

Построение эффективных запросов

Графеева Н.Г.

2020

2 подхода к построению запросов

- Декларативный (SQL).
 - Императивный подход (процедурный стиль).
-
- Как создавать эффективные запросы? Какой подход выбрать в каждом конкретном случае?

Причины возникновения проблемы

- В связи с тем, что в современных СУБД в процедурных расширениях языка SQL появились возможности для встраивания в SQL-запросы результатов работы функций, выдающих в качестве результатов таблицы, возникает большой соблазн использовать эту возможность Однако, оказывается, что это не всегда уместно.
- Для начала научимся в принципе писать функции, выдающие множество строк...

Пример (табличная функция ORACLE)

```
CREATE TYPE t_tf_row AS OBJECT ( id NUMBER, description VARCHAR2(50) );  
/  
CREATE TYPE t_tf_tab IS TABLE OF t_tf_row;  
/  
-- Build the table function itself.  
CREATE OR REPLACE FUNCTION get_tab_tf (p_rows IN NUMBER)  
RETURN t_tf_tab  
AS  
l_tab t_tf_tab := t_tf_tab();  
BEGIN  
FOR i IN 1 .. p_rows LOOP  
    l_tab.extend;  
    l_tab(l_tab.last) := t_tf_row(i, 'Description for ' || i);  
END LOOP;  
RETURN l_tab;  
END;  
/  
-- Test it.  
SELECT * FROM TABLE(get_tab_tf(10)) ORDER BY id DESC;
```

Пример (pipe line функция ORACLE)

```
CREATE TYPE t_tf_row AS OBJECT ( id NUMBER, description VARCHAR2(50) );
/
CREATE TYPE t_tf_tab IS TABLE OF t_tf_row;
/
-- Build a pipelined table function.
CREATE OR REPLACE FUNCTION get_tab_ptf (p_rows IN NUMBER) RETURN t_tf_tab
PIPELINED
AS
BEGIN
    FOR i IN 1 .. p_rows LOOP
        PIPE ROW(t_tf_row(i, 'Description for ' || i));
    END LOOP;
    RETURN;
END;
/

-- Test it.
SELECT * FROM TABLE(get_tab_ptf(10)) ORDER BY id DESC;
```

Когда полезен процедурный стиль?

- Обработка данных существенно упрощается при использовании их упорядоченности.
- Оптимизатору запросов не удается построить план, в котором подзапросы исполняются однократно.

Типичные запросы, использующие упорядоченность

Типичные задачи – построение нарастающих итогов и разного рода индикаторов для финансовых рядов. Например,

- Нарастающие итоги:

$$\text{SUM}(1) = X(1)$$

$$\text{SUM}(t) = \text{SUM}(t-1) + X(t), \text{ при } t > 1$$

- Экспоненциальная скользящая средняя:

$$\text{EMA}(1) = X(1)$$

$$\text{EMA}(t) = X(t) * \alpha + (1-\alpha) * \text{EMA}(t-1), \text{ при } t > 1$$

Где α – коэффициент сглаживания

Пример (нарастающие итоги)

- Напишите запрос, выдающий нарастающие итоги (попробуйте в двух стилях) для таблицы с полями (PERIOD, VALUE).

PERIOD	VALUE
20/10/2020	17
21/10/2020	25
22/10/2020	12
23/10/2020	35
24/10/2020	28

SQL

```
SELECT PERIOD, VALUE,  
       (SELECT SUM(VALUE)  
        FROM VALS VVV  
        WHERE VVV.PERIOD <= VALS.PERIOD)  
FROM VALS  
ORDER BY PERIOD;
```

Процедурный стиль

```
CREATE TYPE acc_row AS OBJECT ( period DATE, value NUMBER,  
acc_total NUMBER );  
/  
CREATE TYPE acc_table IS TABLE OF acc_row;  
/  
-- Build a pipelined table function.  
CREATE OR REPLACE FUNCTION get_acc_table  
RETURN acc_table PIPELINED  
AS  
    total NUMBER;  
BEGIN  
    total := 0;  
    FOR REC IN (SELECT * FROM VALS ORDER BY PERIOD) LOOP  
        total := total + REC.VALUE;  
        PIPE ROW(acc_row(REC.PERIOD, REC.VALUE, total));  
    END LOOP;  
    RETURN;  
END;  
/  
-- Test it.  
SELECT * FROM TABLE(get_acc_table());
```

Классификация запросов

2 типа запросов

- **<Короткие запросы>** (OLTP –on-line transaction processing). Для выполнения таких запросов, как правило, обрабатывается лишь часть содержимого таблиц. Результат также невелик. Допустимая скорость исполнения – порядка 2-5 сек.
- **<Длинные запросы>** (OLAP – on-line analytical processing). Для получения результата необходимо обработать все или значительную часть строк таблиц. Допустимая скорость выполнения таких запросов на порядок выше (но этот порядок существенно зависит от профессиональных качеств разработчика).

