ТРИГГЕРЫ

Графеева Н.Г.

2020

Определение

Триггер - это блок PL/SQL, выполняемый неявно каждый раз, когда происходит конкретное событие.

Типы событий запускающих триггеры:

- 1. DML-события, которые происходят при выполнении инструкций INSERT, UPDATE или DELETE.
- 2. DDL-события, которые происходят при выполнении инструкций CREATE, ALTER или DROP.
- 3. Другие события уровня базы данных.

Триггеры носят глобальный характер и не зависят от того, кто и каким образом вызвал появление событий, на которые они срабатывают.

Назначение триггеров

- Поддержка бизнес-логики данных.
- Аудит действий пользователей в базе.

Создание, редактирование и удаление триггеров (SQL DDL)

- CREATE TRIGGER ….
- ALTER TRIGGER…
- DROP TRIGGER…

Синтаксис (для событий типа 1 - DML)

CREATE [OR REPLACE] TRIGGER *имя_триггера*

{BEFORE | AFTER }

событие_A[OR событие_B[OR событие_C]]

ON {таблица_или_представление |

[FOR EACH ROW]

[WHEN условие_триггера]

тело_триггера;

END *имя_триггера*;

Синтаксис (для событий типа 2 и 3 - DDL и другие)

CREATE [OR REPLACE] TRIGGER *имя_триггера*

{BEFORE | AFTER} событие_триггера

ON {DATABASE | SCHEMA}

[WHEN условие_триггера]

    тело_триггера;

END *имя_триггера*;

Пример (событие типа DML)

```
CREATE  TRIGGER tr_emp_diu
BEFORE
    DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
WHEN (new.empno > 0)
DECLARE
  /*   переменные,константы,курсоры */
BEGIN
 /*  блок PL/SQL */
END;
```

Пример (событие типа 2 - DDL)

```
CREATE TRIGGER "BI_CREATE_OBJECT"
BEFORE CREATE ON SCHEMA
BEGIN
    LogInfo('create db-object');
END;
```

Имена триггеров

- Триггер должен иметь уникальное имя среди триггеров
- Однако его имя может совпадать с именем другого объекта базы (например, таблицы)

События триггера

- Тип 1: INSERT, UPDATE, DELETE
- Тип 2: CREATE , ALTER , DROP (любых объектов базы)
- Тип 3: STURTUP, SHUTDOWN , ERROR MESSAGES, LOGON,LOGOFF …

SQL-операторы, инициирующие исполнение триггеров (DML события)

- DELETE FROM emp;
- INSERT INTO emp VALUES (...);
- INSERT INTO emp SELECT . . . FROM . . .;
- UPDATE emp SET ....;

SQL-операторы, инициирующие исполнение триггеров (DDL события)

- CREATE TABLE….
- CREATE INDEX…
- ALTER TABLE...
- DROP INDEX…
- CREATE PROCEDURE…

Опции триггеров

- Row Triggers, Statement Triggers
- BEFORE Triggers, AFTER Triggers
- System Events, User Events

Пример(statement trigger)

```
SQL> CREATE OR REPLACE TRIGGER print_trigger
  2  BEFORE DELETE OR INSERT OR UPDATE ON emp
  3  BEGIN
  4      dbms_output.put_line('  Trigger works ');
  5  END;
  6  /

Trigger created.
```

SQL>…….

SQL> UPDATE emp SET sal = sal + 100;

Trigger works

16 rows updated.

Пример (row trigger)

```
SQL> CREATE OR REPLACE TRIGGER print_trigger
  2  BEFORE
        DELETE OR INSERT OR UPDATE ON emp
  3  FOR EACH ROW
  4  BEGIN
  5  dbms_output.put_line('  Trigger works ');
  6  END;
  7  /

Trigger created.
```

```
SQL>  UPDATE emp SET sal = sal + 100;
Trigger works
Trigger works
Trigger works
. . .
. . .
Trigger works
Trigger works
Trigger works
Trigger works

16 rows updated.
```

Опция WHEN

Обеспечивает доступ к значениям столбцов (старым и новым) через переменные new и old.

