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

Отчет по лабораторной работе № 6
«Триггеры, вызовы процедур»
по дисциплине «Базы данных»

Работу выполнил:
студент гр. 43501/3 Хуторной Я. В.

Руководитель Мяснов А. В.

«__» _____ 2015 г

Санкт-Петербург
2015

Цели работы

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

Программа работы

- 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
- Создать триггер в соответствии с индивидуальным заданием, полученным у преподавателя
- Создать триггер в соответствии с индивидуальным заданием, вызывающий хранимую процедуру
- Выложить скрипт с созданными сущностями в svn
- Продемонстрировать результаты преподавателю

Триггеры

Для каждой таблицы может быть назначена хранимая процедура без параметров, которая вызывается при выполнении оператора модификации этой таблицы (INSERT, UPDATE, DELETE). Такие хранимые процедуры получили название триггеров. Триггеры выполняются автоматически, независимо от того, что именно является причиной модификации данных - действия человека оператора или прикладной программы. "Усредненный" синтаксис оператора создания триггера:

```
CREATE TRIGGER <имя_триггера>
        ON <имя_таблицы>
        FOR { INSERT | UPDATE | DELETE }
            [, INSERT | UPDATE | DELETE ] ...
        AS <SQL_оператор>
```

Ключевое слово ON задает имя таблицы, для которой определяется триггер, ключевое слово FOR указывает какая команда (команды) модификации данных активирует триггер. Операторы SQL после ключевого слова AS описывают действия, которые выполняет триггер и условия выполнения этих действий. Здесь может быть перечислено любое число операторов SQL, вызовов хранимых процедур и т.д. Использование триггеров очень удобно для выполнения операций контроля ограничений целостности.

Выполнение работы

1. Создан триггер, который автоматически заполняет поле CLIENT_ID таблицы CLIENTS инкрементально.

```
set names WIN1251;
connect 'C:\DB\MUSICLIBRARY.FDB' user 'SYSDBA' password 'masterkey';

create generator autoinc;
set generator autoinc to 0;

create trigger trig_autoinc_clientid for CLIENT before insert
as
begin
if((new.CLIENT_ID is null) or (new.CLIENT_ID = 0))  then
 begin
 new.CLIENT_ID = gen_id(autoinc, 1);
 end
end;

commit;
```

2. Создан триггер для контроля целостности данных в подчиненных таблицах при удалении/изменении записей в таблице CD. При попытке удаления или изменения записи в таблице CD, на которую присутствуют внешние ссылки, триггер выдает ошибку и возвращает старое значение.

```
set names WIN1251;
connect 'C:\DB\MUSICLIBRARY.FDB' user 'SYSDBA' password 'masterkey';

create exception my_exc 'Error! This field have dependence.
Remains the old value.';

create trigger trig_control_cel_data for CD before
update or delete
as
begin
if ((OLD.CD_ID in (select SALE_CD.CD_ID from SALE_CD)) or
(OLD.CD_ID in (select CD_GENRE.CD_ID from CD_GENRE))  or
(OLD.CD_ID in (select CD_PLAYERS.CD_ID from CD_PLAYERS)) or
(OLD.CD_ID in (select CD_RECORDING_STUDIO.CD_ID from
CD_RECORDING_STUDIO)) or
(OLD.CD_ID in (select CD_TRACK.CD_ID from CD_TRACK)))
```

```
then
begin
exception my_exc;
end

when exception my_exc  do
begin
NEW.CD_ID = OLD.CD_ID;
end

end;

commit;
```

3. Сделан триггер в соответствии с индивидуальным заданием. При заказе трека клиентом проверять не куплен ли был это трек ранее как отдельно, так и в рамках альбома. Если был приобретен - не добавлять трек в заказ.

```
set names WIN1251;
connect 'C:\DB\MUSICLIBRARY.FDB' user 'SYSDBA' password 'masterkey';

/* При заказе трека клиентом проверять не куплен ли был это трек ранее
 как отдельно, так и в рамках альбома. Если был приобретен - не добавлять
 трек в заказ.*/

create exception my_exc2 'Error! This song has been already bought.';

create trigger trig_control_sale_track for SALE_TRACK before
insert
as
begin
if ((NEW.TRACK_ID in (select SALE_TRACK.TRACK_ID from
SALE_TRACK where
SALE_TRACK.client_id = NEW.CLIENT_ID)) or (NEW.TRACK_ID in (select
SALE_TRACK.TRACK_ID from SALE_TRACK, CD_TRACK, CD, SALE_CD
where
SALE_TRACK.TRACK_ID = CD_TRACK.TRACK_ID and
CD_TRACK.CD_ID = CD.CD_ID and
CD.CD_ID = SALE_CD.CD_ID and SALE_CD.CLIENT_ID =
NEW.CLIENT_ID)))
then
begin
exception my_exc2;
```

```
end
end;
commit;
```

Попробуем добавить клиенту с номером 105 заказ на трек с номер 1, который уже был ранее приобретён им. В результате выводится ошибка с текстом о том, что такая песня уже была куплена этим клиентом.

