Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №6

По дисциплине «Базы данных»

«Триггеры, вызовы процедур»

Работу выполнили студенты группы №43501/4 Н.С. Шаляпин\_\_\_\_\_\_\_\_\_

Работу принял преподаватель А.В. Мяснов\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2015

1. Цель работы

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

2 . Триггеры

Триггер (англ, trigger) — это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или из­менением UPDATE данных в определенном столбце заданной таблицы реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации слож­ной бизнес-логики. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рас­сматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

Момент запуска триггера определяется с помощью ключевых слов BEFORE (триггер запускается до выполнения связанного с ним события; например, до добавления записи) или AFTER (после события). В случае, если триггер вызывается до события, он может внести изменения в модифицируемую событием запись (конечно, при условии, что собы­тие — не удаление записи). Некоторые СУБД накладывают ограничения на операторы, которые могут быть использованы в триггере (например, может быть запрещено вносить изменения в таблицу, на которой «висит» триггер, и т, п.).

Кроме того, триггеры могут быть привязаны не к таблице, а к представлению (VIEW). В этом случае с их помощью реализуется механизм «обновляемого представления». В этом случае ключевые слова BEFORE и AFTER влияют лишь на последовательность вызова триггеров, так как собственно событие (удаление, вставка или обновление) не происходит.

1. 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 svn;
   5. Продемонстрировать результаты преподавателю,
2. Ход работы

* Был создан триггер, который автоматически заполняет поле marks\_id таблицы marks (имитация автоинкремента):

Триггер автоинкремента.

|  |
| --- |
| CREATE GENERATOR gen\_marks\_id;  SET GENERATOR gen\_marks\_id TO 0;  set term !! ;  CREATE TRIGGER marks\_bi FOR marks  ACTIVE BEFORE INSERT POSITION 0  AS  BEGIN  NEW.marks\_id = GEN\_ID(gen\_marks\_id, 1);  END!!  set term ; !! |

* Был создан триггер, который проверяет данные на целостность: при попытке уда­ления или изменения записи в таблице марок, на которую присутствуют внешние ссылки, он выдает ошибку.

+Триггер проверки целостности.

|  |
| --- |
| create trigger control\_del\_mark for marks before delete or update  as  begin  if (OLD.marks\_id in ( select model\_car.marks\_id from model\_car )) then  exception exc\_err;  end; |

* В соответствии с индивидуальным заданием был создан триггер, который при проверке дублей при добавлении опции к заказу. При дубле - не добавлять.

+Триггер проверки на целостность

|  |
| --- |
| create trigger checkDubles before insert on trade\_additional\_options  AS  declare variable id int;  BEGIN  EXECUTE PROCEDURE forTrigCheckDubl(new. additional\_options\_id)  RETURNING\_VALUES :id;  if(id is null) then exit;  else exception exc\_err;  END  create PROCEDURE forTrigCheckDubl(additional\_options\_id INTEGER)  returns (idd int)  as  BEGIN  select trade\_additional\_options.trade\_additional\_options\_id from trade\_additional\_options  where additional\_options\_id =: additional\_options\_id  rows 1 into idd;  END |

* В соответствии с индивидуальным заданием был создан триггер, который при по­пытке добавить в заказ опцию, которая уже есть в выбранной комплектации не осу­ществляет добавление.

+Триггер проверки на целостность

|  |
| --- |
| create exception check\_opt\_ex 'inserting option that already in deal ';  drop trigger check\_opt ;  create trigger check\_opt before insert on trade\_additional\_options  as  begin  if (new.additional\_options\_id not in ( select trade\_additional\_options.additional\_options\_id  from trade\_additional\_options  where trade\_additional\_options.trade\_id = new. trade\_id)) then  exit ;  exception check\_opt\_ex ;  end; |

4. Вывод

Выполнив данную лабораторную работу, мы изучили триггеры. Триггер - это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определенном столбце заданной таблицы реляционной базы данных. Триггер может быть в одном из двух состояний активном (ACTIVE) или неактивном (INACTIVE). По умолчанию триггеры создаются в активном состоянии. Триггер может выполняться в одной из двух фаз, связанных с запрошенными изменениями состояния данных. Ключевое слово BEFORE означает, что триггер вызывается до наступления соответствующего события (событий, если их указано несколько), AFTER — после наступления события (событий). Для табличного триггера может быть указано одно из событий таблицы (представления) — INSERT (добавление), UPDATE (изменение), DELETE (удаление) — или несколько событий, разделённых ключевым словом OR, при которых вызывается триггер.

Генератор — это специальный объект базы данных, который генерирует уникальные последовательные числа. Одним из применений генераторов является их использование в триггерах автоинкрементирования ключей. В таких триггерах необходимо использовать генераторы, так как они обеспечивают уникальность генерируемых значений даже при параллельной обработке нескольких запросов.

При выполнении работы были созданы триггеры в соответствии с индивидуальным заданием. Проблем в ходе выполнения работы не возникло.