Подсказки для оптимизатора

Графеева Н.Г.

2017

План запроса

- Практически любую задачу по получению каких-либо результатов из базы данных можно решить несколькими способами, т.е. написать несколько разных запросов, которые дадут один и тот же результат. Это, однако не означает, что база данных эти запросы будет выполнять по-разному. Также неверно мнение о том, что структура запроса может повлиять на то, как Oracle будет его выполнять (это касается порядка временных таблиц, **JOINS** и условий отбора в **WHERE**). Решение о том, как построить запрос принимает оптимизатор Oracle. Алгоритм получения сервером данных для конкретного запроса называют **планом запроса**. Практически все продукты для работы с базой данных Oracle позволяют просмотреть план конкретного запроса. ORACLE Apex также не является исключением.

Как это выглядит в ORACLE APEX?

```
select * from dept where dept.deptno in (select emp.deptno from emp group by emp.deptno having count(*) > 3)
```

Results | Explain | Describe | Saved SQL | History

Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			1	1	6	33		
HASH JOIN			1	1	6	33		"DEPT"."DEPTNO" = "DEPTNO"
NESTED LOOPS			1	1	6	33		
NESTED LOOPS			1	1	6	33		
STATISTICS COLLECTOR								
VIEW		VW_NSO_1	1	1	4	13		
FILTER							COUNT(*)>3	
HASH	GROUP BY		1	1	4	3		
TABLE ACCESS	STORAGE FULL	EMP	15	1	3	45		
INDEX	UNIQUE SCAN	SYS_C0036811693	1	1	0			"DEPT"."DEPTNO" = "DEPTNO"
TABLE ACCESS	BY INDEX ROWID	DEPT	1	1	1	20		
TABLE ACCESS	STORAGE FULL	DEPT	1	1	1	20		

* Unindexed columns are shown in red

Некоторые термины в плане запроса

- **TABLE ACCESS FULL** — сервер просмотрит все записи таблицы.
- **TABLE ACCESS BY INDEX ROWID** — таблица будет просмотрена частично с помощью индекса.
- **INDEX RANGE SCAN** — для получения выборки нужных значений будет использован индекс таблицы.
- **HASH JOIN** — для получения выборки нужных значений будет построена хэш-таблица.
- **NESTED LOOPS** — нужные значения будут получены путем полного просмотра основной таблицы и поиском записей во вспомогательной. Это реализация схемы доступа **«один — ко многим»**, т.е. в качестве основной таблицы будет выбрана та в которой наименьшее количество записей, на основе этих записей будет производиться поиск во вспомогательной таблице.

Некоторые термины в плане запроса

- **SORT MERGE JOIN** — используется для соединения записей нескольких независимых источников. Сначала оба источника сортируются по объединяющему ключу, а затем происходит из слияние.
- **BUFFER SORT** — в некоторых случаях Oracle может определить, что при выполнении запроса обращение к некоторому блоку данных может быть выполнено несколько раз, в этом случае Oracle помещает этот блок в специальную область, чтобы ускорить к нему доступ. Запрос может не иметь ключевого слова **SORT**, но при его выполнении будет вызвана эта операция.
- **MERGE JOIN CARTESIAN** — для получения выборки нужных значений будет организовано перемножение записей в двух таблицах (для каждой записи основной таблицы будут просмотрены все записи вспомогательной). Это очень плохая операция, ее наличие в плане запроса говорит о том, что скорей всего упущена какая-то связка в JOIN.

Анализ плана запроса

- При анализе плана запроса необходимо примерно представлять объемы записей в таблицах и наличие у них индексов, которые могут пригодиться при фильтрации записей. Для доступа к данным Oracle использует несколько стратегий, какие из них выбраны для каждой из таблиц можно понять из плана запроса. При просмотре плана, необходимо решить, правильная ли выбрана стратегия в том или ином случае. Далее приведены краткие описания способов доступа и механизмов отбора записей при соединениях результирующих наборов.

Full Table Scan (Table Access Full).

