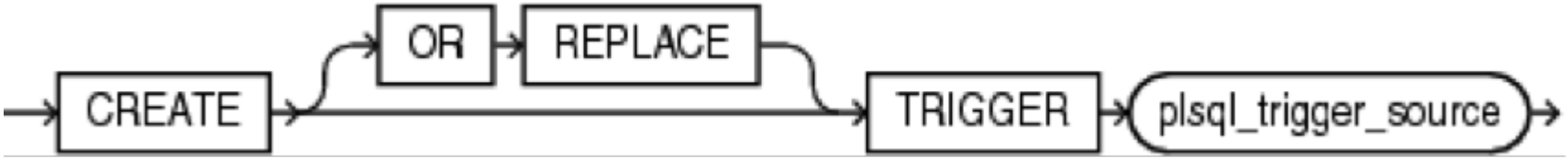
ORACLE 12c

PL/SQL Триггеры

Лекция 13

Триггеры

- **Триггер** – особый вид процедур, которые срабатывают по запускающему их событию



Применение триггеров

- для реализации сложных ограничений целостности базы данных;
- для аудита (контроля хранимой и изменяемой информации);
- для автоматического оповещения программ о произошедших событиях;

Привилегии

- **CREATE TRIGGER** - создавать, удалять, изменять в своей подсхеме
- **CREATE ANY TRIGGER** - создать любой триггер в любой схеме, кроме SYS, не рекомендуется для словаря, не разрешает менять текст триггера
- **ALTER ANY TRIGGER** - разрешать, запрещать, изменять, компилировать, любые, кроме SYS-триггеров, триггеры
- **DROP ANY TRIGGER** - удалять любой триггер, кроме SYS-триггеров
- **ADMINISTER DATABASE TRIGGER** - создавать, изменять, удалять системные триггеры, должен иметь привилегию CREATE TRIGGER или CREATE ANY TRIGGER

Привилегии

```
SYSTEM@sh1>
SYSTEM@sh1>  connect system/system@sh1;
Соединено.
sh1 - SYSTEM - 24.02.11
SYSTEM@sh1> grant create trigger to svvcore;

Привилегии предоставлены.
```

Привилегии

- Триггеры выполняется под правами создателя триггера
- Назначаются напрямую USERу, а не через роль

Транзакции

- Триггер – часть транзакции, ошибка в триггере откатывает операцию, изменения таблиц в триггере становятся частью транзакции.
- Если откатывается транзакция, изменения триггера тоже откатываются.

- Не может выдавать COMMIT/ROLBACK (исключение - только, если в теле триггера есть автономная транзакция)
- Может выдавать RAISE_APPLICATION_ERROR

Автономные транзакции

```
create table dbst_t ( msg varchar2(25) );

create or replace procedure dbst_NonAutonomous_Insert
as
begin
      insert into dbst_t values ( 'NonAutonomous Insert' );
      commit;
end;

create or replace procedure dbst_Autonomous_Insert
as
pragma autonomous_transaction;
begin
      insert into dbst_t values ( 'Autonomous Insert' );
      commit;
end;
```

```
begin
      insert into dbst_t values ( 'Anonymous Block' );
      dbst_NonAutonomous_Insert;
      rollback;
end;

Anonymous Block
NonAutonomous Insert
```

```
begin
      insert into dbst_t values ( 'Anonymous Block' );
      dbst_Autonomous_Insert;
      rollback;
end;

Autonomous Insert
```

Триггеры

- DML-триггеры
- Системные триггеры

Классификация триггеров

- **По привязанному объекту:**
  На таблице
- На представлении - instead of trigger
- **По событиям запуска:**
  Вставка записей - insert
- Обновление записей - update
- Удаление записей - delete
- **По области действия:**
  Уровень оператора - statement level triggers
- Уровень записи - row level triggers
- Составные триггеры - compound triggers
- **По времени срабатывания:**
  Перед выполнением операции – before
- После выполнения операции - after

DML-триггеры

- Время события:
  - AFTER (после события) – после записи в журнал,
  - BEFORE (до события) – до записи в журнал;

Уровни триггеров

- FOR EACH ROW (для каждой строки) - срабатывает для каждой измененной строки
- ПО УМОЛЧАНИЮ (операторный уровень) - срабатывает один раз на тригтерное событие

Порядок выполнения DML-триггеров

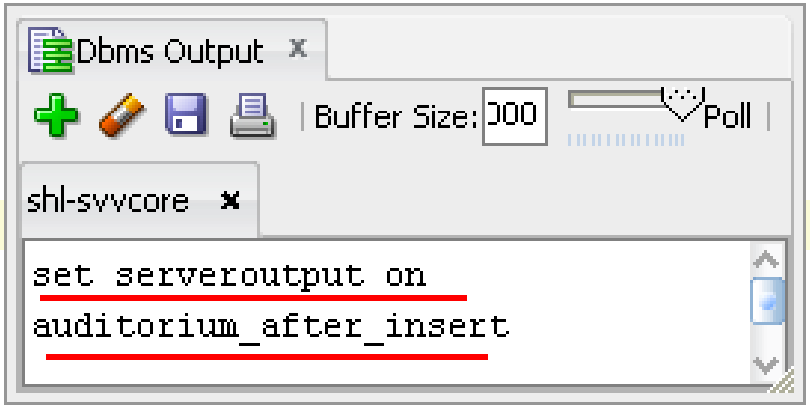
- операторные BEFORE;
- для каждой строки BEFORE;
- выполняется оператор;
- для каждой строки AFTER;
- операторные AFTER.

Триггерные события DML

INSERT	Событие возникает, когда добавляется строка (строки) в таблицу или представление.
UPDATE	Событие возникает, когда выполняется операция UPDATE над данными в таблице или представлении. Можно дополнительно задавать выражение OF для указания полей, при изменении которых срабатывает триггер.
DELETE	Событие возникает, когда удаляется строка (строки) из таблицы или представления. Триггер не срабатывает при выполнении команды TRUNCATE table.

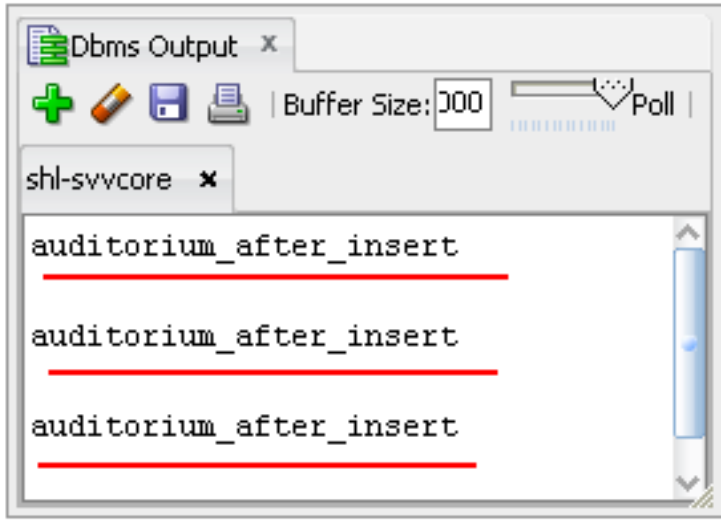
Создание триггера на вставку

```
create or replace trigger auditorium_after_insert
after insert on auditorium
begin
  dbms_output.put_line('auditorium_after_insert');
end;

insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
      values ('324-1', 'ЛК', 60);
```



Выполнение триггера

```
insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
       values ('313-1', 'ЛК', 80);
insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
       values ('222-4', 'ЛК', 60);
insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
       values ('114-4', 'ЛК', 90);
```