Позволяет задавать условие (для работы триггера) в виде логического выражения SQL (не PL/SQL!!!). При написании выражения можно использовать:

*+ - * / arithmetic operators*

|| character operators

All comparison operators

NOT logical operator

AND logical operator

OR logical operator

Старые и новые значения

Доступны через переменные new и old;

В триггере для INSERT имеют смысл только новые значения;

Триггер для UPDATE независимо от того, выполняется он “до” или “после”, может обращаться как к старым, так и к новым значениям;

В триггере для DELETE имеют смысл только старые значения;

Не могут применяться к столбцам типа LONG и LONG RAW.

```
SQL> CREATE OR REPLACE TRIGGER print_trigger
  2  BEFORE DELETE OR INSERT  OR UPDATE ON emp
  3  FOR EACH ROW
  4  WHEN (OLD.JOB = 'CLERK' OR  OLD.JOB = 'MANAGER')
  5  BEGIN
  6       dbms_output.put_line('  Trigger works ');
  7  END;
  8  /

Trigger created.
```

Пример

```
SQL> CREATE OR REPLACE TRIGGER print_trigger
  2  BEFORE DELETE OR INSERT  OR UPDATE ON emp
  3  FOR EACH ROW
  4  WHEN (OLD.JOB = 'CLERK' OR  OLD.JOB = 'MANAGER')
  5  BEGIN
  6       dbms_output.put_line('  Trigger works ');
  7  END;
  8  /

Trigger created.
```

Проверим исходные данные:

```
SQL> SELECT * FROM EMP;

     EMPNO ENAME      JOB
---------- ---------- --------- -----
      7369 SMITH      CLERK
      7499 ALLEN      SALESMAN
      7521 WARD       SALESMAN
      7566 JONES      MANAGER
      7654 MARTIN     SALESMAN
      7698 BLAKE      MANAGER
      7782 CLARK      MANAGER
      7788 SCOTT      ANALYST
      . . . . . .
16 rows selected.
```

- Выполним операцию UPDATE:

- SQL> UPDATE emp SET sal = sal + 100;
- Trigger works
- Trigger works
- Trigger works
- Trigger works
- Trigger works
- Trigger works
- Trigger works

- 16 rows updated.

- Триггер вызывался7 раз!

Выполним операцию INSERT:

```
SQL> INSERT INTO EMP (EMPNO,ENAME,JOB)
    VALUES(700,'TOM','MANAGER');

1 row created.
```

Триггер не работал.

Выполним операцию DELETE:

SQL> DELETE FROM emp WHERE empno = 700;
Trigger works

1 row deleted.

Триггер вызывался один раз.

Тело триггера

- Представляет собой блок PL/SQL.
- Обеспечивает доступ к старым и новым значениям столбцов через переменные :NEW и :OLD.
- Может содержать предикаты INSERTING, DELETING, UPDATING (для событий типа DML) и, так называемые, **атрибутные** функции.

Атрибутные функции

Имя функции	Тип значения	Событие	Описание
SYSEVENT	VARCHAR2(20)	Все события	Возвращает системное событие активизировавшее триггер
INSTANCE_NUM	NUMBER	Все события	Возвращает номер текущего экземпляра
DATABASE_NAME	VARCHAR2(50)	Все события	Возвращает имя текущей БАЗЫ ДАННЫХ.
SERVER_ERROR	NUMBER	SERVERERROR	Возвращает ошибку из стека ошибок,
LOGIN_USER	VARCHAR2(20)	Все события	Возвращает идентификатор пользователя активизирующего триггер.
DICTIONARY_OBJ_TYPE	VARCHAR2(30)	CREATE, DROP, ALTER	Возвращает тип объекта словаря, над которым выполнялась операция DDL
DICTIONARY_OBJ_NAME	VARCHAR2(30)	CREATE, DROP, ALTER	Возвращает имя объекта словаря, над которым выполнялась операция DDL
DICTIONARY_OBJ_OWNER	VARCHAR2(30)	CREATE, DROP, ALTER	Возвращает владельца того объекта словаря, над которым выполнялась операция DDL
DES_ENCRYPTED_PASSWORD	VARCHAR2(30)	CREATE USER, ALTER USER	Возвращает зашифрованный в стандарте DES пароль создаваемого или изменяемого пользователя.