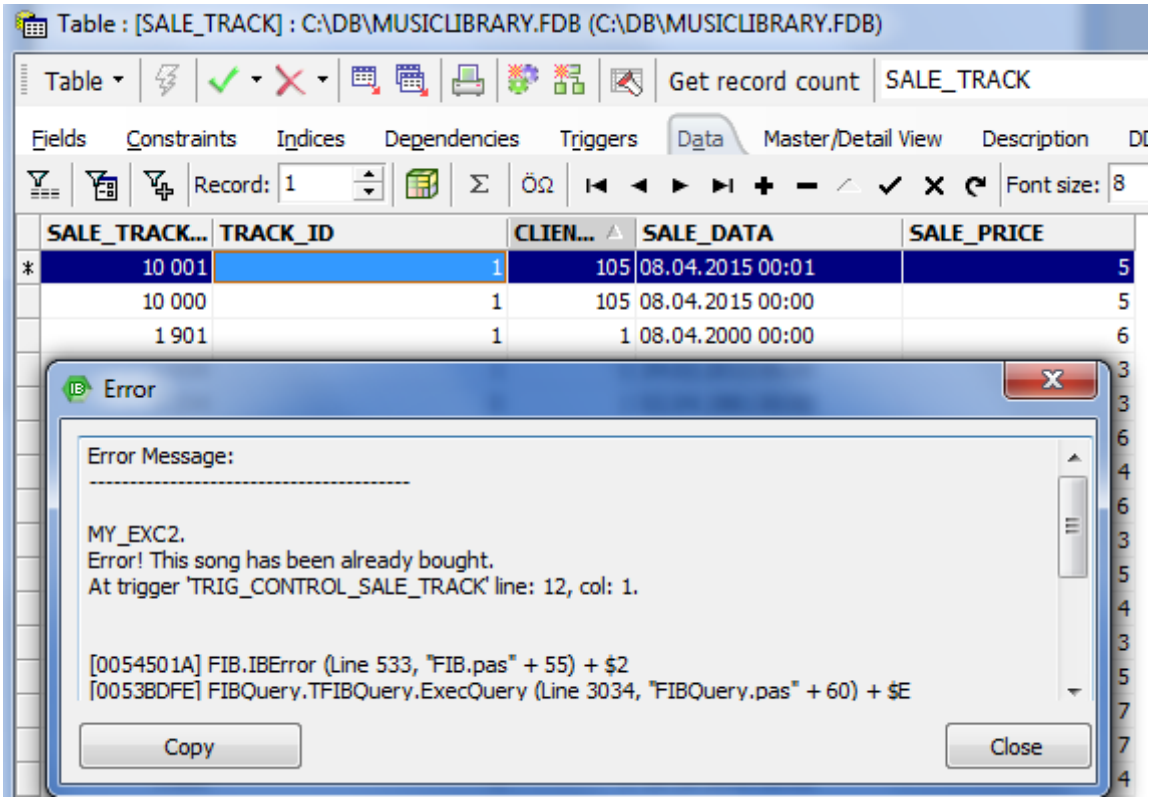

Рис. 1. Результат работы триггера

4. Сделан триггер в соответствии с индивидуальным заданием. При добавлении трека в альбом сделать проверку на дубли. При дублировании - не добавлять.

```
set names WIN1251;
connect 'C:\DB\MUSICLIBRARY.FDB' user 'SYSDBA' password 'masterkey';

/* При добавлении трека в альбом сделать проверку на дубли. При
 дублировании - не добавлять.*/

create exception my_exc3 'Error! This song has been already in cd.';

create trigger trig_control_insert_track_in_cd for CD_TRACK before
insert
as
begin
```

```
if (
NEW.TRACK_ID in (select CD_TRACK.TRACK_ID from CD_TRACK where
CD_TRACK.CD_ID = NEW.CD_ID))
then
begin
exception my_exc3;
end
end;

commit;
```

Попробуем добавить в CD трек, который в нем уже содержится. При попытке добавления в альбом повторного трека, выводится ошибка с текстом о том, что такой трек уже содержится в этом CD.