Триггер на обновление

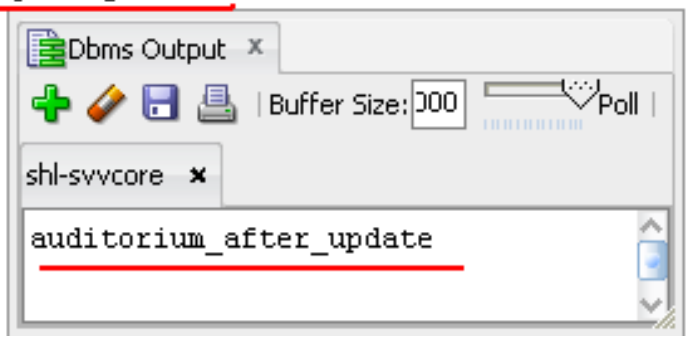
```
create or replace trigger auditorium after update
after update on auditorium
begin
  dbms_output.put_line('auditorium_after_update');
end;
```

```
select count(*) from auditorium;
```

COUNT(*)
4

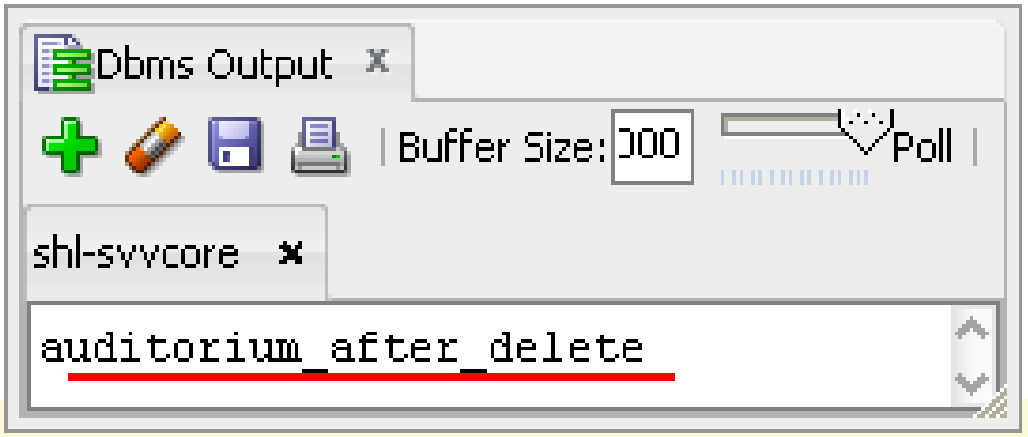
```
update auditorium set auditorium_capacity = 100;
```

4 rows updated



Триггер на удаление

```
create or replace trigger auditorium_after_delete
after delete  on auditorium
begin
  dbms_output.put_line('auditorium_after_delete');
end;

delete auditorium;

 4 rows deleted
```

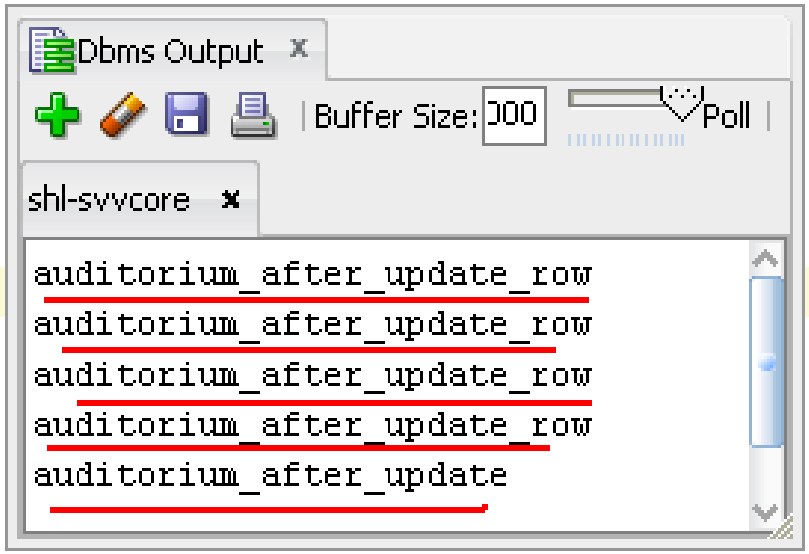


Триггеры for each row

```
create or replace trigger auditorium_after_update_row
after update on auditorium
for each row
begin
   dbms_output.put_line('auditorium_after_update_row');
end;
```

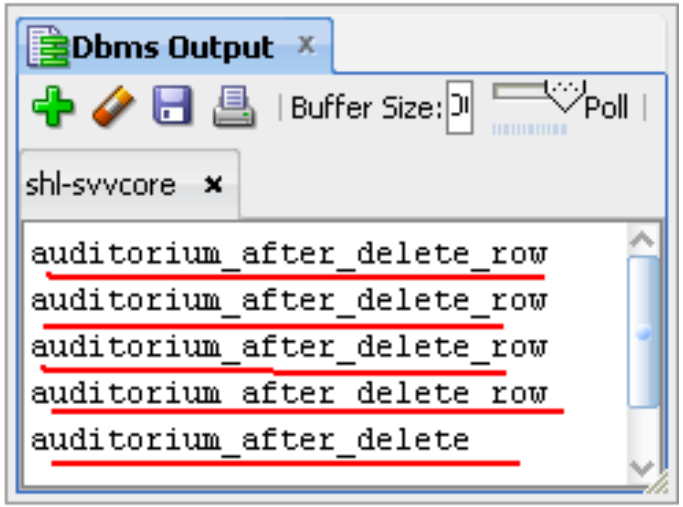
```
update auditorium set auditorium_capacity = 120;
```

```
4 rows updated
```



Триггеры for each row

```
create or replace trigger auditorium_after_delete_row
after delete  on auditorium
for each row
begin
   dbms_output.put_line('auditorium_after_delete_row');
end;

delete auditorium;

 4 rows deleted
```



Предикаты триггера

- Чтобы различать DML команды и события, которые выполняют триггер, используются триггерные предикаты INSERTING, UPDATING, and DELETING в условиях IF

```
drop trigger auditorium_after_insert;
drop trigger auditorium_after_update;
drop trigger auditorium_after_delete;


create or replace trigger auditorium_after_l
after insert or update or delete  on auditorium
begin
 if inserting then
   dbms_output.put_line('auditorium_after_insert_l');
 elsif updating then
    dbms_output.put_line('auditorium_after_update_l');
 elsif deleting then
    dbms_output.put_line('auditorium_after_delete_l');
 end if;
end;
```

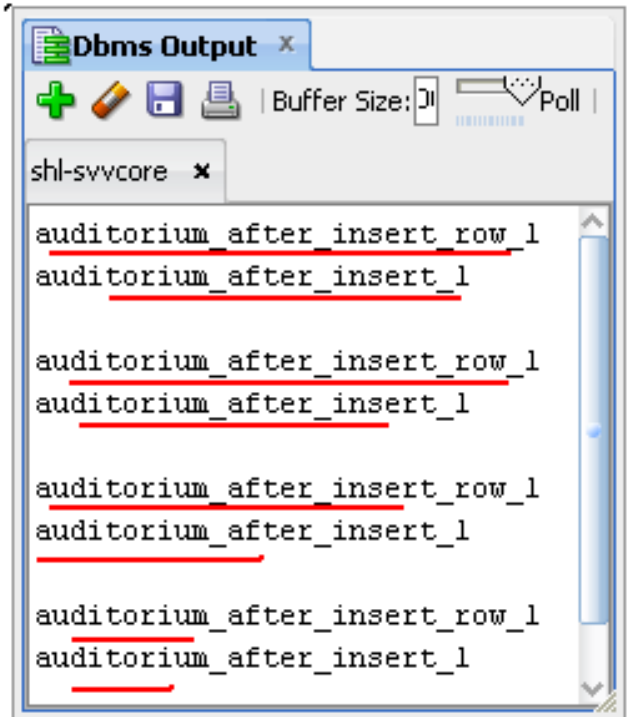
Предикаты триггера

- Для триггера for each row

```
drop trigger auditorium_after_update_row;
drop trigger auditorium_after_delete_row;

create or replace trigger auditorium_after_row_1
after insert or update or delete  on auditorium
for each row
begin
 if inserting then
    dbms_output.put_line('auditorium_after_insert_row_1');
 elsif updating then
    dbms_output.put_line('auditorium_after_update_row_1');
 elsif deleting then
    dbms_output.put_line('auditorium_after_delete_row_1');
 end if;
end;
```

