные значения (например, для таблицы ITGruz установлен интервал от ‘00000’ до ‘99999’).

Единственная таблица, которая отличается от других, – это ITSoob. Она также создана для отбора записей из таблицы ITCommon и имеет схожую с ней структуру. Таблица ITSoob связана с таблицами типа SBStanL, имеющими ту же структуру, что и таблицы, описанные выше.

Показанная на рисунке структура является логической, между таблицами установлена в основном связь "многие ко многим". На физическом уровне промежуточные таблицы отсутствуют. Можно было бы создать промежуточные таблицы для реализации этой связи (их автоматически генерирует пакет ERwin), но тогда количество таблиц увеличилось бы примерно вдвое, что сделало бы структуру базы данных непонятной и менее гибкой.

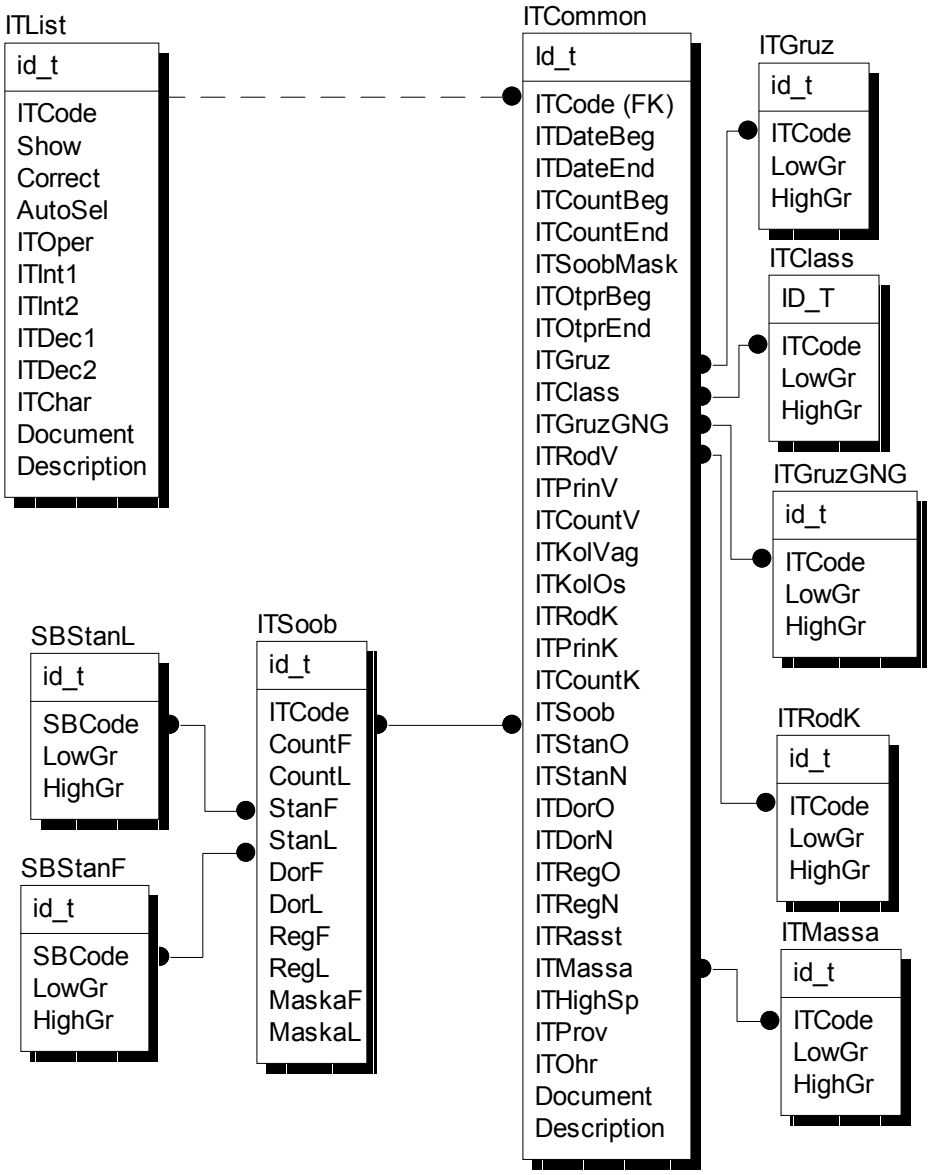

Рис. 1. Фрагмент схемы базы данных.

