

Построение запросов

Графеева Н.Г.

2017

2 подхода к построению запросов

- Императивный подход (процедурный стиль)
- Декларативный (SQL)
- 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, 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;`

Упражнение 1

- Напишите в разных стилях запрос (к демонстрационной базе ORACLE), в котором будет отображено сколько наименований следующих видов продукции:
 - Bag, Blouse, Jacket
 - было отправлено в каждый штат.

Когда полезен процедурный стиль?

- Обработка данных существенно упрощается при использовании их упорядоченности.
- Оптимизатору запросов не удается построить план, в котором подзапросы исполняются однократно. В этом случае предварительное выполнение подзапросов может существенно ускорить результат.

Типичные запросы, использующие упорядоченность

- Типичные задачи – построение нарастающих итогов и разного рода индикаторов для финансовых рядов. Например,
 - Нарастающие итоги:
 - $SUM(1) = X(1)$
 - $SUM(t) = SUM(t-1) + X(t)$, при $t > 1$
 - Экспоненциальная скользящая средняя:
 - $EMA(1) = X(1)$
 - $EMA(t) = EMA(t-1) + (X(t) - EMA(t-1)) * 2/N$, при $t > 1$
 - Где N – количество элементов ряда, используемых для сглаживания

Упражнение 2

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

Правила гранулярности запросов

- Количество операторов SQL, не должно зависеть от количества обрабатываемых строк (в рамках одного запроса).
- Идентичные операторы SELECT не должны исполняться многократно.
- Если некоторая строка таблицы передается в приложение, то в этом же запросе следует передать все поля таблицы, которые могут понадобиться приложению.
- Все условия на выбираемые строки таблиц следует включать в операторы SQL.

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 порядки выше.

Рекомендации для написания эффективных запросов

- 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 *.
- Избегать многократных просмотров данных.
- Примечание. * - особенно актуально для длинных запросов.

Домашнее задание 8(10 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
- (в качестве цены можно использовать $(OPEN + HIGH + LOW + CLOSE)/4$)
- Ссылку на приложение, логин и пароль для входа отправьте по адресу: N.Grafeeva@spbu.ru
- Тема - DB_Application_2017_job8

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