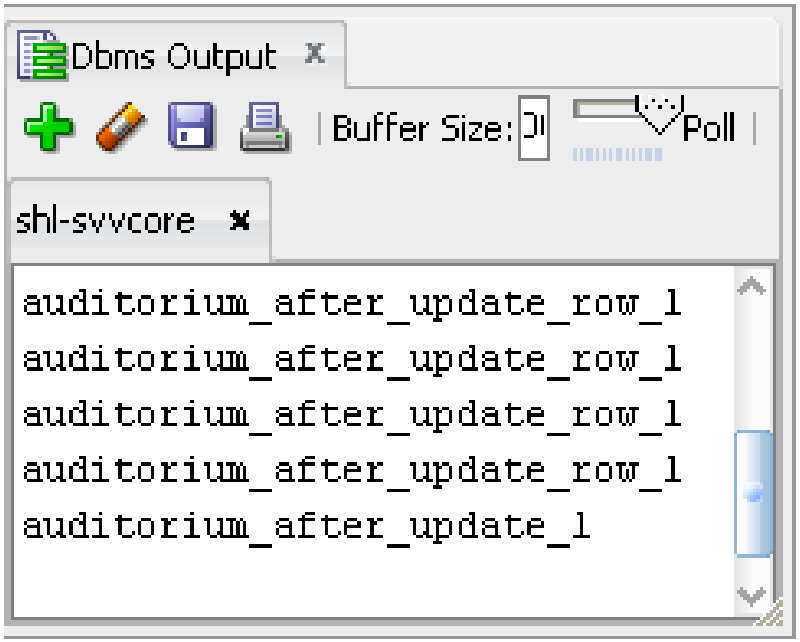
Применение набора триггеров

```
insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
       values ('324-1', 'ЛК', 60);
insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
       values ('313-1', 'ЛК', 80);
insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
       values ('222-4', 'ЛК', 60);
insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
       values ('114-4', 'ЛК', 90);
```



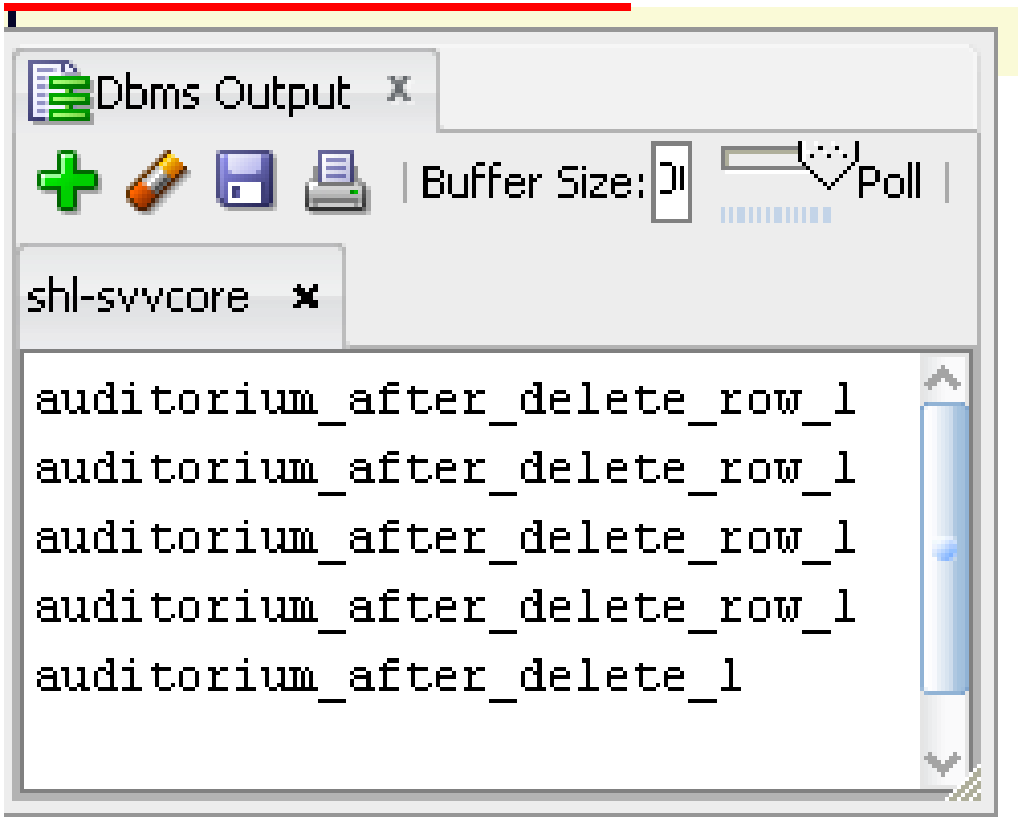
Применение набора триггеров

```
update auditorium set auditorium_capacity = 120;
```



Применение набора триггеров

```
delete auditorium;
```

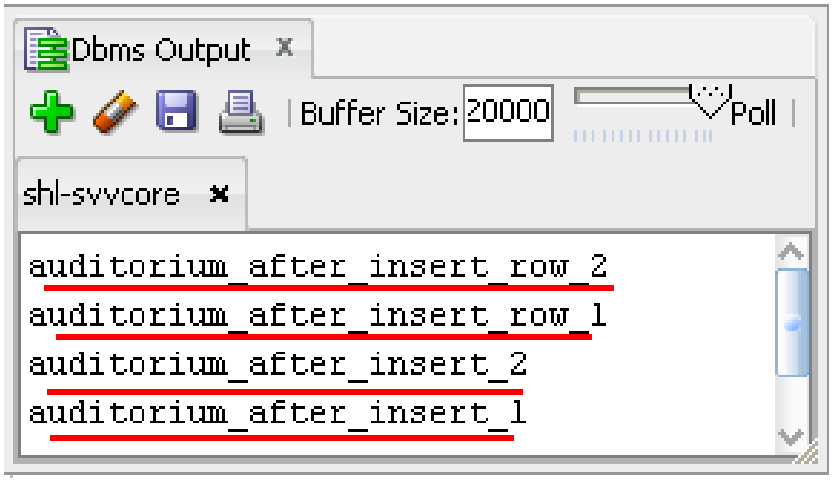


Порядок выполнения триггеров

```
create or replace trigger auditorium_after_2
after insert or update or delete  on auditorium
begin
 if inserting then
   dbms_output.put_line('auditorium_after_insert_2');
 elsif updating then
    dbms_output.put_line('auditorium_after_update_2');
 elsif deleting then
    dbms_output.put_line('auditorium_after_delete_2');
 end if;
end;

create or replace trigger auditorium_after_row_2
after insert or update or delete  on auditorium
for each row
begin
 if inserting then
    dbms_output.put_line('auditorium_after_insert_row_2');
 elsif updating then
    dbms_output.put_line('auditorium_after_update_row_2');
 elsif deleting then
    dbms_output.put_line('auditorium_after_delete_row_2');
 end if;
end;
```