Конфигурация сервера базы данных

Далее приведены результаты натурных экспериментов для различных вариантов кодирования сложного запроса, часто выполняемого при расчете железнодорожных тарифов. Эксперименты проводились с использованием сервера, имеющего следующую конфигурацию:

двухпроцессорный сервер с процессорами – Pentium III 933 Мгц;
материнская плата – Intel Server Board SIA2;
оперативная память – 1 Гбт;
операционная система – Windows 2003 Server;
СУБД – MSSQL 2000 Enterprise Edition;
средняя загрузка сервера – 50%;
внешняя память: Seagate Cheetah – 72 Гб.

На момент проведения экспериментов в базе данных хранилось около 3000 записей.

Вариант 1: запрос с использованием подзапроса и без индексации атрибутов связи (исходный запрос)

Чтобы рассчитать тариф, необходимо выбрать записи из таблицы ITList (см. рис. 1). Они содержат данные, необходимые для вычислений непосредственно уже в самой программе (хранимой процедуре). Записи выбираются по исходным данным и по данным, полученным в программе. Для простоты вместо переменных в запросе явно указаны конкретные значения (они выделены жирным шрифтом). Текст запроса приведен ниже.

```
/*****************************************************************************/
SELECT it.*
FROM ITList it, ITCommon c
WHERE it.ITCode = c.ITCode and  c. id_t IN
(
SELECT ID_T                                    // подзапрос 1
FROM ITCommon
WHERE
(‘20040426’ BETWEEN ITDateBeg and ITDateEnd) AND
(‘20’ BETWEEN ITCountBeg and ITCountEnd) AND
(‘02020’ LIKE RTRIM( ITSoobMask))  AND
(‘01’ BETWEEN ITOtprBeg and ITOtprEnd ) AND
(ITGruz IN (SELECT ITCode FROM ITGruz WHERE ISNULL(‘01100’,'00000') between LowGr
and HighGr ) ) AND
(ITGruzGNG IN (SELECT ITCode FROM ITGruzGNG WHERE ISNULL(‘100110’,'000000')
between LowGr and HighGr)) AND
(ITClass IN (SELECT ITCode FROM ITClass WHERE ISNULL(1,0) between LowGr and HighGr))
AND
(ITKolVag IN (SELECT ITCode FROM ITKolVag WHERE ISNULL(1,0) between LowGr and
HighGr)) AND
(ITRasst IN (SELECT ITCode FROM ITRasst  WHERE ISNULL( 54,0) between LowGr and
HighGr)) AND
(ITRodV IN (SELECT ITCode FROM ITRodV WHERE ISNULL(‘0950’,'0000') between LowGr and
HighGr)) AND
(ITPrinV IN (SELECT ITCode FROM ITPrinV  WHERE ISNULL(0, 0) between LowGr and
HighGr)) AND
(ITCountV IN (SELECT ITCode FROM ITCountV WHERE ISNULL(‘20’,'00') between LowGr and
HighGr)) AND
(ITRodK IN (SELECT ITCode FROM ITRodK WHERE ISNULL(‘0000’,'0000') between LowGr and
HighGr)) AND
(ITPrinK IN (SELECT ITCode FROM ITPrinK  WHERE ISNULL(0,0) between LowGr and
HighGr)) AND
(ITCountK IN (SELECT ITCode FROM ITCountK WHERE ISNULL(‘00’,'00') between LowGr and
HighGr)) AND
(ITStanO IN (SELECT ITCode FROM ITStanO  WHERE ISNULL(‘19350’,'00000') between LowGr
and HighGr)) AND
(ITStanN IN (SELECT ITCode FROM ITStanN WHERE ISNULL(‘19150’,'00000') between LowGr
and HighGr)) AND
```

```
(ITDorO IN (SELECT ITCode FROM ITDorO WHERE ISNULL(‘17’,'00') between LowGr and
HighGr)) AND
(ITDorN IN (SELECT ITCode FROM ITDorN WHERE ISNULL(‘17’,'00') between LowGr and
HighGr)) AND
(ITRegO IN (SELECT ITCode FROM ITRegO WHERE ISNULL(‘2045000’,'0000000') between
LowGr and HighGr)) AND
(ITRegN IN (SELECT ITCode FROM ITRegN WHERE ISNULL(‘2045000’,'0000000') between
LowGr and HighGr)) AND
(ITMassa IN (SELECT ITCode FROM ITMassa  WHERE ISNULL(67,0) between LowGr and
HighGr)) AND
(ITKolOs IN (SELECT ITCode FROM ITKolOs  WHERE ISNULL(4,0)  between LowGr and
HighGr))  AND
ITSoob  IN
  (
  SELECT ITCode
  FROM ITSoob
  WHERE
  (MaskaF=0 OR (MaskaF & 0)>0) AND ( MaskaL=0 OR (MaskaL & 0 )>0) AND
  (CountF IN (SELECT SBCode FROM SBCountF WHERE  ISNULL(‘20’,'00') between LowGr and
  HighGr ) ) AND
  (CountL IN (SELECT SBCode FROM SBCountL WHERE ISNULL(‘20’,'00') between LowGr and
  HighGr)) AND
  (StanF IN (SELECT SBCode FROM SBStanF  WHERE  ISNULL(‘19350’,'00000') between
  LowGr and HighGr)) AND
  (StanL IN (SELECT SBCode FROM SBStanL  WHERE  ISNULL(‘19150’, '00000' ) between
  LowGr and HighGr ) ) AND
  (DorF IN (SELECT SBCode FROM SBDorF  WHERE ISNULL(‘17’,'00') between LowGr and
  HighGr)) AND
  (DorL IN (SELECT SBCode FROM SBDorL WHERE ISNULL(‘17’,'00')  between LowGr and
  HighGr)) AND
  (RegF IN (SELECT SBCode FROM SBRegF WHERE ISNULL(‘2045000’,'0000000') between
  LowGr and HighGr)) AND
  (RegL IN (SELECT SBCode FROM SBRegL  WHERE ISNULL(‘2045000’,'0000000') between
  LowGr and HighGr))
  )
AND
(ITHighSp IS  NULL or  ITHighSp = 0) AND
(ITProv IS  NULL or ITProv = 0) AND
( ITOhr IS NULL or ITOhr = 0)
)

/*************************************************************************/
```