- Может показаться, что доступ к данным таблицы быстрее осуществлять через индекс, но это не так. Иногда дешевле прочитать всю таблицу целиком, чем прочитать, например, 80% записей таблицы через индекс, так как чтение индекса тоже требует ресурсов. Очень нежелательна ситуация, когда эта операция стоит первой в объединении наборов записей и таблица, которая читается полностью, большая. Еще хуже ситуация с большой таблицей на второй позиции в объединении, это означает, что она также будет прочитана полностью, а если объединение производится через **NESTED LOOPS**, то таблица будет читаться несколько раз, поэтому запрос будет работать очень долго.

Nested Loops

- Такое соединение может использоваться оптимизатором, когда небольшой основной набор записей (стоит первым в плане запроса) объединяется с помощью условия, позволяющего эффективно выбрать записи из второго набора. Важным условием успешного использования такого соединения является наличие связи между основным и второстепенным набором записей. Если такой связи нет, то для каждой записи в первом наборе, из второго набора будут извлекаться одни и те же записи, что может привести к значительному увеличению времени запроса. Если вы видите, что в плане запроса применен **NESTED LOOPS**, а соединяемые наборы не удовлетворяют этому условию, то это плохой запрос.

Hash Joins

- Используется при соединении больших наборов данных. Оптимизатор использует наименьший из наборов данных для построения в памяти хэш-таблицы по ключу соединения. Затем он сканирует большую таблицу, используя хэш-таблицу для нахождения записей, которые удовлетворяют условию объединения. Оптимизатор использует **HASH JOIN**, если наборы данных соединяются с помощью операторов и ключевых слов эквивалентности (**=, AND**) и если присутствует одно из условий:

  ■ Необходимо соединить наборы данных большого объема.
  ■ Большая часть небольшого набора данных должна быть использована в соединении.

Sort Merge Join

- Данное соединение может быть применено для независимых наборов данных. Обычно Oracle выбирает такую стратегию, если наборы данных уже отсортированы ранее, и если дальнейшая сортировка результата соединения не требуется. Обычно это имеет место для наборов, которые соединяются с помощью операторов **<, <=, >, >=**. Для этого типа соединения нет понятия главного и вспомогательного набора данных, сначала оба набора сортируются по общему ключу, а затем сливаются в одно целое. Если какой-то из наборов уже отсортирован, то повторная сортировка для него не производится.

Cartesian Joins

- Это соединение используется, когда одна и более таблиц не имеют никаких условий соединения с какой-либо другой таблицей в запросе. В этом случае произойдет объединение каждой записи из одного набора данных с каждой записью в другом. Наличие такого соединения может (но не обязательно) означать присутствие серьезных проблем в запросе. В этом случае, возможно, упущены дополнительные условия соединения наборов данных.

Определение

- Подсказка (hint) – это указание оптимизатору на необходимость исполнения определенной формы доступа к данным на некотором шаге построения плана исполняемого запроса.

Синтаксис

- {DELETE|INSERT|SELECT|UPDATE} /*+ hint [text] [hint[text]]... */...
- or
- {DELETE|INSERT|SELECT|UPDATE} --+ hint [text] [hint[text]]...

Примеры

- ==========
- SELECT
- /*+ ALL_ROWS */
- empno, ename, sal, job
- FROM emp
- WHERE ename = ‘CAT’;

- ==========

- SELECT
- --+ RULE
- empno, ename, sal, job
- FROM emp
- WHERE empno > 7566;
- ==========

Что будет, если подсказка написана неправильно…

- ORACLE игнорирует подсказки, которые не следуют за ключевыми словами DELETE, INSERT, SELECT or UPDATE (рассматривает, как простые комментарии).
- ORACLE игнорирует подсказки, написанные с синтаксическими ошибками , но при этом учитывает правильные подсказки, написанные в этом же операторе.
- ORACLE игнорирует конфликтующие подсказки, но при этом учитывает правильные подсказки, написанные в этом же операторе.