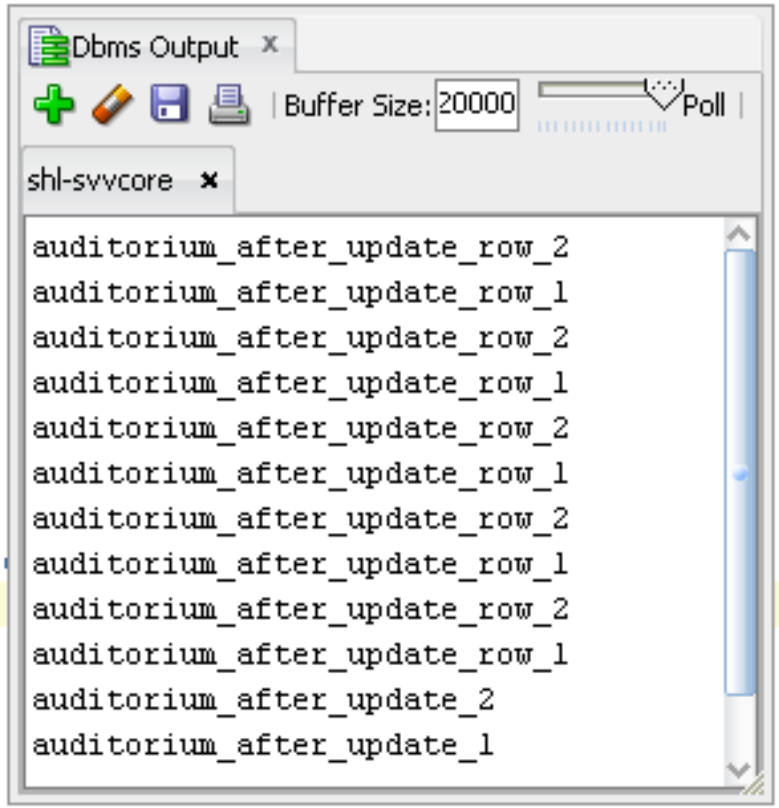
Порядок выполнения триггеров

```
insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
       values ('301-4', 'ЛК', 100);
```



Порядок выполнения триггеров

```
update auditorium set auditorium_capacity = 120;
```



Before - триггеры

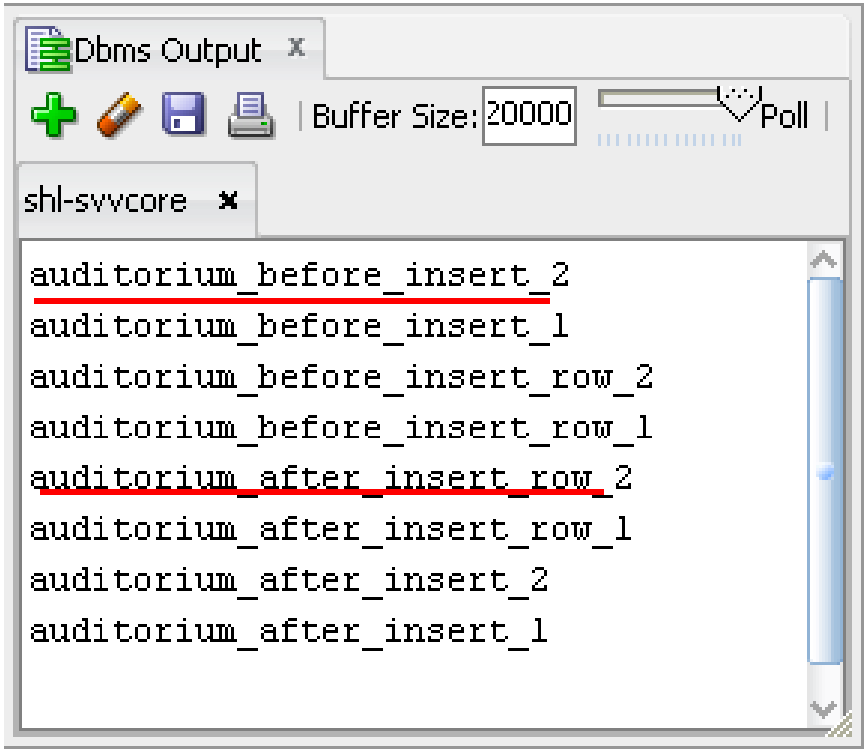
```
create or replace trigger auditorium_before_1
before insert or update or delete  on auditorium
begin
 if inserting then
   dbms_output.put_line('auditorium_before_insert_1');
 elsif updating then
   dbms_output.put_line('auditorium_before_update_1');
 elsif deleting then
   dbms_output.put_line('auditorium_before_delete_1');
 end if;
end;

create or replace trigger auditorium_before_row_1
before insert or update or delete  on auditorium
for each row
begin
 if inserting then
    dbms_output.put_line('auditorium_before_insert_row_1');
 elsif updating then
    dbms_output.put_line('auditorium_before_update_row_1');
 elsif deleting then
    dbms_output.put_line('auditorium_before_delete_row_1');
 end if;
end;
```

Before - триггеры

```
create or replace trigger auditorium_before_2
before insert or update or delete  on auditorium
begin
 if inserting then
   dbms_output.put_line('auditorium_before_insert_2');
 elsif updating then
   dbms_output.put_line('auditorium_before_update_2');
 elsif deleting then
   dbms_output.put_line('auditorium_before_delete_2');
 end if;
end;

create or replace trigger auditorium_before_row_2
before insert or update or delete  on auditorium
for each row
begin
 if inserting then
    dbms_output.put_line('auditorium_before_insert_row_2');
 elsif updating then
    dbms_output.put_line('auditorium_before_update_row_2');
 elsif deleting then
    dbms_output.put_line('auditorium_before_delete_row_2');
 end if;
end;
```

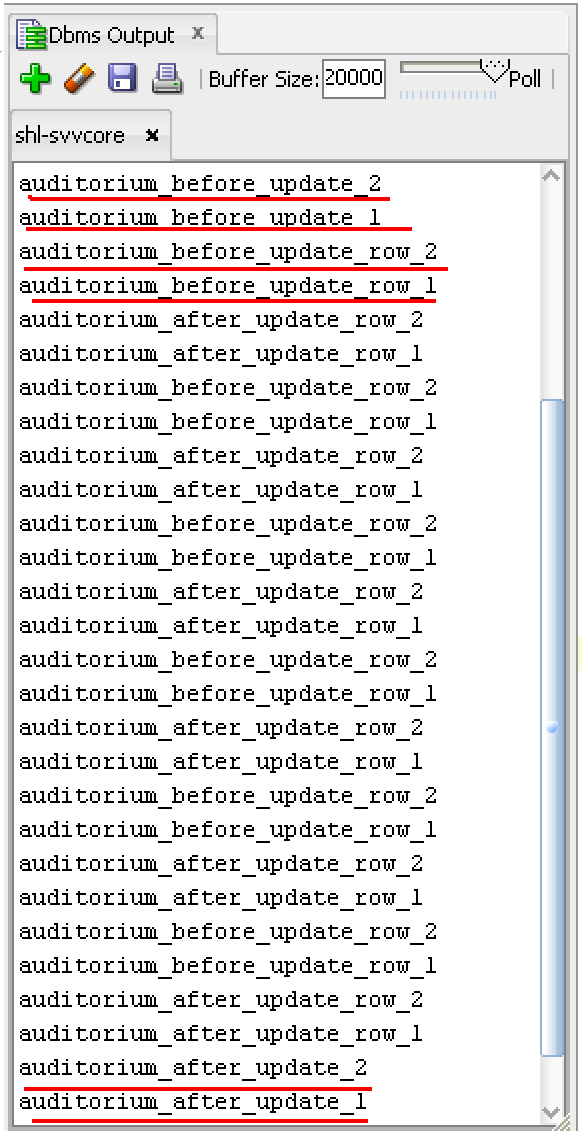
Before - триггеры

```
insert into auditorium(auditorium, auditorium_type,auditorium_capacity)
       values ('137-4', 'ЛК', 60);
```