В результате натурных экспериментов было получено время выполнения приведенного выше запроса для различных кодов грузов ITGruz (итерации 1...20). Измеренные значения времени выполнения исходного запроса приведены в табл. 1.

Таблица 1

Номер итерации	1	2	3	4	5	6	7	8	9	10
Время выполнения, сек.	0,076	0,093	0,076	0,08	1,080	0,076	0,08	0,076	0,08	0,08

Продолжение табл. 1

Номер итерации	11	12	13	14	15	16	17	18	19	20
Время выполнения, сек.	0,093	0,076	0,060	0,08	1,080	0,076	0,08	0,063	0,08	0,08

Итак, среднее время выполнения запроса по всем 20 итерациям составило 0,1285 сек.

Вариант 2: запрос с использованием подзапроса и с индексацией атрибутов связи

Для атрибутов ITGruz ... ITMassa таблицы ITCommon и атрибутов ITCode соответствующих таблиц ITGruz, ..., ITMassa были созданы индексы. Сам запрос не менялся (см. вариант 1). В результате натурных экспериментов было получено время выполнения исходного запроса для различных кодов грузов ITGruz (табл. 2).

Таблица 2

Номер итерации	1	2	3	4	5	6	7	8	9	10
Время выполнения, сек.	0,046	0,046	0,046	0,063	0,063	0,063	0,046	0,06	0,046	0,046

Номер итерации	11	12	13	14	15	16	17	18	19	20
Время выполнения, сек.	0,046	0,046	0,05	0,046	0,043	1,05	0,046	0,046	0,05	0,046

Для этого варианта среднее время выполнения запроса по всем 20 итерациям составило 0,0997 сек.

Вариант 3: запрос без использования подзапроса и без индексации атрибутов связи

Анализ исходного запроса (см. вариант 1) показывает, что "подзапрос 1" является лишним, т.к. соединение таблиц ITList it и ITCommon снова соединяется с таблицей ITCommon.

После внесения изменений запрос преобразуется к следующему виду:

```
/*************************************************************************/
SELECT it.*
FROM ITList it, ITCommon c
WHERE it.ITCode = c.ITCode and
// условие, указанное в подзапросе 1
```

(**‘20040426’** BETWEEN ITDateBeg and ITDateEnd) AND
и т. д.

Для данного запроса был приведен ряд экспериментов. Время выполнения запросов представлено в табл. 3.