Рекомендации для написания эффективных запросов

- Избегать многократных просмотров данных.
- Активнее использовать индексы для коротких запросов.
- Активнее использовать теоретико-множественные операции (произведение, объединение - UNION, минус - MINUS, пересечение - INTERSECTION) *
- Использовать операции группировки как можно раньше *.
- При необходимости выполнять операции соединения – выполнять их в правильной последовательности (минимизируя количество соединяемых записей) *.
- Использовать избыточные критерии селекции для сокращения количества выбираемых записей *.
- Примечание. * - особенно актуально для длинных запросов.

Что делать, если упомянутых рекомендаций все равно недостаточно?

- Использовать временные таблицы.
- В SQL запросах писать подсказки оптимизатору.

Временные таблицы

- Временные таблицы существуют во многих СУБД и предназначены для хранения данных на протяжении сеанса или транзакции.
- Отличительной особенностью этих таблиц является то, что структура этих таблиц определяется на уровне общей схемы базы, а данные в этих таблицах хранятся только на период сессии или транзакции (в зависимости от используемой при их определении опции) и располагаются во временных сегментах памяти.
- Данные из временных таблиц не видны в других клиентских сессиях.
- Они находят широкое применение в качестве промежуточных таблиц при сложных расчётах, в аналитических отчетах (особенно при промежуточных агрегированиях) и оптимизации сложных запросов.

Создание временных таблиц

Синтаксис:

```
CREATE GLOBAL TEMPORARY TABLE  
{ON COMMIT PRESERVE ROWS |  
ON COMMIT DELETE ROWS}
```

ON COMMIT PRESERVE ROWS - хранение
данных на время сеанса

ON COMMIT DELETE ROWS - хранение
данных на время транзакции

Ограничения для временных таблиц

- **Нельзя добавлять внешние ключи** на временную таблицу и ссылаться на нее как на родительскую.
- **Нельзя создавать индексы** и выполнять другие DDL операторы после того, как в таблице **уже появились данные**.
- Временная таблица **не может быть партиционирована** или организована как индексная таблица.
- **Нельзя распараллеливать** запросы к временным таблицам.
- **Распределенные транзакции** не могут работать с временными таблицами.

Возможности временных таблиц

- Временные таблицы могут использовать правила целостности (за исключением ссылочных).
- Временные таблицы могут сопровождаться индексами.
- Примечание: и те и другие могут добавляться только тогда, когда в таблице нет записей **ни в одной сессии или транзакции!!!**

Пример (создание временной таблицы)

```
CREATE GLOBAL TEMPORARY TABLE CITY_DEPT
(
    DEPTNO NUMBER(2,0),
    DNAME VARCHAR2(14),
    CONSTRAINT PK_CITY_DEPT PRIMARY KEY (DEPTNO)
)
ON COMMIT DELETE ROWS;

=====
COMMENT ON COLUMN CITY_DEPT.DEPTNO IS 'DEPARTMENT NUMBER';
COMMENT ON COLUMN CITY_DEPT.DNAME IS 'DEPARTMENT NAME';
=====
CREATE UNIQUE INDEX IDX_DEPTNO_DNAME ON CITY_DEPT (DEPTNO,DNAME) ;
CREATE INDEX IDX_DNAME ON CITY_DEPT (DNAME)
=====
```

Сбор и использование статистики

- Существует два вида статистики применительно к временным таблицам:
- **SESSION** - уровня клиентской сессии
- **SHARED** - разделяемая между клиентскими сессиями

SESSION и SHARED-статистики

- SESSION-статистика собирается и используется только во время текущей клиентской сессии.
- Если одновременно существует два вида статистики (SESSION и SHARED), то оптимизатор отдаст предпочтение SESSION-статистике.
- SESSION-статистика удаляется как только заканчивается сессия.
- SHARED-статистика сохраняется после завершения сессии.

Какой параметр отвечает за
выбранный тип статистики?

параметр - **GLOBAL_TEMP_TABLE_STATS**

Как узнать его значение:

```
SELECT DBMS_STATS.get_prefs('GLOBAL_TEMP_TABLE_STATS') FROM dual;
```

Упражнение

- Уточните в ORACLE APEX тип установленной статистики для временных таблиц.

Как изменить тип статистики?

```
BEGIN
  DBMS_STATS.set_global_prefs
    ( pname => 'GLOBAL_TEMP_TABLE_STATS',
      pvalue => 'SHARED');
```

```
END;
```

```
/
```

```
BEGIN
  DBMS_STATS.set_global_prefs
    ( pname => 'GLOBAL_TEMP_TABLE_STATS',
      pvalue => 'SESSION' );
```

```
END;
```

```
/
```

Примечание: выполнение этих операций возможно только при наличии соответствующих привилегий!!

