

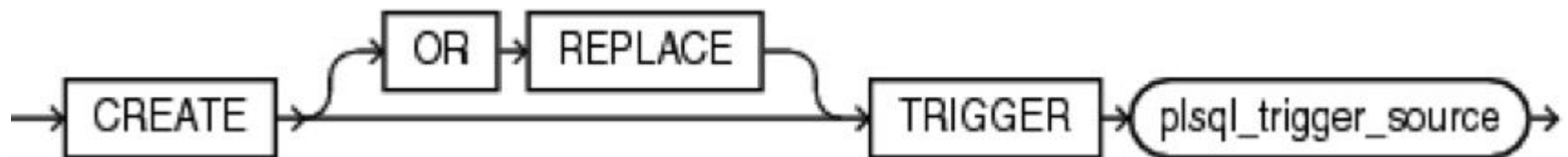
ORACLE 12c

PL/SQL Триггеры

Лекция 13

Триггеры

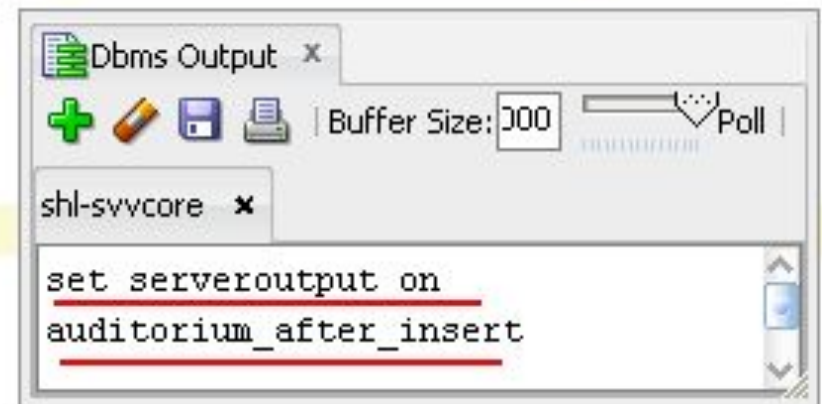
- ▣ **Триггер** – особый вид процедур, которые срабатывают по запускающему их событию



Создание триггера на вставку

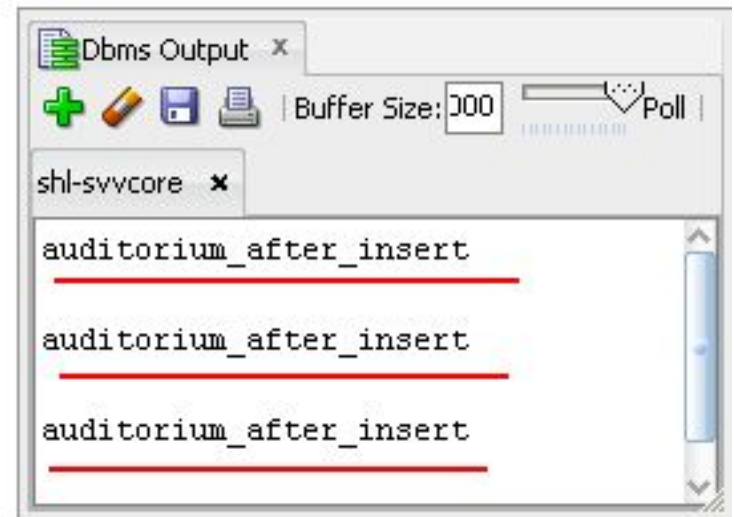
```
create or replace trigger auditorium_after_insert  
after insert on auditorium  
begin  
    dbms_output.put_line('auditorium_after_insert');  
end;
```

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



Выполнение триггера

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



Триггер на обновление

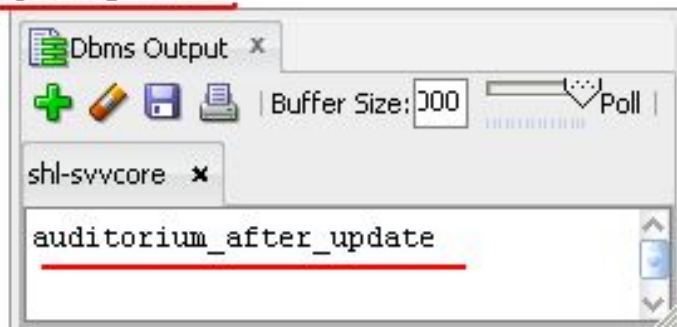
```
create or replace trigger auditorium after update  
after update on auditorium  
begin  
    dbms_output.put_line('auditorium_after_update');  
end;
```