Before - триггеры

```
update auditorium set auditorium_capacity = 100;
```



Псевдозаписи new, old

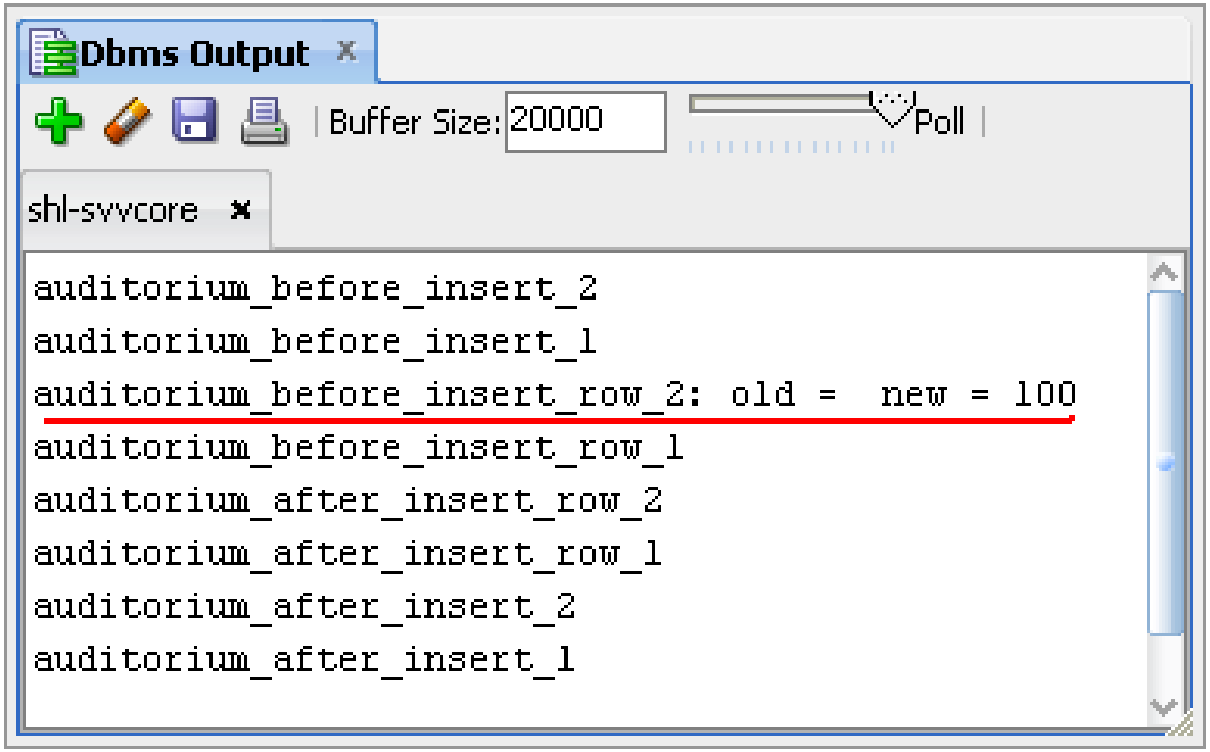
Операция срабатывания триггера	OLD.column	NEW.column
Insert	Null	Новое значение
Update	Старое значение	Новое значение
Delete	Старое значение	Null

Псевдозаписи new, old

```
create or replace trigger auditorium_before_row_2
before insert or update or delete  on auditorium
for each row
begin
 if inserting then
    dbms_output.put_line
            ('auditorium_before_insert_row_2:'
             ||' old = '|| :old.auditorium_capacity
             ||' new = '|| :new.auditorium_capacity
            );
 elsif updating then
   dbms_output.put_line
            ('auditorium_before_update_row_2:'
             ||' old = '|| :old.auditorium_capacity
             ||' new = '|| :new.auditorium_capacity
            );
 elsif deleting then
    dbms_output.put_line
            ('auditorium_before_delete_row_2:'
             ||' old = '|| :old.auditorium_capacity
             ||' new = '|| :new.auditorium_capacity
            );
 end if;
end;
```

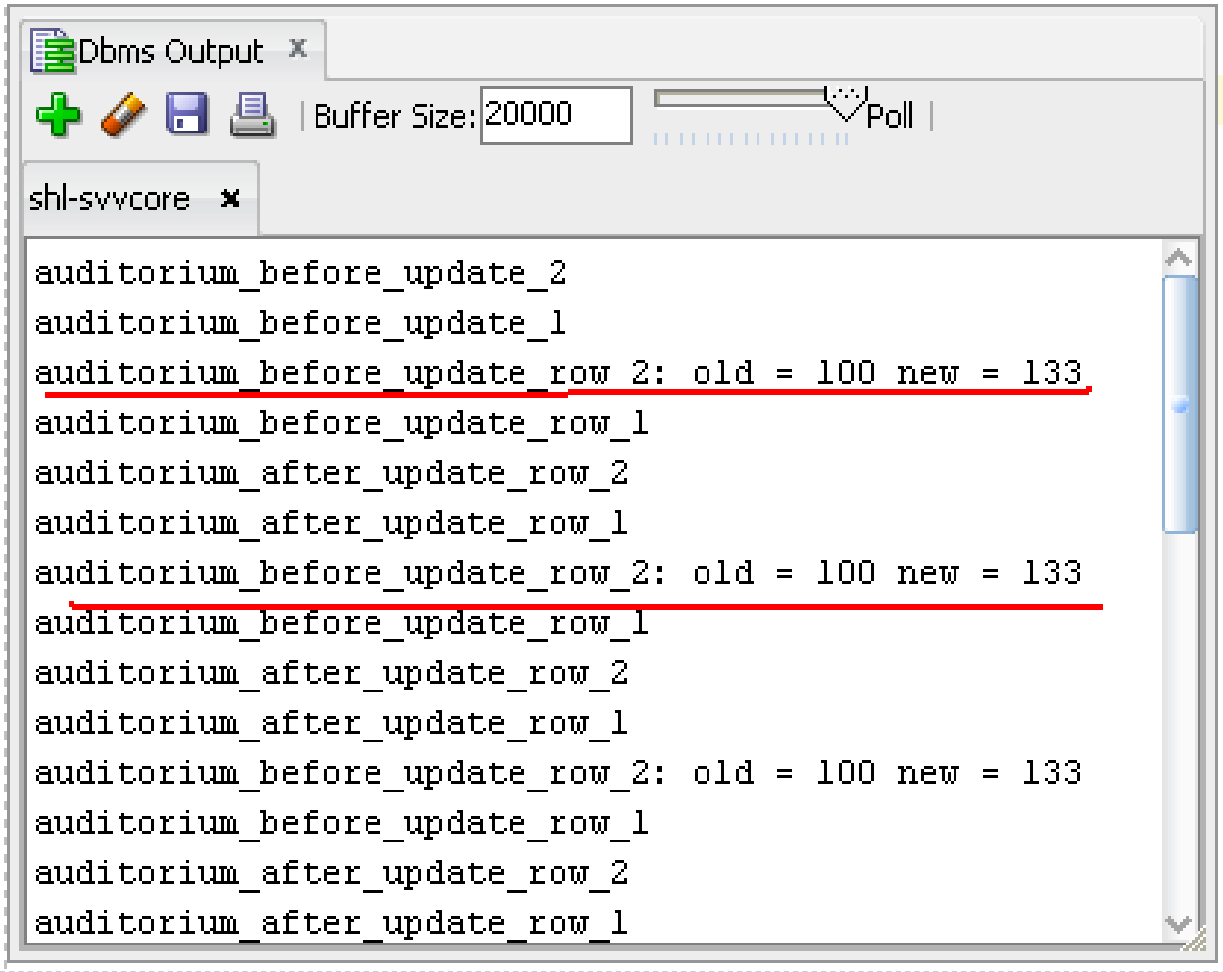
Псевдозаписи new, old

```
insert into auditorium(auditorium, auditorium type,auditorium capacity)
       values ('405-1', 'ЛК', 100);
```



