

Санкт-Петербургский государственный политехнический
университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

Дисциплина: Базы данных

Тема: SQL-программирование: Триггеры, вызовы процедур

Выполнил
студент гр. 43501/3

_____ Д. А. Зобков
(подпись)

Преподаватель

_____ А. В. Мяснов
(подпись)

“__” _____ 2016 г.

Санкт-Петербург
2016 г.

1 Цель работы

Познакомиться с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур и триггеров.

2 Программа работы

1. Создать два триггера: один триггер для автоматического заполнения ключевого поля, второй триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице;
2. Создать триггер в соответствии с индивидуальным заданием, полученным у преподавателя;
3. Создать триггер в соответствии с индивидуальным заданием, вызывающий хранимую процедуру;
4. Выложить скрипт с созданными сущностями в системе контроля версий;
5. Продемонстрировать результаты преподавателю.

3 Индивидуальные задания

1. Реализовать триггеры, не позволяющие несколько раз добавить трек в плейлист и альбом.
2. При добавлении новых данных в таблицу продаж альбомов запускать пересчет агрегатов по окончившемуся периоду: если последняя запись в таблицу продаж была добавлена до окончания очередного агрегационного этапа, а добавляемая — после, то запускать процедуру пересчета для окончившегося периода.

4 Ход работы

4.1 Триггер для автоматического заполнения ключевого поля

Реализовать триггер автоинкремента первичного ключа можно различными методами, приведем два варианта:

1. С использованием генератора:

```
1 CREATE SEQUENCE SEL_GEN;  
2 ALTER SEQUENCE SEL_GEN RESTART WITH 28; -- Установка значения гене  
    ратора  
    -- Завершение транзакции  
4 COMMIT;
```

Создадим триггер для таблицы SELLING:

```
1 CREATE OR ALTER TRIGGER SEL_ID_AUTOINC FOR SELLING  
2 ACTIVE BEFORE INSERT POSITION 0  
AS  
4 BEGIN
```

```

        IF (NEW.SEL_ID IS NULL OR NEW.SEL_ID = 0) THEN NEW.SEL_ID =
        NEXT VALUE FOR SEL_GEN;
6 END
   -- Завершение транзакции
8 COMMIT;

```

Следует отметить, что в данной реализации генератор будет увеличиваться даже в том случае, когда добавление записей было отменено командой ROLLBACK. В случае, если создаваемый ID уже существует в таблице, триггер создаёт новый до тех пор, пока не достигнет уникального.

2. С нахождением максимального значения первичного ключа

Создадим триггер для таблицы ARTISTS:

```

1 CREATE OR ALTER TRIGGER art_id_autoinc FOR ARTISTS
2 ACTIVE BEFORE INSERT POSITION 0
  AS
4 DECLARE VARIABLE NEW_ID INTEGER;
  BEGIN
6     SELECT MAX(ART_ID) FROM ARTISTS INTO :NEW_ID;
      NEW.ART_ID = :NEW_ID + 1;
8 END
   -- Завершение транзакции
10 COMMIT;

```

Преимуществом данного варианта является однозначное значение первичного ключа (максимальное значение в таблице + 1).

4.2 Триггер контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице

Так как все связи между таблицами в разработанной схеме БД определены через связи PK⇐FK, а действие на обновление ограничений (CONSTRAINTS) не было определено, то изменить первичный ключ в главной таблице, пока на нее ссылаются подчиненные, невозможно. Поэтому триггер учитывает только удаление записей в таблице AWARDS:

```

1 CREATE OR ALTER TRIGGER AWARD_CTL FOR AWARDS
2 ACTIVE BEFORE DELETE POSITION 0
  AS
4 BEGIN
      DELETE FROM ALB_AWARDS WHERE ALB_AWARDS.AWARD_ID = OLD.AWARD_ID;
6      DELETE FROM TRACK_AWARDS WHERE TRACK_AWARDS.AWARD_ID = OLD.AWARD_ID
      ;
  END
8 -- Завершение транзакции
  COMMIT;

```

4.3 Реализовать триггеры, не позволяющие несколько раз добавить трек в плейлист и альбом

Для создания данных триггеров предварительно создадим исключение:

```
1 CREATE OR ALTER EXCEPTION ISCOPY 'This entry is already exist!';
```