Группы подсказок

Подсказки можно разделить на следующие группы:

- подсказки задающие цели оптимизации
- подсказки задающие методы доступа
- подсказки для операций соединения
- другие подсказки

Подсказки, задающие цели оптимизации

- ALL_ROWS
- FIRST_ROWS(n)
- CHOOSE
- RULE

Пример (ALL_ROWS)

```
SELECT /*+ ALL_ROWS */
        employee_id,
        last_name,
        salary,
        job_id
FROM employees
```

Пример (FIRST_ROWS(n))

- SELECT
- /*+ FIRST_ROWS(10) */
- empno, ename, sal, job
- FROM emp

Пример (CHOOSE)

- SELECT
- /*+ CHOOSE */
- empno, ename, sal, job
- FROM emp
- WHERE empno = 7566;

Пример (RULE)

- SELECT
- --+ RULE
- empno, ename, sal, job
- FROM emp
- WHERE empno = 7566;

Подсказки, задающие методы доступа

- FULL
- ROWID
- INDEX
- INDEX_ASC
- INDEX_DESC
- INDEX_FFS
- NO_INDEX
- INDEX_COMBINE
- INDEX_JOIN
- ….

Пример (FULL)

- SELECT /*+ FULL(e) */
- employee_id,
- last_name
- FROM hr.employees e
- WHERE last_name LIKE ‘%A’;

Пример (ROWID)

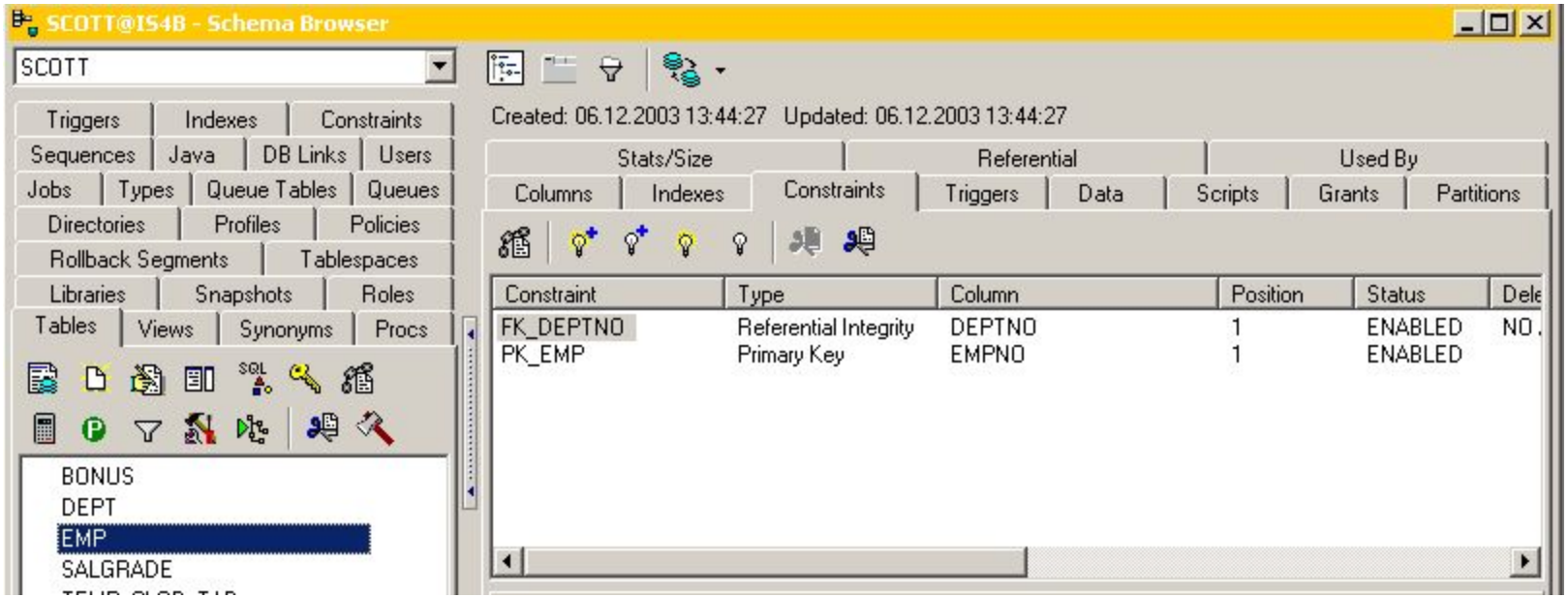
- SELECT
- /*+ROWID(emp)*/ *
- FROM emp
- WHERE rowid > 'AAAAtkAABAAAFNTAAA'
- AND empno = 155;

Пример (INDEX)