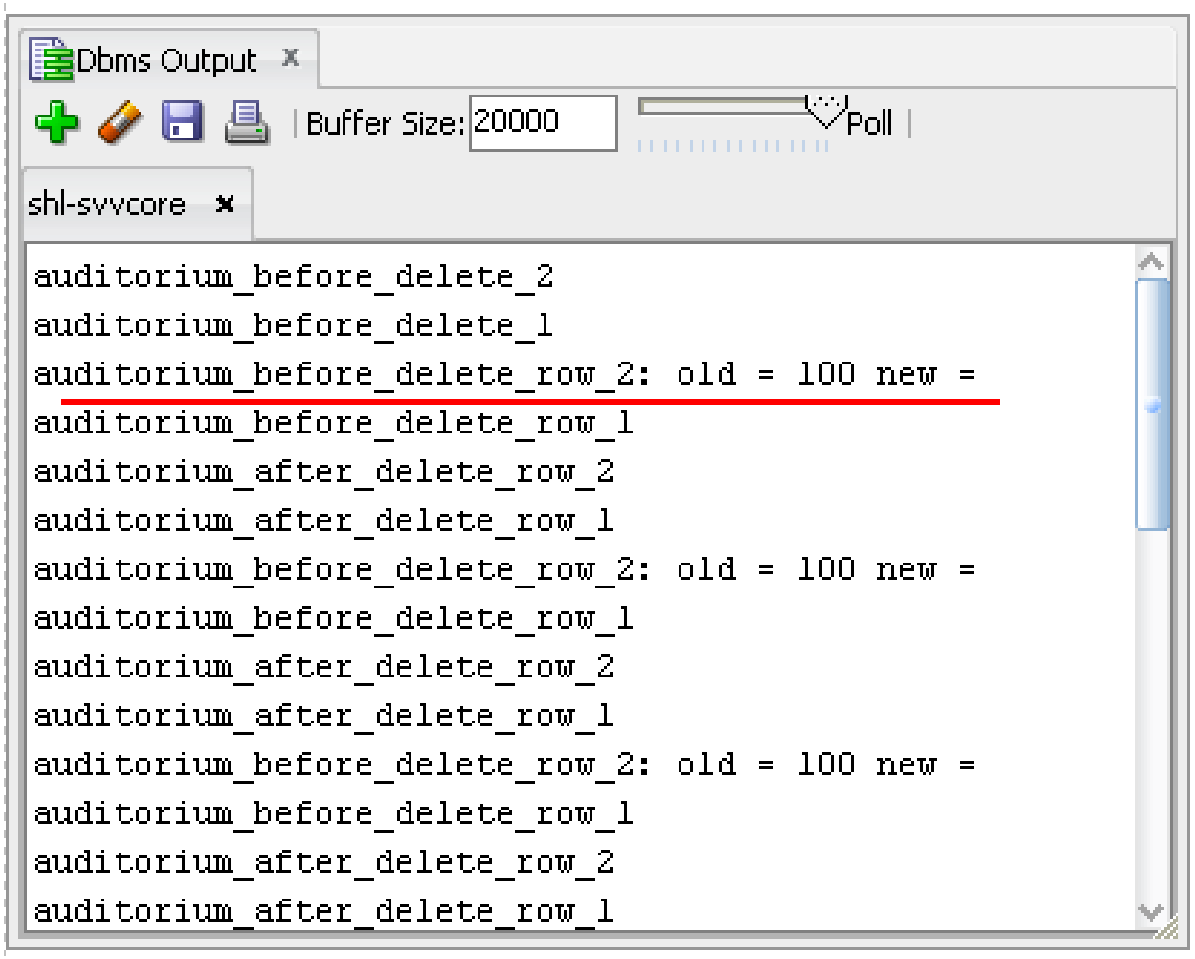
Псевдозаписи new, old

```
update auditorium set auditorium_capacity = 133;
```



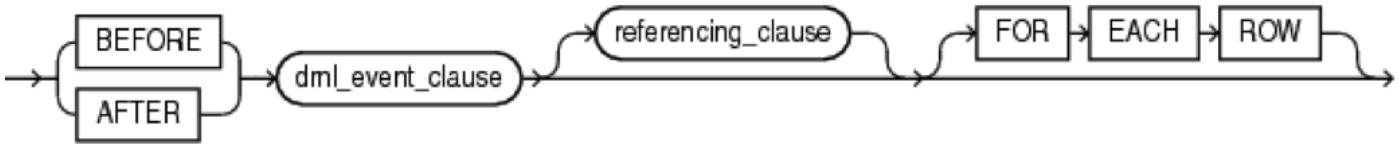
Псевдозаписи new, old

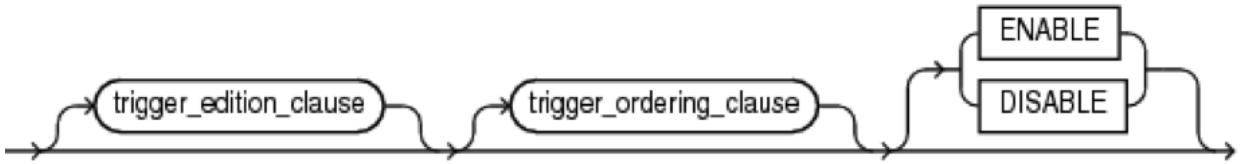
```
delete  auditorium;
```

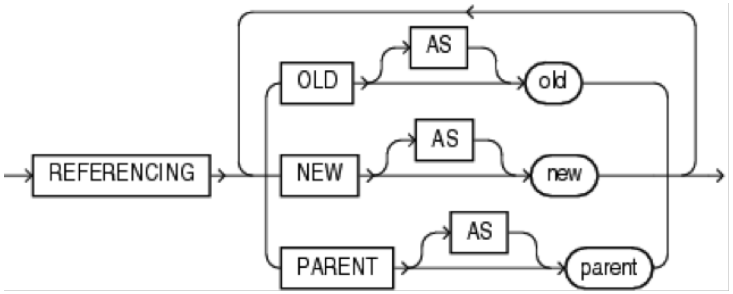


Выражение REFERENCING

- **REFERENCING** позволяет определить имена для триггерных записей, отличные от имен по умолчанию

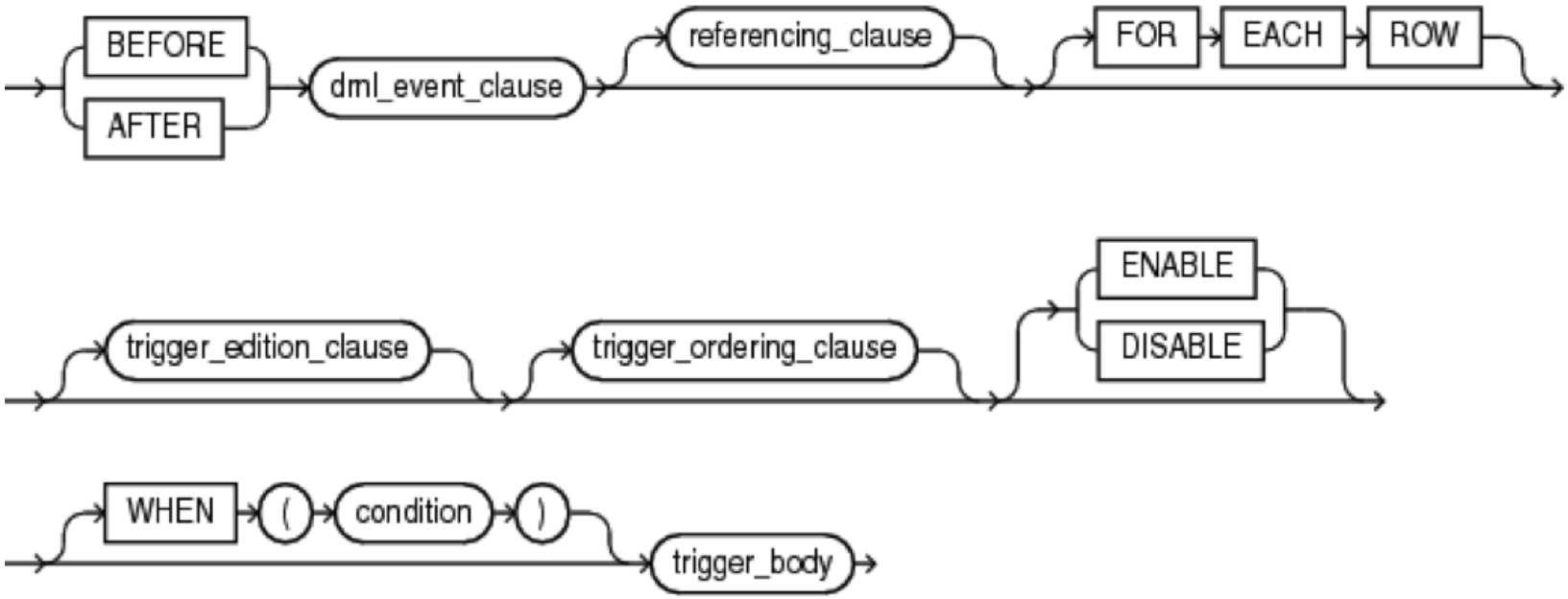






Выражение WHEN

- Выражение **WHEN** определяет условия, при которых срабатывает триггер
- Хранимые функции и объектные методы не разрешены для использования в выражении WHEN