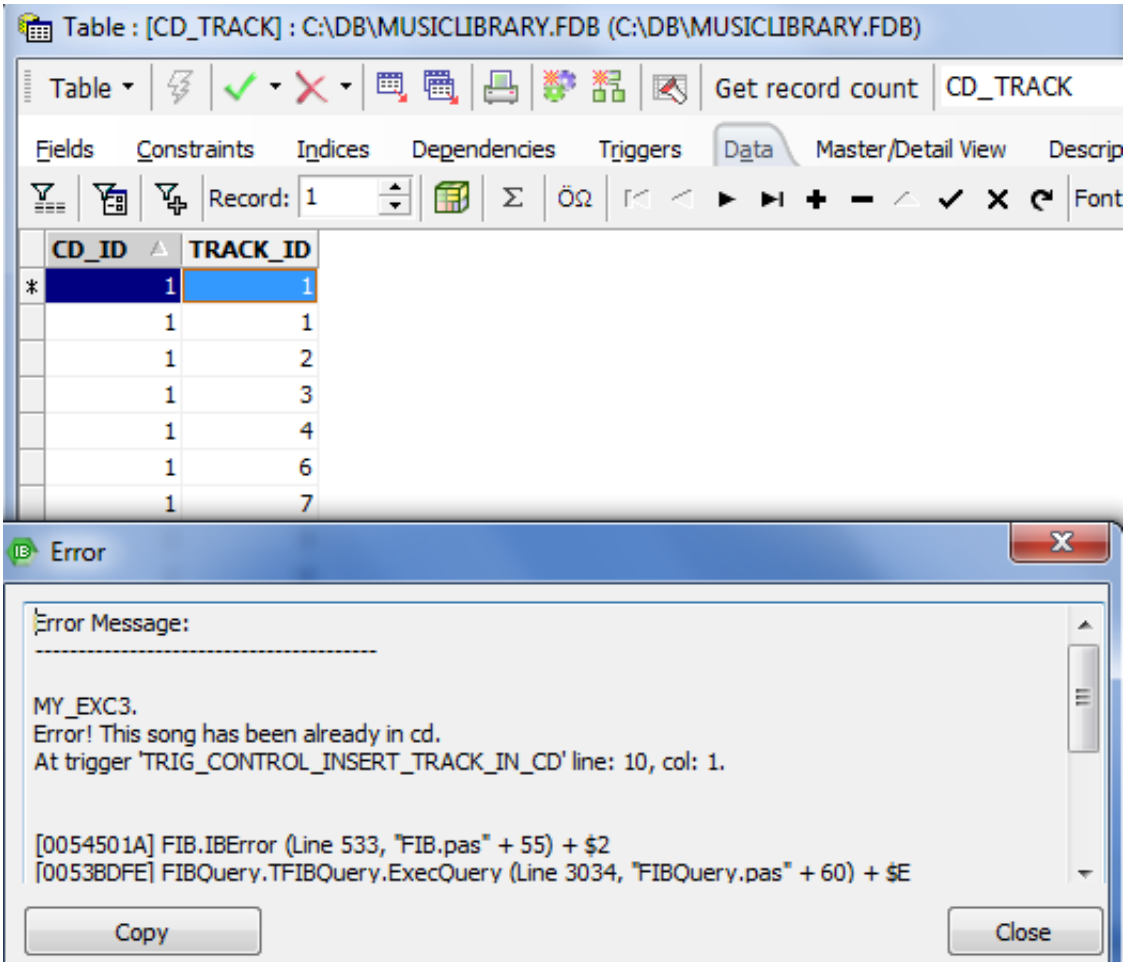

Рис. 2. Результат работы триггера

Вывод

В данной работе были изучены триггеры и генераторы, исключения. Триггер – это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнения которой обусловлено действием по модификации данных. Триггеры позволяют контролировать и изменять операции, проводимые над таблицами БД. Например, триггеры позволяют симулировать автоинкрементирующиеся ключи таблицы, позволяют обеспечивать логическую целостность данных в соответствии с предметной областью. Генераторы являются специальными объектами БД, которые генерируют уникальные последовательные числа. Одним из применений генераторов является их использование в триггерах для автоинкрементирующихся ключей. Генераторы обеспечивают уникальность генерируемых значений даже при параллельной обработке нескольких запросов. Исключение - это переменная в PL/SQL, возбуждаемая во время выполнения блока. В случае возбуждения исключения (исключительной ситуации), блок всегда прекращает работу, но можно задать обработчик исключения для выполнения заключительных действий.