Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

Отчёт

по лабораторной работе № 5

“Триггеры SQL”

по дисциплине «Базы данных»

Работу выполнил студент гр.43501/13

Михайлов Д.М. \_\_\_\_\_

Преподаватель:

Моисеев М.Ю. \_\_\_\_\_

Санкт-Петербург

2013

1. **Цель работы**

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

1. **Программа работы**

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

1. Создать триггер, который рассчитывает стоимость путевки (за время проживания в отелях) после добавления booking. Продемонстрировать его корректную работу.
2. Создать триггер, который при добавлении клиентом рейтинга отеля проверяет, жил ли он там. Если нет – удалить рейтинг. Продемонстрировать его корректную работу.
3. **Выполнение программы работы**

**3.1. После добавления бронирования необходимо просчитать стоимость общего проживания за путевку. Для этого пересчитываем для каждого бронирования, принадлежащего путевке, стоимость комнаты, умноженную на время проживания. Триггер носит характер “после добавления”.**

/\*Стоимость путевки рассчитывается после добавления booking\*/

connect 'C:\Users\Danya\Documents\Database\tour\_named\tour\_named.fdb' user 'SYSDBA' password 'masterkey';

create or alter procedure card\_cost\_update(

upd\_card\_id integer, sum\_card\_cost integer)

-- returns (card\_cost\_output integer)

as

begin

update card set card\_cost = :sum\_card\_cost where card\_id = :upd\_card\_id;

-- card\_cost\_output = :sum\_card\_cost;

suspend;

end;

CREATE OR ALTER trigger card\_cost for booking

active after insert position 0

AS

declare variable sum\_card\_cost integer;

declare variable card\_connection integer;

declare variable card\_days integer;

declare variable card\_room\_cost integer;

begin

sum\_card\_cost = 0;

card\_days = 0;

card\_room\_cost =0;

card\_connection = new.card\_booking\_id; -- thats how to choose fields of table

-------------------------- finding card - booking - room

for select booking.b\_duration, room\_info.ri\_room\_cost

from card, booking, room\_info

where card\_id = :card\_connection and card\_id = card\_booking\_id

and ri\_booking\_id = ri\_id

-- and tk\_tk\_connect\_id = tk\_id

into :card\_days,:card\_room\_cost

do begin

sum\_card\_cost = :sum\_card\_cost + (:card\_days \* :card\_room\_cost);

end

execute procedure card\_cost\_update(:card\_connection,:sum\_card\_cost);

end;

Используем хранимую процедуру для обновления стоимости.

**Для тестирования функциональности используем команды:**

---------------Для примера возьмем путевку с id=100

--------------- смотрим текущую стоимость

select card.card\_cost

from card

where card\_id = '100';

--------------- Внося бронирование, привязываем выбранную комнату путевке

insert into booking values('100',(SELECT MAX(ri\_id) FROM room\_info)-2,'11-DEC-2017', 2);

--------------- Смотрим таблицу дней проживания и стоимости проживания в день

select booking.b\_duration, room\_info.ri\_room\_cost

from card, booking, room\_info

where card\_id = '100' and card\_id = card\_booking\_id

and ri\_booking\_id = ri\_id;

--------------- Снова смотрим текущую стоимость

select card.card\_cost

from card

where card\_id = '100';

На Рис 1 и 2 показаны результаты работы триггера при выполнении вышеописанных операций.



Рис.1

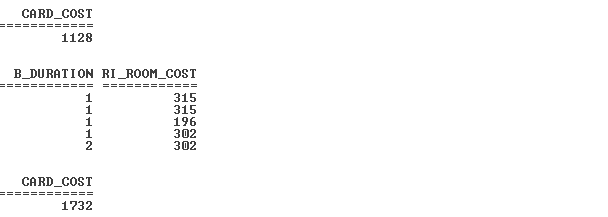


Рис.2

Как видно из результатов работы триггера, обновление стоимости произошло успешно, в таблице Card соответствующие поля были обновлены.

**3.2. При добавлении клиентом рейтинга отеля проверить, жил ли он там. Если нет - удалить рейтинг. Триггер носит характер “после добавления”.**

connect 'C:\Users\Danya\Documents\Database\tour\_named\tour\_named.fdb' user 'SYSDBA' password 'masterkey';

CREATE OR ALTER trigger check\_rating for hotel\_rating

active after insert position 0

AS

begin

if ( (select hotel.h\_id

from client, card, booking, room\_info, hotel

where new.p\_hotel\_rating\_id = p\_id and p\_id = p\_card\_id

and card\_id = card\_booking\_id and ri\_booking\_id = ri\_id

and h\_ri\_id = h\_id and new.h\_hotel\_rating\_id = h\_id)

is null) then

delete from hotel\_rating where p\_hotel\_rating\_id = new.p\_hotel\_rating\_id;

end

Делаем вложенный запрос, который проверяет наличие связки клиент – отель , если ее нет, удаляет рейтинг.

**Для тестирования функциональности используем команды:**

--------------- activating trigger

insert into hotel\_rating values('0','0',10);

--------------- show trigger activation result

select hotel\_rating.p\_hotel\_rating\_id

from hotel\_rating

where p\_hotel\_rating\_id = '0' and p\_hotel\_rating\_id = '0';

При выполнении команд ничего не выводится, т.е. запись после добавления удаляется.

Следует отметить, что триггеры в нашем случае можно использовать и при обновлении, а в случае с обновлением стоимости и при удалении.

Пример: active after insert or update position 0

Однако, в некоторых случаях будет необходимо учесть расположение псевдозаписей old и new.

|  |  |  |
| --- | --- | --- |
| **Активизирующий оператор** | **:OLD** | **:NEW** |
| INSERT | Не определена во всех полях содержится NULL значения | Значения, которые будут введены после выполнения оператора. |
| UPDATE | Исходные значения содержащиеся в строке перед обновлением данных | Новые значения которые будут введены после выполнения оператора |
| DELETE | Исходные значения содержащиеся в строке перед ее удалением | Не определена во всех полях содержится NULL значения |

1. **Выводы**

Триггер — это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено наступлением определенного события (действием) —добавлением INSERT или удалением DELETE строки в заданной таблице, или модификации UPDATE данных в определенном столбце заданной таблицы реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики, приводящей к сокращению пользовательских операций. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан.

Рассмотрим достоинства и недостатки триггеров:

Использование триггеров имеет следующие преимущества:

- автоматическое исполнение сервером при выбранных условиях

- обеспечение надежности функционирования системы в условиях использования системы пользователем

- возможность автоматической проверки триггерами DML корректности данных, определенные значения которых необходимо хранить в соответствующей таблице;

- создание гибкой системы поддержки целостности БД;

- возможность создания системы автоматической фиксации изменений в таблицах БД (ведение журнала изменений).

- как и в хп, используется более развернутое манипулирование данными, включающее сочетание описания действий с неким языком программирования (поддержка локальных переменных, условных операторов, циклов, вызовов процедур).

К недостаткам можно отнести возможную запутанность системы, так как триггеры не видны пользователю, в следствие чего, если досконально не знать структуру БД, можно легко допустить ошибки, которые будут не понятны в первого взгляда.

Кроме того, если в структуре триггера допущены алгоритмические ошибки, то возможны непредвиденные результаты работы триггера, которые будут получены возможно в редких ситуациях, но которые приведут к непредвиденным ошибкам. Пример – рекурсивное удаление записей.

Недостатком может быть невозможность создания виртуальных таблиц view, что приводит ко вложенным циклам и необходимости перебора большего количества данных.