Таблица 3

Номер итерации	1	2	3	4	5	6	7	8	9	10
Время выполнения, сек.	0,08	0,076	0,063	0,08	0,063	0,063	0,06	0,06	0,063	0,076

Номер итерации	11	12	13	14	15	16	17	18	19	20
Время выполнения, сек.	0,06	0,063	0,076	0,08	0,06	1,063	0,076	0,08	0,076	0,063

Среднее время выполнения запроса составило 0,11905 сек.

Вариант 4: запрос без использования подзапроса и с индексацией атрибутов связи

Для 3-го варианта запроса были созданы индексы для атрибутов ITGruz ... ITMassa таблицы ITCommon и атрибутов ITCode соответствующих таблиц ITGruz, ..., ITMassa.

В результате натурных экспериментов было получено время выполнения запроса для различных кодов грузов ITGruz (табл. 4).

Таблица 4

Номер итерации	1	2	3	4	5	6	7	8	9	10
Время выполнения, сек.	0,046	0,046	0,043	0,063	0,046	0,06	1,05	0,046	0,046	0,046

Номер итерации	11	12	13	14	15	16	17	18	19	20
Время выполнения, сек.	0,05	0,046	0,063	0,046	0,046	0,06	0,046	0,046	0,046	0,043

Среднее время выполнения запроса по всем 20 итерациям составило 0,0992 сек.

Анализ результатов четырех экспериментов показывает, что создание индексов атрибутов более существенно влияет на время выполнения запроса, чем удаление лишнего подзапроса.

Вариант 5: запрос без использования подзапроса с индексацией атрибутов связи и применением функции агрегации COUNT

Используемые в исходном запросе (см. вариант 1) конструкции IN можно заменить на эквивалентные по действию операторы SELECT с функцией агрегации COUNT. Например, следующие конструкции эквивалентны:

1. (ITGruz IN (SELECT ITCode FROM ITGruz WHERE ISNULL(**'01100'**,'00000') between LowGr and HighGr))
2. (SELECT COUNT(i. id_t) FROM ITGruz i WHERE ITCode = c.ITGruz and (ISNULL(**'01100'**, '00000') BETWEEN LowGr and HighGr)) > 0

Запрос с функциями COUNT имеет следующий вид:

```
/*************************************************************************/
SELECT it.*
FROM ITList it, ITCommon c
WHERE it.ITCode = c.ITCode and
// условие, указанное в подзапросе 1
 ('20040426' BETWEEN ITDateBeg and ITDateEnd) AND
('20' BETWEEN ITCountBeg and ITCountEnd) AND
('02020' LIKE RTRIM( ITSoobMask))  AND
('01' BETWEEN ITOtprBeg and ITOtprEnd ) AND
//  преобразование конструкций IN в опреаторы SELECT с функцией агрегации COUNT
( SELECT COUNT(i. id_t) FROM ITGruz i  WHERE ITCode = c.ITGruz and  (ISNULL( '01100',
'00000' )  BETWEEN LowGr and HighGr ) ) > 0
AND
( SELECT COUNT(i. id_t) FROM ITClass i      WHERE ITCode = c.ITClass and  (ISNULL( 1, 0 )
between LowGr and HighGr ) ) >0
и т.д.
```

Результаты экспериментов (время выполнения запроса) для этого варианта приведены в табл. 5.

Таблица 5

Номер итерации	1	2	3	4	5	6	7	8	9	10
Время выполнения, сек.	0,06	0,063	0,063	0,046	0,046	0,06	0,05	0,046	0,63	0,63

Номер итерации	11	12	13	14	15	16	17	18	19	20
Время выполнения, сек.	0,063	0,06	0,063	1,63	0,063	0,063	0,063	0,063	0,6	0,06

Здесь среднее время запроса равно 0,2211 сек., т.е. использование функции агрегации COUNT приводит к резкому увеличению (почти в два раза) времени выполнения запроса.

Это свидетельствует, что применение в данном случае функции COUNT приводит к увеличению потребления ресурсов сервера.

Вариант 6: выборка записей из индексированных таблиц с использованием курсора и применением функции агрегации COUNT

Идея заключается в организации соединения таблиц ITCommon и ITList, указанных в запросе (см. вариант 1), программным путем. Для этой цели в программе создаются два курсора.