Пример

```
SQL>  CREATE OR REPLACE TRIGGER print_salary_changes
  2       BEFORE DELETE OR INSERT OR UPDATE ON emp
  3       FOR EACH ROW
  4       WHEN ( new.empno > 20)
  5       DECLARE
  6           sal_diff number;
  7       BEGIN
  8           sal_diff := :new.sal - :old.sal;
  9           dbms_output.put('Old salary: ' || :old.sal);
 10           dbms_output.put('  New salary: ' || :new.sal);
 11           dbms_output.put_line('  Difference ' || sal_diff);
 12       END;
 13     /

Trigger created.
```

```
SQL> UPDATE emp SET sal = sal + 100;
Old salary: 1100  New salary: 1200  Difference 100
Old salary: 1900  New salary: 2000  Difference 100
Old salary: 1550  New salary: 1650  Difference 100
Old salary: 3275  New salary: 3375  Difference 100
Old salary: 1550  New salary: 1650  Difference 100
Old salary: 3150  New salary: 3250  Difference 100
Old salary: 2750  New salary: 2850  Difference 100
Old salary: 3300  New salary: 3400  Difference 100
Old salary: 5300  New salary: 5400  Difference 100
Old salary: 1800  New salary: 1900  Difference 100
Old salary: 1400  New salary: 1500  Difference 100
Old salary: 1250  New salary: 1350  Difference 100
Old salary: 3300  New salary: 3400  Difference 100
Old salary: 1600  New salary: 1700  Difference 100
```

Предикаты INSERTING, DELETING, UPDATING

В заголовке триггера:

...INSERT OR UPDATE OR DELETE ON emp

В теле триггера:

IF INSERTING THEN .... END IF;

IF UPDATING THEN .... END IF;

IF DELETING   THEN .... END IF;

Ограничения

- В теле триггера могут встречаться команды DML, но не DDL.
- Не могут встречаться команды управления транзакцией.
- Ограничения на каскадные триггеры до 32.

Включение\выключение триггеров

- ALTER TRIGGER <trigger-name> [ENABLE/DISABLE]
- ALTER TABLE <table-name> [ENABLE/DISABLE] ALL TRIGGERS

Когда нужно выключать триггер?

- Когда объекты, на которые триггер ссылается недоступны
- При загрузке данных большого объема (если при этом триггер не генерирует жизненно необходимые значения)

Пример

Выключение триггеров:

ALTER TRIGGER reorder DISABLE;
ALTER TABLE inventory DISABLE ALL TRIGGERS;

Включение триггеров:

ALTER TRIGGER reorder ENABLE;
ALTER TABLE inventory ENABLE ALL TRIGGERS;

Системные представления

- USER_TRIGGERS
- ALL_TIGGERS
- DBA_TRIGGERS

Задание 1

Создайте триггер, обеспечивающий автоматическую генерацию значений в одной из таблиц своей базы (для получения очередного номера используйте секвенцию).

Задание 2

Создайте триггер, обеспечивающий автоматическую генерацию значений в одной из таблиц своей базы (без использования секвенции).

Задание 3

Создайте триггер, который будет записывать в журнал события, связанные с созданием, изменением и удалением таблиц , представлений и секвенций (какое событие, имя объекта, когда и т.п.).

Задание 4

Продемонстрируйте созданные триггеры через соответствующие системные представления.

А еще триггеры могут применяться для:

- обеспечения сложного протоколирования
- обеспечения ссылочной целостности (если этого нельзя сделать средствами правил целостности)
- задания сложных правил целостности
- обеспечения контроля над некоторыми событиями
- синхронной репликация таблиц
- …..