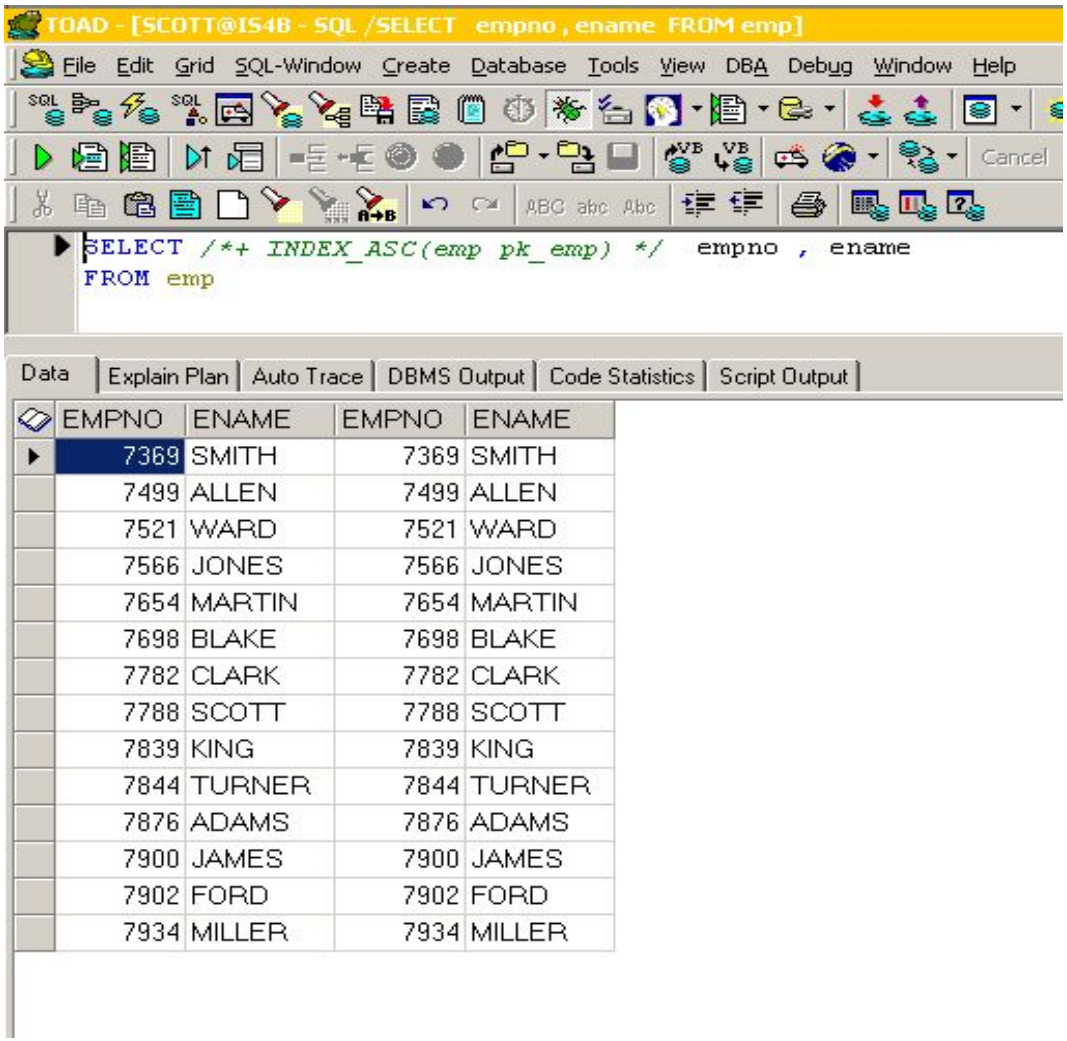
- SELECT /*+ INDEX (employees emp_department_ix)*/
  employee_id, department_id
- FROM employees

- WHERE department_id > 50;

Пример (INDEX_ASC)