Проверка дублирования для плейлистов:

```
1 CREATE OR ALTER TRIGGER pl_not_copy FOR PL_CONTENT
2 ACTIVE BEFORE INSERT POSITION 0
3 AS
4 BEGIN
5     IF (NEW.TRACK_ID IN (SELECT TRACK_ID FROM PL_CONTENT WHERE PL_ID =
6         NEW.PL_ID)) THEN EXCEPTION ISCOPY;
7 END
```

Проверка дублирования для треков в альбомах:

```
1 CREATE OR ALTER TRIGGER trk_not_copy FOR TRACKLISTS
2 ACTIVE BEFORE INSERT POSITION 0
3 AS
4 BEGIN
5     IF (NEW.TRACK_ID IN (SELECT TRACK_ID FROM TRACKLISTS WHERE ALB_ID =
6         NEW.ALB_ID)) THEN EXCEPTION ISCOPY;
7 END
8 -- Завершение транзакции
9 COMMIT;
```

4.4 При добавлении новых данных в таблицу продаж альбомов запускать пересчет агрегатов по окончившемуся периоду: если последняя запись в таблицу продаж была добавлена до окончания очередного агрегационного этапа, а добавляемая — после, то запускать процедуру пересчета для окончившегося периода

```
1 CREATE OR ALTER TRIGGER aggregate FOR SELLING
2 ACTIVE BEFORE INSERT POSITION 0
3 AS
4 DECLARE VARIABLE OLD_DATE DATE;
5 DECLARE VARIABLE OLD_START_WEEK DATE;
6 DECLARE VARIABLE NEW_START_WEEK DATE;
7 DECLARE VARIABLE EXT INT;
8 BEGIN
9     SELECT FIRST 1 SOLD_DATE FROM SELLING ORDER BY SEL_ID DESC INTO :
10    OLD_DATE;
11
12    /* Получение дат начала недель */
13    EXT = EXTRACT(WEEKDAY FROM :OLD_DATE);
14    IF (EXT = 0) THEN EXT = 7; -- Воскресенье
15    OLD_START_WEEK = :OLD_DATE+1-EXT;
16
17    EXT = EXTRACT(WEEKDAY FROM NEW.SOLD_DATE);
18    IF (EXT = 0) THEN EXT = 7;
```

```

18 NEW_START_WEEK = NEW.SOLD_DATE+1-EXT;
20 IF (:OLD_DATE != NEW.SOLD_DATE) THEN
    EXECUTE PROCEDURE SP2 'DAY', :OLD_DATE;
22 IF (:OLD_START_WEEK != :NEW_START_WEEK) THEN
    EXECUTE PROCEDURE SP2 'WEEK', :OLD_DATE;
24 IF (DATEDIFF(MONTH, :OLD_DATE, NEW.SOLD_DATE) > 0) THEN
    EXECUTE PROCEDURE SP2 'MONTH', :OLD_DATE;
26 IF (DATEDIFF(YEAR, :OLD_DATE, NEW.SOLD_DATE) > 0) THEN
    EXECUTE PROCEDURE SP2 'YEAR', :OLD_DATE;
28 EXECUTE PROCEDURE SP2 'CITY', :OLD_DATE;
    EXECUTE PROCEDURE SP2 'COUNTRY', :OLD_DATE;
30 END
    -- Завершение транзакции
32 COMMIT;

```

5 Выводы

В ходе выполнения данной лабораторной работы было проведено ознакомление с возможностями реализации более сложной обработки данных на стороне сервера с помощью триггеров, разработаны триггеры автоматического заполнения первичного ключа, контроля целостности БД, остановки добавления записи в таблицу по исключению и и триггер, вызывающий процедуру.

Основной областью применения триггеров является контроль целостности базы данных любой сложности. С помощью триггеров существенно упрощаются приложения, т.к. часть логики приложения автоматически активизируется при обновлении БД и выполняется на стороне сервера.

К недостаткам триггеров можно отнести сложность разработки и поддержки, а также скрытая от пользователя функциональность, что может привести к незапланированным последствиям, причины которых потенциально непросто обнаружить. Триггеры влияют на производительность системы, т.к. перед выполнением каждой команды по изменению состояния БД необходимо проверять необходимость запуска триггера.

Таким образом, триггеры — очень полезный инструмент разработки БД, требующий аккуратного к нему подхода.