Временные таблицы Oracle

Графеева Н.Г.

2017

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 они располагаются во временных сегментах и данные в этих таблицах хранятся только на период сессии или транзакции в зависимости от используемой при их определении опции.
- Они находят широкое применение в качестве промежуточных таблиц при расчётах, отчетах (особенно при промежуточных агрегированиях) и оптимизации сложных запросов.

Создание временных таблиц

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

Упражнение 1

В ORACLE APEX выполните следующие команды и объясните результаты:

==========

```
CREATE GLOBAL TEMPORARY TABLE table1 (id NUMBER(5),name VARCHAR2(20))
ON COMMIT PRESERVE ROWS;
```

==========

```
INSERT INTO table1 (id,name) VALUES(1,'items1');
```

==========

```
SELECT * FROM table1
```

==========

```
DECLARE
   VAR NUMBER;
BEGIN
  INSERT INTO table1 (id,name) VALUES(1,'items1');
  SELECT COUNT(*) INTO VAR FROM table1;
  DBMS_OUTPUT.PUT_LINE('VAR=' || VAR);
END;
```

==========

```
DECLARE
   VAR NUMBER;
BEGIN
  INSERT INTO table1 (id,name) VALUES(1,'items1');
  COMMIT;
  SELECT COUNT(*) INTO VAR FROM table1;
  DBMS_OUTPUT.PUT_LINE('VAR=' || VAR);
END;
```

==========

Упражнение 2

В ORACLE APEX выполните следующие команды и объясните результаты:

==========

```
CREATE GLOBAL TEMPORARY TABLE table2 (id NUMBER(5),name VARCHAR2(20))
ON COMMIT DELETE ROWS;
```

==========

```
INSERT INTO table2 (id,name) VALUES(1,'items1');
```

==========

```
SELECT * FROM table2
```

==========

```
DECLARE
   VAR NUMBER;
BEGIN
  INSERT INTO table2 (id,name) VALUES(1,'items1');
  SELECT COUNT(*) INTO VAR FROM table2;
  DBMS_OUTPUT.PUT_LINE('VAR=' || VAR);
END;
```

==========

```
DECLARE
   VAR NUMBER;
BEGIN
  INSERT INTO table2 (id,name) VALUES(1,'items1');
  COMMIT;
  SELECT COUNT(*) INTO VAR FROM table2;
  DBMS_OUTPUT.PUT_LINE('VAR=' || VAR);
END;
```

==========

Комментарии…

- Данные из таблицы **table2** удалились сразу после завершения транзакции (опция **on commit delete rows**).
- Отличительной особенностью временных таблицы является то, что данные таблиц не только удаляются, но и не видны из других сеансов.
- Пользователи могут одновременно использовать одну и туже временную таблицу, не пересекаясь данными.

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

```
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)
=============================================
```

Использование статистики при выполнении запросов к временным таблицам

- 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!!!*

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 процедурах и функциях?

BEGIN

чистим временную таблицу;

заполняем временную таблицу данными (как правило, агрегированными);

собираем или не сбираем статистику (SESSION /SHARED);

выбираем данные из временной таблицы;

END

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

Пример:создание вспомогательных типов

```
CREATE TYPE t_tf_row AS OBJECT ( id NUMBER, description VARCHAR2(50) );
 /
CREATE TYPE t_tf_tab IS TABLE OF t_tf_row;
/
```

Пример: создание функции, использующей временную таблицу

```
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

- select * from table(get_tab_ptf(10))

Упражнение

Создайте функцию, которая выдает результат следующего вида на основе таблицы EMP (используйте временные таблицы):

	1980	1981	1982	1983	Итого
CLERK					
SALEMAN					
MANAGER					
PRESIDENT					
Итого					

Домашнее задание 9(10 баллов)

- На основе данных из задания об электроэнергии создайте приложение с аналитическим отчетом о суммарном потреблении электроэнергии за указанные периоды :

	0-1	1-2	2-3	3-4	…	23 - 0	Итого
1 квартал							
2 квартал							
3 квартал							
4 квартал							
Итого за 2008 год							
1 квартал							
2 квартал							
3 квартал							
4 квартал							
Итого за 2009 год							
Итого за 2008-2009							

Для формирования промежуточных результатов используйте временные таблицы.

Результат отправьте по адресу N.Grafeeva@spbu.ru. Тема письма – DB_Application_2017_job9.

Примечание:задание должно быть отправлено в течение 2 недель. За более позднее отправление будут сниматься штрафные баллы (по баллу за каждые 2 недели).