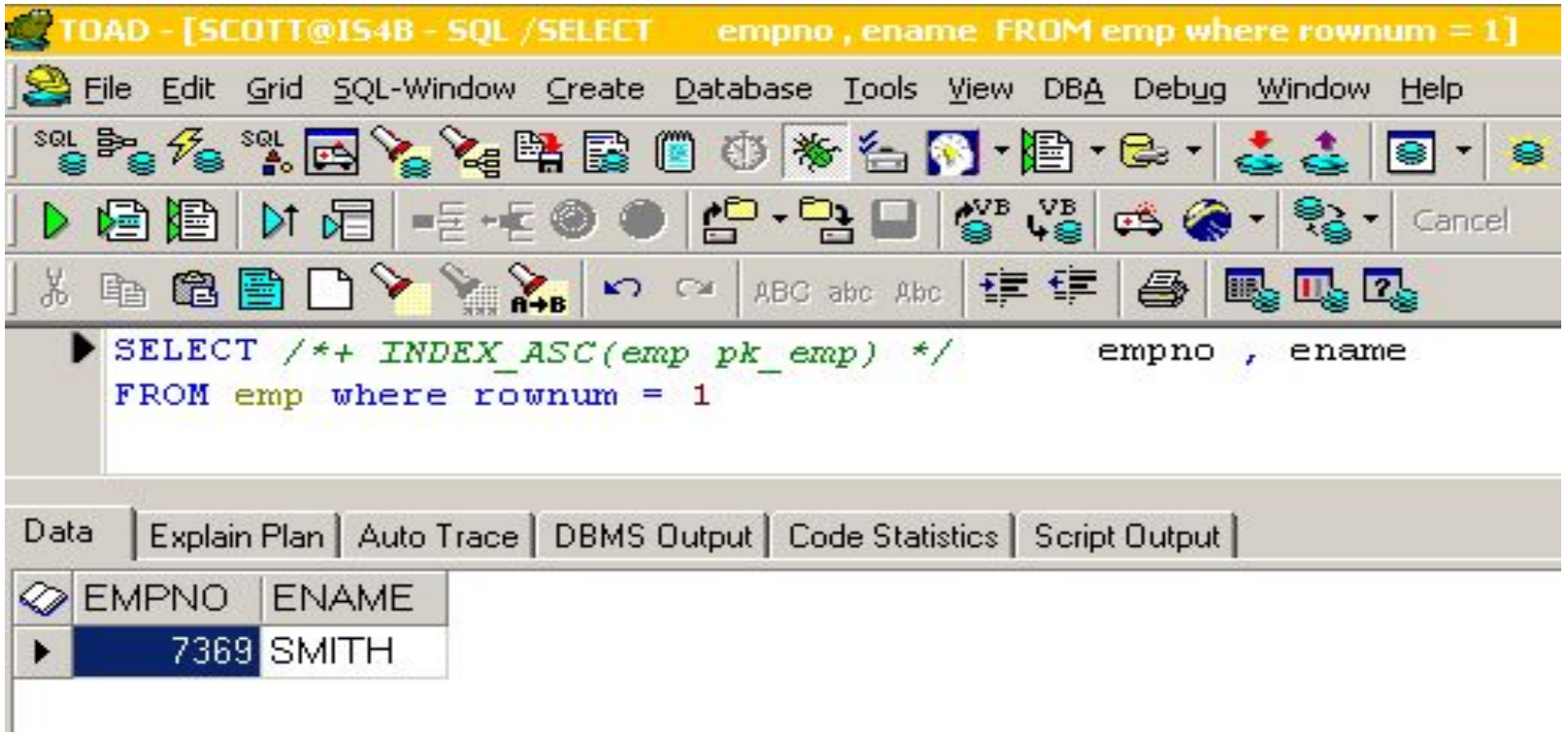
- 

Пример (INDEX_ASC)

- 

EMPNO	ENAME	EMPNO	ENAME
7369	SMITH	7369	SMITH
7499	ALLEN	7499	ALLEN
7521	WARD	7521	WARD
7566	JONES	7566	JONES
7654	MARTIN	7654	MARTIN
7698	BLAKE	7698	BLAKE
7782	CLARK	7782	CLARK
7788	SCOTT	7788	SCOTT
7839	KING	7839	KING
7844	TURNER	7844	TURNER
7876	ADAMS	7876	ADAMS
7900	JAMES	7900	JAMES
7902	FORD	7902	FORD
7934	MILLER	7934	MILLER

Пример (INDEX_ASC)

- 

Пример (INDEX_DESC)