Как собрать статистику?

```
DBMS_STATS.gather_table_stats ('<schema>',  
                                '<temporary-table>');
```

- *Примечание: вызов процедуры `gather_table_stats` доступен простым пользователям APEX!!!*

Пример сбора статистики

```
dbms_stats.gather_table_stats('GRAFEEVA','TABLE1');
```

Где можно посмотреть собранную статистику?

- DBA_TAB_STATISTICS
- DBA_IND_STATISTICS
- DBA_TAB_HISTOGRAMS
- DBA_TAB_COL_STATISTICS

*Смотреть можно при наличии
достаточных административных
привилегий...*

Как выглядит весь цикл использования временных таблиц в приложения?

- Определяем временную таблицу на уровне схемы базы;
- Определяем подпрограмму (процедуру или функцию), собирающую данные во временную таблицу;
- Вызываем подпрограмму.

Создаем временную таблицу

```
CREATE GLOBAL TEMPORARY TABLE  
table1 ( id NUMBER,  
NAME VARCHAR2(100))  
ON COMMIT PRESERVE ROWS;
```

Как выглядит подпрограмма, использующая временную таблицу?

BEGIN

- чистим временную таблицу;
- заполняем временную таблицу данными (как правило, агрегированными);
- собираем или не собираем статистику (SESSION /SHARED);
- выбираем данные из временной таблицы;

END

- *Примечание: при этом в подпрограмме, собирающей данные, должна быть объявлена автономная транзакция!!!*

Пример функции с временной таблицей

1. Определим вспомогательные типы данных для создания функции.

```
CREATE TYPE t_tf_row AS OBJECT ( id NUMBER,  
    description VARCHAR2(50) );
```

```
/
```

```
CREATE TYPE t_tf_tab IS TABLE OF t_tf_row;
```

```
/
```

Пример функции с временной таблицей

- 2. Определим саму функцию.

```
create or replace function get_tab_ptf(p_rows in number) return t_tf_tab pipelined
is PRAGMA AUTONOMOUS_TRANSACTION;
begin
    execute immediate('truncate table table1'); -- чистим временную таблицу
    for i in 1..p_rows loop -- размещаем данные в таблице
        insert into table1(id,name) values(i, 'Description for ' || i);
    end loop;
    dbms_stats.gather_table_stats('GRAFEEVA','TABLE1'); -- собираем статистику
    for rec in (select * from table1) loop -- формируем результат
        pipe row(t_tf_row(rec.id, rec.name));
    end loop;
    return;
end;
/
```

Пример: вызов функции, использующей временную таблицу

- `select * from table(get_tab_ptf(10))`

Зачетное задание 8 (8 баллов)

- Экспортируйте 5-мин котировки индекса RTS по следующей ссылке:
- http://www.finam.ru/profile/mosbirzha-fyuchersy/rts-12-16-riz6/export/?market=14&em=407238&code=RIZ6&apply=0&df=7&mf=9&yf=2016&from=07.10.2016&dt=7&mt=9&yt=2016&to=07.10.2016&p=3&f=RIZ6_161007_161007&e=.txt&cn=RIZ6&dtf=1&tmf=1&MSOR=1&mstime=on&mstimever=1&sep=1&sep2=1&datf=1&at=1
- за 7 октября 2016 года и постройте приложение, в котором будут отображаться цена индекса RTS и экспоненциальное скользящее среднее цены в виде графика (на одном графике две линии и легенда, в качестве цены можно использовать $(OPEN + HIGH + LOW + CLOSE)/4$)

Обзор

Новости

Комментарии

Календарь событий

Календарь
статистики


Теханализ Live! new

Теханализ Live!

Теханализ Лайт


Сравнение с...

Экспорт котировок

Обсудить в
форуме 

МосБиржа фьючерсы RTS-12.16(RIZ6)

Интервал и периодичность

07.10.2016 

— 07.10.2016 

5 мин. 

Имя выходного файла

SPFB.RTS-12.16_161007_161007

.txt 

Имя контракта

SPFB.RTS-12.16

Формат

даты

ггггммдд 

времени

ччммсс 

Выдавать время



начала свечи



окончания свечи



московское

Разделитель

полей

запятая (,) 

разрядов

нет 

Формат записи в файл

TICKER, PER, DATE, TIME, OPEN, HIGH, LOW, CLOSE, VOL 

Добавить заголовок файла



Заполнять периоды без сделок



Получить файл

Зачетное задание 9(8 баллов)

На основе данных из файла о потреблении электроэнергии (electric power.xml) создайте приложение с аналитическим запросом о суммарном потреблении электроэнергии за указанные периоды (с использованием временных таблиц) :

	Утро [6-12)	День[12-18)	Вечер[18-24)	Ночь [0-6)	Итого
1 квартал					
2 квартал					
3 квартал					
4 квартал					
Итого за 2009 год					
1 квартал					
2 квартал					
3 квартал					
4 квартал					
Итого за 2010 год					
Итого					