С помощью 1-го курсора читаются и обрабатываются записи таблицы ITCommon. Для каждой записи таблицы ITCommon, удовлетворяющей условию поиска, создается вложенный курсор, с помощью которого читаются записи таблицы ITList. Спецификации этой программы приведены ниже.

```
/*************************************************************************/
// Описание курсора для таблицы ITCommon и его открытие
/*************************************************************************/
DECLARE cur_ITCommon CURSOR SCROLL FOR
SELECT ID_T, ITCode, и т.д.
FROM ITCommon
WHERE
   ( '20040426' BETWEEN ITDateBeg and ITDateEnd ) and
   ( '20' BETWEEN ITCountBeg and ITCountEnd ) and
   ( '02020' LIKE RTRIM( ITSoobMask ) )  and
   ('01' BETWEEN ITOtprBeg and ITOtprEnd )   and
   ( SELECT count(i.ID_T) FROM SBCountF i WHERE SBCode = s.CountF and (ISNULL( '20',
     '00' )  between LowGr and HighGr ) ) >0 AND и т. д.
OPEN cur_ITCommon
/*************************************************************************/
// Организация цикла и проверка дополнительных условий для таблицы ITCommon
/*************************************************************************/
FETCH NEXT FROM cur_ITCommon INTO  @C_ID_T, и т. д.
WHILE @@FETCH_STATUS = 0
BEGIN
IF  ( SELECT COUNT (i. id_t) FROM ITGruz i   WHERE ITCode = @C_ITGruz and  (ISNULL(
@Gruz, '00000' ) between LowGr and HighGr ) ) > 0
и т.д.
     BEGIN
/*************************************************************************/
// Описание курсора для таблицы ITList и его открытие
/*************************************************************************/
    DECLARE cur_ITList CURSOR SCROLL FOR
    SELECT AutoSel, и т.д.
    FROM ITList
    WHERE ITCode = @C_ITCode
    OPEN cur_ITList
/*************************************************************************/
// Организация цикла и чтение атрибутов таблицы ITList, закрытие курсора
/*************************************************************************/
  FETCH NEXT FROM cur_ITList INTO  @L_AutoSel, и т.д.
  CLOSE cur_ITList
```

```
  DEALLOCATE cur_ITList
  END
// обработка следующей записи таблицы ITCommon
FETCH NEXT FROM cur_ITCommon INTO  @C_ID_T, и т.д.
END
// закрытие курсора для таблицы ITCommon
CLOSE cur_ITCommon
DEALLOCATE cur_ITCommon
/*************************************************************************/
```

Результаты экспериментов для рассматриваемого варианта приведены в табл. 6.

Таблица 6

Номер итерации	1	2	3	4	5	6	7	8	9	10
Время выполнения, сек.	0,14	0,14	0,153	0,156	0,14	0,14	0,173	0,156	0,153	0,153

Номер итерации	11	12	13	14	15	16	17	18	19	20
Время выполнения, сек.	0,143	0,156	1,25	0,14	1,14	0,14	0,156	0,153	0,156	0,14

Для данного варианта среднее время выполнения запроса равно 0,2539 сек. Таким образом, применение курсоров для соединения таблиц в данном случае не приводит к уменьшению времени выполнения запроса.

Заключение

1. Создание индексов для атрибутов связи между таблицами позволило уменьшить время выполнения исходного запроса к базе данных в среднем на 22% (варианты 1 и 2).

2. Вопреки ожиданиям удаление внутреннего подзапроса 1 не привело к существенному уменьшению времени выполнения исходного запроса (варианты 1 и 3). Выигрыш составил в среднем 7%.

3. Использование функции агрегации COUNT (вариант 5) вызвало увеличение времени выполнения запроса почти в два раза.

4. Соединение таблиц с помощью курсоров (вариант 6) также не приводит к желаемому результату. Запрос в виде одного оператора SELECT выполняется быстрее. По-видимому, СУБД применяет более тонкие методы соединения таблиц [1,2], чем используемые в программе вложенные циклы, когда каждая запись 1-й таблицы сравнивается с каждой записью 2-й соединяемой таблицы.

ЛИТЕРАТУРА

1. *Григорьев Ю.А., Плутенко А.Д.* Теоретические основы анализа процессов доступа к распределенным базам данных. Новосибирск: Наука, 2002.
2. *Гарсиа-Молина Г., Ульман Д., Уидом Д.* Системы баз данных. Полный курс. М.: Издат. дом "Вильямс", 2003.