Пример (протоколирование с помощью триггеров)

```
SQL> CREATE TRIGGER audit_employee
  2  AFTER INSERT OR DELETE OR UPDATE ON emp
  3  FOR EACH ROW
  4  BEGIN
  5  INSERT INTO audit_employee VALUES
  6     (:old.ename, :old.job, :old.sal, :old.deptno,
  7      :new.ename, :new.job, :new.sal, :new.deptno,
  8     user, sysdate );
  9  END;
 10  /

Trigger created.
```

Пример (нестандартные правила целостности и триггеры)

```
SQL> CREATE TRIGGER dept_set
  2  AFTER DELETE OR UPDATE OF deptno ON dept
  3  FOR EACH ROW
  4  BEGIN
  5     IF UPDATING AND :OLD.deptno != :NEW.deptno  THEN
  6        UPDATE emp SET emp.deptno = :new.deptno
  7           WHERE emp.deptno = :old.deptno;
  8     ELSE  IF DELETING THEN
             UPDATE emp SET emp.deptno = NULL
  9           WHERE emp.deptno = :old.deptno;
 10     END IF; END IF;
 11  END;
 12  /

Trigger created.
```

Пример (нестандартные правила целостности и триггеры)

```
SQL> CREATE or REPLACE TRIGGER salary_check
  2  BEFORE INSERT OR UPDATE OF sal, job ON emp
  3  FOR EACH ROW
  4  DECLARE
  5     minsal              NUMBER;
  6     maxsal              NUMBER;
  7     salary_out_of_range   EXCEPTION;
  8  BEGIN
  9  SELECT MIN(losal),MAX(hisal) INTO minsal, maxsal FROM salgrade;
 10     IF (:new.sal < minsal OR :new.sal > maxsal) THEN
 11        RAISE salary_out_of_range;
 12     END IF;
 13  EXCEPTION
 14     WHEN salary_out_of_range THEN
 15     raise_application_error (-20300,'Salary '||TO_CHAR(:new.sal)||
         ' out of range for ’  ||' for employee '||:new.ename);
 16  END;
```

```
SQL>  UPDATE emp SET sal = 100 WHERE empno = 7900;
UPDATE emp SET sal = 100 WHERE empno = 7900
 *
ERROR at line 1:
ORA-20300: Salary 100 out of range for  for employee JAMES
ORA-06512: at line 12
ORA-04088: error during execution of trigger 'SCOTT.SALARY_CHECK'
```

Пример (обеспечение контроля над событиями)

```
SQL> CREATE OR REPLACE TRIGGER emp_permit_changes
  2  BEFORE INSERT OR DELETE OR UPDATE ON emp
  3  DECLARE
  4    not_on_weekends EXCEPTION;
  5    non_working_hours EXCEPTION;
  6  BEGIN
  7    IF (TO_CHAR(sysdate, 'DY') = 'SAT' OR
  8      TO_CHAR(sysdate, 'DY') = 'SUN') THEN
  9      RAISE not_on_weekends;
 10    END IF;
 11    IF (TO_CHAR(sysdate, 'HH24') < 8 OR
 12      TO_CHAR(sysdate, 'HH24') > 18) THEN
 13      RAISE non_working_hours;
 14    END IF;
 15  EXCEPTION
 16    WHEN not_on_weekends THEN
 17      raise_application_error(-20324,'May not change '
 18        ||'employee table during the weekend');
 19    WHEN non_working_hours THEN
 20      raise_application_error(-20326,'May not change '
 21      ||'emp table during non-working hours');
 22  END;
 23  /
```

- Trigger created.

```
SQL> DELETE FROM emp WHERE empno = 1001;
DELETE FROM emp WHERE empno = 1001
 *
ERROR at line 1:
ORA-20326: May not change emp table during non-working hours
ORA-06512: at line 18
ORA-04088: error during execution of trigger
    'SCOTT.EMP_PERMIT_CHANGES'
```

Домашнее задание 4 (8 баллов)

Создайте переиспользуемый скрипт (т.е. скрипт, который можно запускать повторно) в котором кроме удаления и создания пары таблиц будет предусмотрено заполнение таблиц начальными данными (с использованием секвенций и триггеров). Предусмотрите создание журнала и процедуры, обеспечивающей запись в журнал. Кроме того, должен быть предусмотрен триггер, который фиксирует в журнале типы и имена всех создаваемых в базе объектов.

Результат (листинг работавшего скрипта) разместите на GOOGLE DISK, а ссылку на него отправьте по адресу N.Grafeeva@spbu.ru. Тема письма – DB_Application_2020_job4.

Примечание: задание должно быть отправлено в течение 14 дней. За более позднее отправление будут сниматься штрафные баллы (по баллу за каждые две недели).