##

ЛАБОРАТОРНАЯ РАБОТА N4

ТЕМА: Триггеры базы данных.
Защита данных. Привилегии

Используемые объекты:
1. Учебная база данных - таблицы SAL, CUST, ORD
2. Описание учебной базы данных - таблица OPDB2

Триггеры базы данных - это процедуры, которые хранятся в базе данных и автоматически вызываются (запускаются), когда модифицируется таблица.

Так же, как и хранимая процедура, триггер может состоять из предложений SQL и блоков PL/SQL и может вызывать другие хранимые процедуры. Однако процедуры и триггеры различаются по способу их вызова. В то время как процедура явно вызывается пользователем или приложением, триггер неявно запускается (исполняется) ORACLE, когда выдается предложение INSERT, UPDATE или DELETE, независимо от того, какой пользователь сейчас подключен.

Триггеры обычно используются для:
- автоматической генерации значений вычисляемых столбцов,
- предотвращения незаконных транзакций,
- регистрации событий

и других действий, дополняя стандартные возможности ORACLE и способствуя созданию гибко настраиваемой системы управления базой данных.

Триггер имеет три основные части:
* событие, или предложение, триггера
* ограничение триггера
* действие триггера

Событие триггера, или предложение триггера, - это предложение SQL, которое заставляет триггер выполниться. Событием триггера может быть предложение INSERT, UPDATE или DELETE для конкретной таблицы.
Например:

```
AFTER UPDATE OF comm ON mysal
```

которое означает, что триггер срабатывает при обновлении столбца COMM в строке таблицы MYSAL.

Ограничение триггера - это необязательная возможность, которая используется в триггерах, возбуждаемых по каждой строке. Цель ограничения - наложить условие на выполнение триггера. Оно задает с помощью фразы WHEN булевское (логическое) выражение, которое должно быть истинным (TRUE) для того, чтобы триггер сработал. Например:

```
WHEN (new.comm < 0.25)
```

Действие триггера - это процедура (блок PL/SQL), содержащая предложения SQL и PL/SQL, которые будут выполнены, если выдано предложение триггера, а ограничение триггера вычислено как TRUE. Ниже показан пример процедуры действий триггера.

```
FOR EACH ROW

BEGIN
    INSERT INTO register VALUES ('update', 'mysal', user, sysdate);
END;
```

В данном примере триггер выполняет вставку строки в таблицу REGISTER. Строка содержит информацию о том, что была выполнена команда UPDATE для таблицы MYSAL, имя пользователя, выполнившего изменение, а также дату и время изменения.

Триггеры создаются с помощью команды CREATE TRIGGER. Следующее предложение создает триггер, ассоциированный с таблицей EMP:

```
CREATE TRIGGER dummy
  BEFORE UPDATE ON emp
  FOR EACH ROW
  WHEN (new.empno > 0)
DECLARE
   /* переменные, константы, курсоры и т.п. */
BEGIN
   /* блок PL/SQL */
END;
```

ЗАДАНИЕ 1.

Создайте триггер, который регистрирует вставку каждой строки в одну из ваших таблиц. При регистрации триггер должен сохранять, кроме названия команды и таблицы, имя того, кто вставляет строки, и дату и время вставки.

Либо опция BEFORE, либо опция AFTER должна быть указана в предложении CREATE TRIGGER, чтобы точно специфицировать, когда должно исполняться тело триггера по отношению к исполнению предложения триггера.

Предложения в действии триггера имеют доступ как к новым, так и к старым значениям столбцов текущей строки, обрабатываемой триггером. Этот доступ предоставляют два имени (OLD и NEW), которые задаются как префиксы перед именами столбцов. Заметьте, что перед квалификаторами OLD и NEW должно кодироваться двоеточие, когда они используются в теле триггера, но двоеточие не допускается, когда эти имена используются в фразе WHEN.

Если триггер может быть возбужден более чем одним типом команды DML (например, "INSERT OR DELETE OR UPDATE OF emp"), то в теле триггера можно использовать условные предикаты INSERTING, DELETING и UPDATING, для того чтобы выполнять различные участки кода в зависимости от типа предложения,

активизирующего триггер. В коде внутри тела триггера вы можете использовать следующие условия:

```
IF INSERTING THEN . . . END IF;
IF DELETING  THEN . . . END IF;
IF UPDATING  THEN . . . END IF;
```

ЗАДАНИЕ 2.

Измените триггер из Задания 1 так, чтобы он регистрировал любые изменения таблицы, а не только вставку строк.

Для отмены команды, изменяющей данные, в триггере можно сгенерировать событие ошибки, если нарушается заданное условие. В следующем примере создается триггер, который срабатывает, если новый рейтинг продавца слишком высок (больше 500). Триггер вызывает ошибку, и команда изменения рейтинга не выполняется.

```
create or replace trigger large_rating
   before update or insert on mycust
   for each row
   begin
      if :new.rating > 500 then
      raise_application_error(-20001,'Mnogo');
      end if;
   end;
```

ЗАДАНИЕ 3.

Создайте триггер, который не позволяет добавлять новых продавцов из Рима.

Транзакция - это логический модуль, включающий набор команд SQL и блоков PL/SQL. Эти команды объединяются в транзакции для выполнения цельного, логически законченного действия. Целью такого объединения является обеспечение либо фиксации в БД всех изменений данных, выполненных командами транзакции, либо отмена всех этих изменений.