- SELECT /*+ INDEX_DESC(emp pk_emp) */
- empno , ename
- FROM emp

- SELECT /*+ INDEX_DESC(emp pk_emp) */
- empno , ename
- FROM emp where rownum = 1

Пример (INDEX_FFS)

- SELECT
- /*+INDEX_FFS(emp emp_empno)*/ empno
- FROM emp
- WHERE empno > 200;

Пример (NO_INDEX)

- SELECT
- /*+NO_INDEX(emp emp_empno)*/
- empno
- FROM emp
- WHERE empno > 200;

Пример (INDEX_COMBINE)

- SELECT
- /*+ INDEX_COMBINE(e emp_manager_ix emp_department_ix) */ *
- FROM employees e
- WHERE manager_id = 108 OR department_id = 110;

- *Примечание: emp_manager_ix, emp_department_ix - bitmap индексы по полям manager_id и department_id. Оптимизатору рекомендовано построить логическое выражение (операция OR) из этих индексов.*

Пример (INDEX_JOIN)

- SELECT
- /*+ INDEX_JOIN(e emp_manager_ix emp_department_ix) */ department_id
- FROM employees e
- WHERE manager_id < 110 AND department_id < 50;

- *Оптимизатору рекомендовано построить логическое выражение (операция AND) из этих индексов.*

Подсказки для операции соединения (JOIN)

- **USE_NL** - использовать вложенные циклы для соединения указанных в подсказке таблиц;
- **USE_MERGE** – сначала выполнить сортировку, а затем ‘склеивание’ указанных таблиц;
- **USE_HASH** – HASH-соединение (сначала строится HASH-таблица, а затем ‘склеиваются’ фрагменты с одинаковыми HASH-значениями)
- ……..

Пример (USE_NL)

- SELECT
- /*+ USE_NL(customers) to get first row faster */
- accounts.balance,
- customers.last_name,
- customers.first_name
- FROM accounts, customers
- WHERE accounts.custno = customers.custno;

Пример (USE_MERGE)

- SELECT
- /*+USE_MERGE(emp dept)*/ *
- FROM emp, dept
- WHERE emp.deptno = dept.deptno;

Другие подсказки

- **MATERIALIZE** – материализовать промежуточную таблицу
- **PARALLEL** – распараллелить выполнение запроса
- И др.

Сбор статистики, полезной для оптимизатора

- Статистика по таблицам
  - Количество записей
  - Количество блоков
  - Средняя длина записи
- Статистика по колонкам
  - Количество различных значений в колонках
  - Количество null-значений в колонках
- Статистика по индексам
- Системная статистика

Процедуры для сбора статистики (пакет DBMS_STATS)

- GATHER_INDEX_STATS
- GATHER_TABLE_STATS
- GATHER_SCHEMA_STATS
- GATHER_DATABASE_STATS
- GATHER_SYSTEM_STATS

- *Примечание: эти процедуры не запускаются автоматически! Необходимо встраивать в приложения регулярный сбор статистики (или создавать отдельные приложения для администрирования).*

Представления словаря для просмотра статистики

- DBA_TABLES
- DBA_TAB_COL_STATISTICS
- DBA_INDEXES

Пример

```
SELECT TABLE_NAME,
       NUM_ROWS,
       BLOCKS,
       AVG_ROW_LEN,
      TO_CHAR(LAST_ANALYZED,
                'MM/DD/YYYY HH24:MI:SS')
FROM DBA_TABLES
WHERE
TABLE_NAME IN
('SO_LINES_ALL','SO_HEADERS_ALL','SO_LAST_ALL');
```

Домашнее задание 8(10 баллов)

Загрузите данные о потреблении электроэнергии (XML-файл electric power.xml прилагается к презентации). Проанализируйте среднестатистическое потребление электроэнергии по временам года и дням недели. Результат выдайте в виде таблицы и соответствующих графиков (для времен года). Примерный вид таблицы:

	воскресенье	понедельник	вторник	среда	четверг	пятница	суббота
зима							
весна							
лето							
осень							

Продемонстрируйте план исполнения запроса, соответствующий данным в таблице (запрос должен быть декларативным) .
Ссылку на приложение, логин и пароль для входа отправьте по адресу: N.Grafeeva@spbu.ru
Тема - DB_Application_2017_job8
Срок сдачи задания без потери баллов - 2 недели.