Мутация таблиц

```
create or replace trigger tr_salary_ch
before update of salary
on employees
for each row
when (
  :new.name = 'Ivanov' and :old.salary < :new.salary
)
begin
  update employees set salary = salary *1.1
   where name <>'Ivanov';
end;
```

```
update employees set salary = 2500
 where name = 'Ivanov';
```

```
Error starting at line 28 in command:
update employees set salary = 2500
 where name = 'Ivanov'
Error report:
SQL Error: ORA-04091: таблица SYSTEM.EMPLOYEES изменяется, триггер/функция может не заметить это
ORA-06512: на  "SYSTEM.TR_SALARY_CH", line 2
ORA-04088: ошибка во время выполнения триггера 'SYSTEM.TR_SALARY_CH'
04091. 00000 -  "table %s.%s is mutating, trigger/function may not see it"
*Cause:    A trigger (or a user defined plsql function that is referenced in
           this statement) attempted to look at (or modify) a table that was
           in the middle of being modified by the statement which fired it.
*Action:   Rewrite the trigger (or function) so it does not read that table.
```

	NAME	SALARY
1	Ivanov	2000
2	Petrov	1000
3	Smirnov	1200
4	Azimof	1300

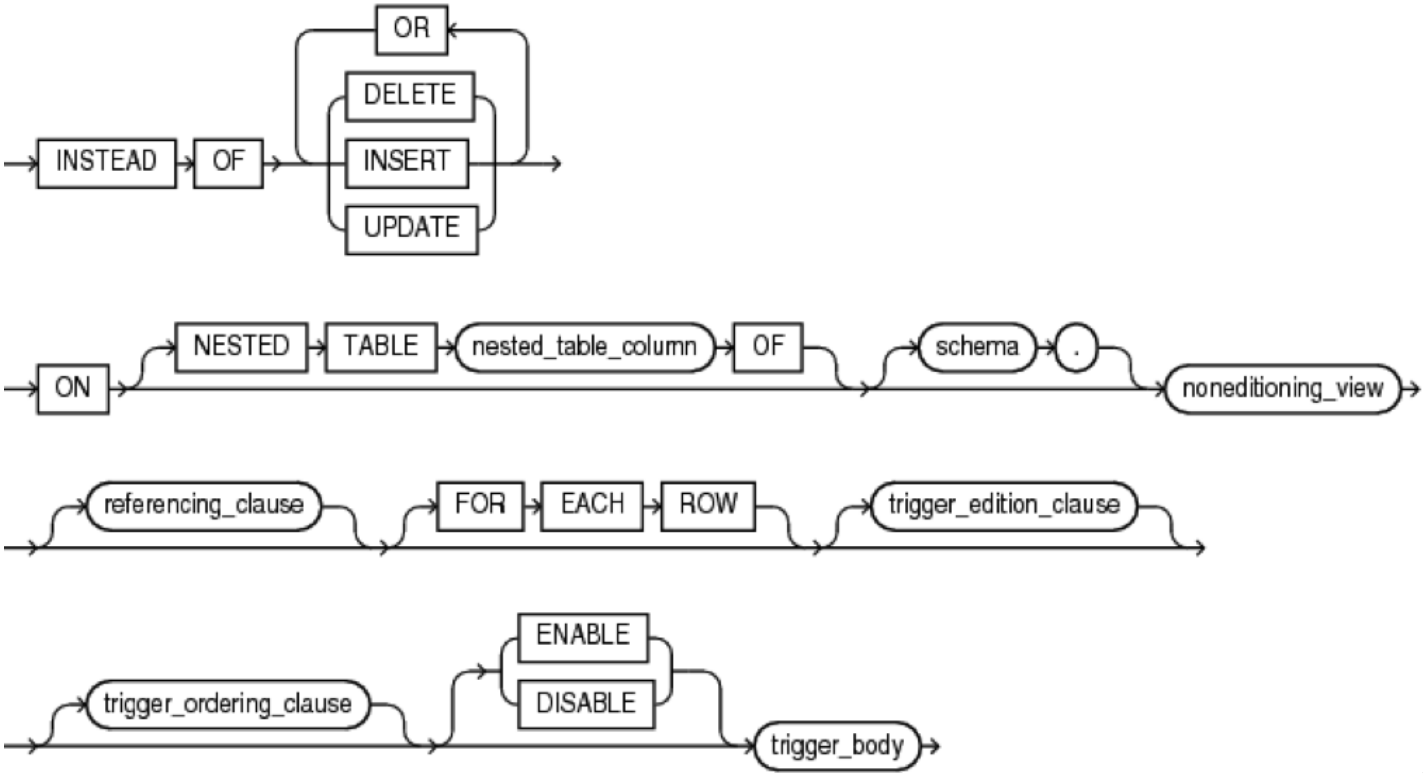
Мутация таблиц

```
create or replace trigger tr_salary_change
for update of salary on employees
referencing old as previous

compound trigger

  change_salary  boolean;

  before each row
  is
  begin
    if :new.name = 'Ivanov' and :previous.salary < :new.salary
    then
      change_salary := true;
    end if;
  end before each row;

  after statement
  is
  begin
    if change_salary then update employees
      set salary = salary *1.1 where name <>'Ivanov';
    end if;
  end after statement;
end tr_salary_change;
```

	NAME	SALARY
1	Ivanov	2500
2	Petrov	1100
3	Smirnov	1320
4	Azimof	1430

Триггеры замещения - INSTEAD OF

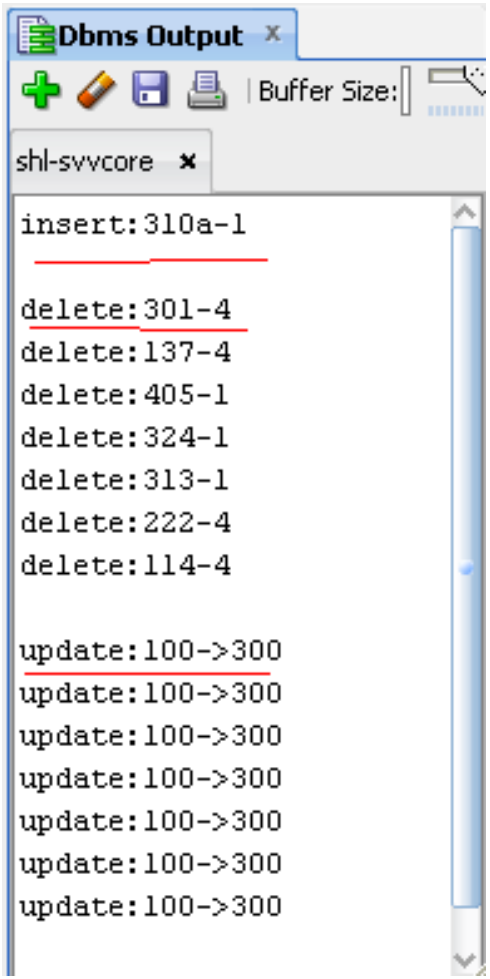
- Создаются только для представлений, для таблиц нельзя.
- Только уровня строки.