Транзакция начинается, когда пользователь подключается к БД Oracle и начинает с ней работать или сразу же после завершения предыдущей транзакции. В Oracle транзакция завершается, когда выполняется команда COMMIT или ROLLBACK или одна из команд SQL, относящаяся к категории DDL-команд. Кроме того, завершение транзакции происходит при окончании работы программы - нормальном или аварийном. В следующем примере показаны три транзакции:

```
SELECT * FROM mysal;
DELETE FROM mysal WHERE snum = 1002;
COMMIT;
DELETE FROM mysal WHERE snum = 1001;
DELETE FROM mysal WHERE snum = 1003;
ROLLBACK;
DELETE FROM mysal WHERE snum = 1006;
COMMIT;
```

Изменения, которые произведены в транзакции над данными, становятся видны для других пользователей только после их фиксации, например командой COMMIT. Как только транзакция зафиксирована в БД, ее откат невозможен.

Блок PL/SQL действует как единый оператор, и все изменения, сделанные в этом блоке, отменяются при его неудачном завершении. В следующем примере иллюстрируется эта ситуация:

```
DECLARE
  vSnum number;
BEGIN
   DELETE FROM mysal;
   DELETE FROM mycust;
   SELECT snum INTO vSnum
FROM myord WHERE onum = 8000;
END;
/
```

В этом примере, несмотря на то, что две первых команды DELETE сработали, блок PL/SQL завершается аварийно из-за ошибки в третьей команде - SELECT, и все изменения, выполненные в блоке, не будут зафиксированы в БД.

Любая команда SQL, относящаяся к категории DDL-команд, автоматически заканчивает транзакцию путем выдачи команды COMMIT. После этого начинается новая транзакция. К командам DDL относятся такие команды, как CREATE, ALTER, DROP, GRANT, RENAME, REVOKE, TRUNCATE.

В следующем примере показана команда DDL, которая автоматически выдает COMMIT. Команда ROLLBACK выполнит откат изменений, сделанных только двумя последними командами, а не всеми тремя командами DELETE:

```
DELETE FROM mysal;
CREATE TABLE tab (myfield NUMBER);
DELETE FROM mycust;
DELETE FROM myord;
ROLLBACK;
```

ЗАДАНИЕ 4.

Создайте свои копии таблиц SAL, CUST и ORD. Создайте скрипт, в котором удалите все данные из созданных копий таблиц следующим образом: из первой и второй таблиц - командой DELETE, а из третьей - командой TRUNCATE. Затем отмените изменения командой ROLLBACK и выведите содержимое таблиц. Выполните скрипт и объясните результат.

Для регламентации того, какие действия пользователь может выполнять, работая с базой данных Oracle, используется механизм привилегий. Привилегии бывают системного и объектного уровня. Системные привилегии для пользователей предоставляет, как правило, администратор БД. В основном они нужны для создания объектов базы данных. Чтобы пользователь мог определить, какими

системными привилегиями он обладает, он может воспользоваться представлением USER_SYS_PRIVS.

Объектные привилегии позволяют пользователю давать право другим пользователям работать с его объектами. Представления USER_TAB_PRIVS и TABLE_PRIVILEGES показывают, какие объектные привилегии предоставлены пользователю.

Для предоставления объектных привилегий используется команда GRANT с указанием вида привилегий, объектов, на которые выдаются привилегии, и пользователей, которым выдаются данные привилегии. В следующем примере выдаются объектные привилегии пользователю Alex на чтение и изменение таблицы MYTAB:

```
GRANT select, update ON mytab TO Alex;
```

Если необходимо предоставить привилегию объектного уровня всем пользователям базы данных, ее предоставляют пользователю PUBLIC.

При необходимости изъять объектную привилегию у одного или нескольких пользователей это можно сделать с помощью команды REVOKE. Например, приведенная ниже команда отнимает привилегию на изменение таблицы MYTAB у пользователей Alex и Tommy:

```
REVOKE update ON mytab FROM Alex, Tommy;
```

ЗАДАНИЕ 5.

Выдайте привилегию на чтение одной Вашей таблицы всем пользователям БД. Отнимите выданные Вами привилегии и посмотрите, как изменился набор привилегий.

Роли - это совокупность системны и объектных привилегий, которые сгруппированы под одним именем. Роли облегчают предоставление, изменение и отмену привилегий, особенно, когда это нужно делать для большого числа пользователей.

Роль создается командой CREATE ROLE, а удаляется командой DROP ROLE. Затем созданной роли предоставляются привилегии точно так же, как это делается для пользователей. Теперь роль с присвоенными ей привилегиями может быть предоставлена любому пользователю. Выдача привилегий ролям и ролей пользователям выполняется командой GRANT, а изъятие привилегий у ролей и ролей у пользователей – командой REVOKE. В следующем примере создается роль FRIEND, ей предоставляется привилегия на чтение таблицы MYTAB, а затем роль присваивается всем пользователям:

```
CREATE ROLE friend;
GRANT select ON mytab TO friend;
GRANT friend TO PUBLIC;
```

Для просмотра системной информации о ролях существует ряд представлений словаря данных:

DBA_ROLE_PRIVS - позволяет выяснить, какая роль была дана тому или иному пользователю
ROLE_SYS_PRIVS - набор системных привилегий, назначенных роли
ROLE_TAB_PRIVS - набор объектных привилегий, назначенных роли
SESSION_ROLES - набор ролей, действительных для сеанса пользователя
SESSION_PRIVS - набор привилегий, действительных для сеанса пользователя.

ЗАДАНИЕ 6.

Создайте роль и присвойте ей привилегии на чтение и изменение Вашей таблицы.

КОНТРОЛЬНЫЕ ВОПРОСЫ
==========

Контрольные вопросы выдает преподаватель после выполнения всех заданий.