Триггеры замещения - INSTEAD OF

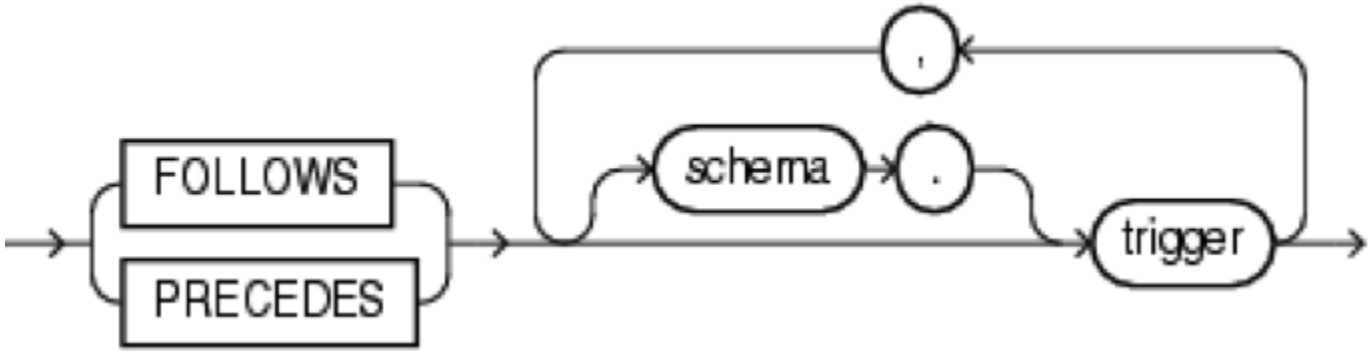
```
create view vauditorium
as select auditorium ua , auditorium_name na , auditorium_capacity ca, auditorium_type ta
   from AUDITORIUM

select * from vauditorium

create or replace  trigger trauditorium
 instead  of insert or update or delete  on vAUDITORIUM
 for each row
 begin
  if inserting  then dbms_output.put_line('insert:'||:new.ua);
  elsif updating then dbms_output.put_line('update:'||rtrim(:old.ca) ||'->'||:new.ca);
  elsif deleting then dbms_output.put_line('delete:'||:old.ua);
  end if;
 end trauditorium;


 insert into vauditorium (ua, ca)values ('310a-1',16);
 delete vauditorium;
 update vauditorium set ca = 300;
```

Триггеры замещения - INSTEAD OF



Порядок выполнения триггеров

- В каком порядке выполняются триггеры?



Включение/отключение триггеров

- Включение и отключение триггеров:
  - alter trigger { disable | enable }
- Всех для таблицы:
  - ALTER TABLE table_name { ENABLE | DISABLE } ALL TRIGGERS;
- Компиляция триггера:
  alter trigger TRIGGER_NAME compile;
- Переименование триггера

Триггеры - словарь

- dba_triggers
- dba_source
- dba_objects

Триггеры - словарь

```
select * from user_triggers;
```

TRIGGER_NAME	TRIGGER_TYPE	TRIGGERING_EVENT	TABLE_OWNER	BASE_C
AUDITORIUM_BEFORE_ROW_2	BEFORE EACH ROW	INSERT OR UPDATE OR DELETE	SVVCORE	TABLE
AUDITORIUM_BEFORE_2	BEFORE STATEMENT	INSERT OR UPDATE OR DELETE	SVVCORE	TABLE
AUDITORIUM_BEFORE_ROW_1	BEFORE EACH ROW	INSERT OR UPDATE OR DELETE	SVVCORE	TABLE
AUDITORIUM_BEFORE_1	BEFORE STATEMENT	INSERT OR UPDATE OR DELETE	SVVCORE	TABLE
AUDITORIUM_AFTER_2	AFTER STATEMENT	INSERT OR UPDATE OR DELETE	SVVCORE	TABLE
AUDITORIUM_AFTER_ROW_2	AFTER EACH ROW	INSERT OR UPDATE OR DELETE	SVVCORE	TABLE
AUDITORIUM_AFTER_1	AFTER STATEMENT	INSERT OR UPDATE OR DELETE	SVVCORE	TABLE
AUDITORIUM_AFTER_ROW_1	AFTER EACH ROW	INSERT OR UPDATE OR DELETE	SVVCORE	TABLE

Системные триггеры

- По времени срабатывания:
  - BEFORE, AFTER
- По уровню триггера:
  - SCHEMA, DATABASE
- По виду события:
  - 1)серверные события
  - 2) DDL-события
  - 3)события сбора статистики
  - 4) события аудита
  - 5) DCL-события

Триггерные события DDL уровня схемы

CREATE	Событие возникает, когда создается объект базы данных
ALTER	Событие возникает, когда выполняется команда ALTER над объектом базы данных
DROP	Событие возникает, когда выполняется команда удаления объекта DROP

- К объектам события относятся таблицы, пакеты и другие объекты базы данных, которые можно найти в системном представлении ALL_OBJECTS
- Может применяться к отдельной схеме или базе данных в целом

Триггерные события DDL уровня схемы

```
create or replace trigger tr_drop_table
before drop on SCHEMA
begin
raise_application_error(-20000, 'Do not drop table .'||ORA_DICT_OBJ_TYPE||' '||ORA_DICT_OBJ_NAME);
end;
```

```
drop table employees;
```

```
ORA-20000: Do not drop table .TABLE EMPLOYEES
```

```
truncate table employees;
```

```
table EMPLOYEES truncated.
```

Триггерные события базы данных

BEFORE GRANT AFTER GRANT	При выполнении команды grant
BEFORE REVOKE AFTER REVOKE	При выполнении команды revoke
BEFORE DDL AFTER DDL	Срабатывает на большинство команд DDL, кроме: alter database, create control file, create database
BEFORE TRUNCATE AFTER TRUNCATE	При выполнении команды truncate

Триггерные события базы данных

SERVERERROR	Событие возникает, когда сервер генерирует ошибку. Допустимы только AFTER триггеры.
LOGON	Событие возникает, когда создается соединение пользователя с базой данных. Допустимы только AFTER триггеры.
LOGOFF	Событие возникает, когда завершается соединение пользователя с базой данных. Допустимы только BEFORE триггеры.
STARTUP	Событие возникает, когда база данных открывается для работы. Допустимы только AFTER триггеры.
SHUTDOWN	Событие возникает, когда база данных закрывается. Допустимы только BEFORE триггеры.

Системные триггеры

- Все кроме LOGOFF работают в режиме автофиксации
- LOGOFF входит в транзакцию отключения
- Системный триггер может генерировать исключение RAISE

logon/logoff – триггер

```
create or replace trigger svvguest.trlogon
   after logon on  svvguest.schema
begin
 insert into svvguest.eventreg(reguser, regcomment, regdate)
        values (user,'logon', sysdate);
-- dbms_output.put_line('svvguest.trlogon');
end trlogon;



create or replace trigger svvguest.trlogoff
before  logoff on  svvguest.schema
begin
   insert into svvguest.eventreg(reguser, regcomment, regdate)
        values (user,'logoff', sysdate);
 -- dbms_output.put_line('svvguest.trlogoff');
end trlogoff;
```

logon/logoff – триггер

```
select * from SVVGUEST.eventreg
```

REGUSER	REGCOMMENT	REGDATE
SVVGUEST	logon	10.03.11
SVVGUEST	logon	10.03.11
SVVGUEST	logon	10.03.11
SVVGUEST	logoff	10.03.11
SVVGUEST	logon	10.03.11
SVVGUEST	logoff	10.03.11
SVVGUEST	logoff	10.03.11
SVVGUEST	logon	10.03.11

Вопросы?