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

COUNT(*)
4

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

4 rows updated

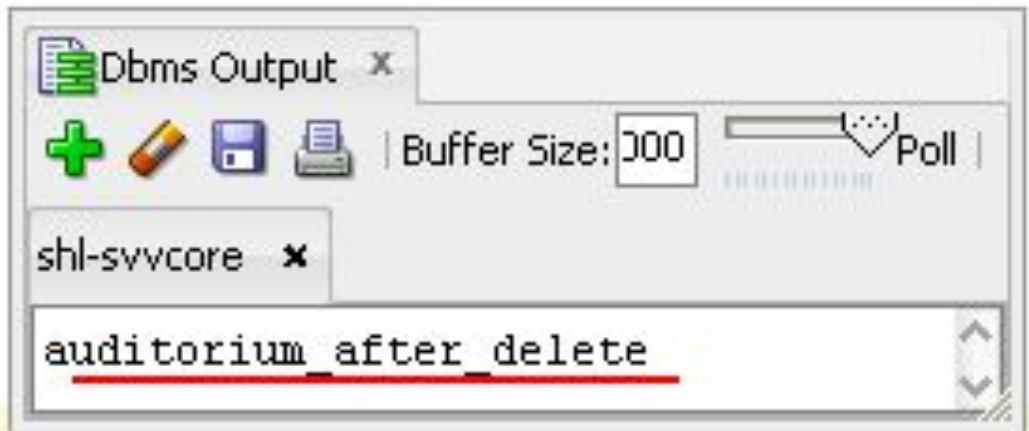


Триггер на удаление

```
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
```



Применение триггеров

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



Триггеры

- DML-триггеры
- Системные триггеры



Триггерные события DML

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



Привилегии

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



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 svucore;
```

Привилегии предоставлены.



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/ROLLBACK (исключение - только, если в теле триггера есть автономная транзакция)
- Может выдавать RAISE_APPLICATION_ERROR



Транзакции

- Основное назначение транзакции – переводить БД из одного согласованного состояния в другое
- Свойства транзакций:
 - Неделимость – atomicity
 - Согласованность – consistency
 - Изолированность – isolation
 - Продолжительность – durability
- COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION
- Блоки PL/SQL и транзакции



Транзакции

- Распределенные транзакции
- Транзакции и данные повтора
- Транзакции и данные отката
- Автономные транзакции



DML-триггеры

□ Время события:

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



Порядок выполнения DML-триггеров

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



Уровни триггеров

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



Количество триггеров

- Всего типов триггеров = 28
- (7 комбинаций операторов) × 2 момента × 2 уровня.
- для таблицы может быть любое количество триггеров

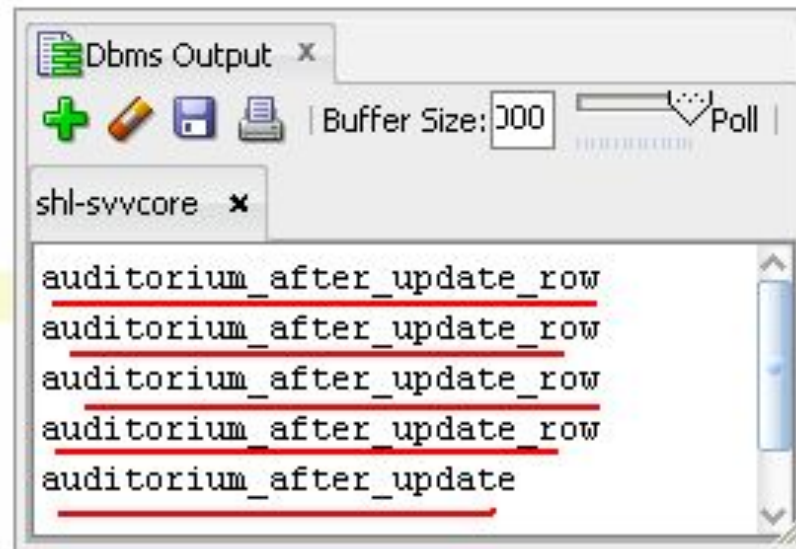


Триггеры for each row

```
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;
```

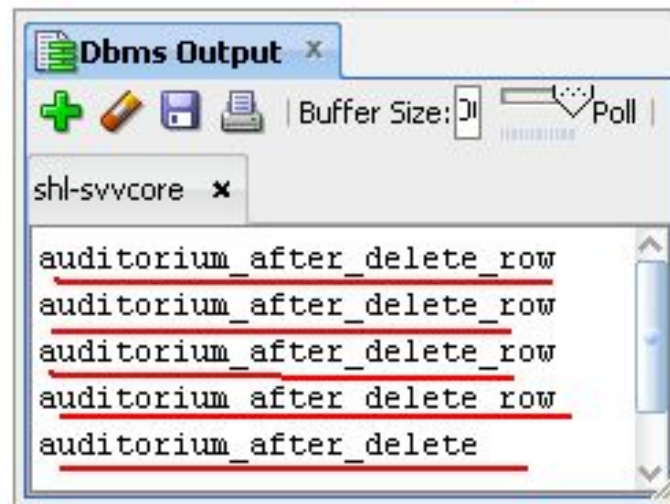
4 rows updated



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_1  
after insert or update or delete on auditorium  
begin  
  if inserting then  
    dbms_output.put_line('auditorium_after_insert_1');  
  elsif updating then  
    dbms_output.put_line('auditorium_after_update_1');  
  elsif deleting then  
    dbms_output.put_line('auditorium_after_delete_1');  
  end if;  
end;
```



Предикаты триггера

□ Для триггера for each row

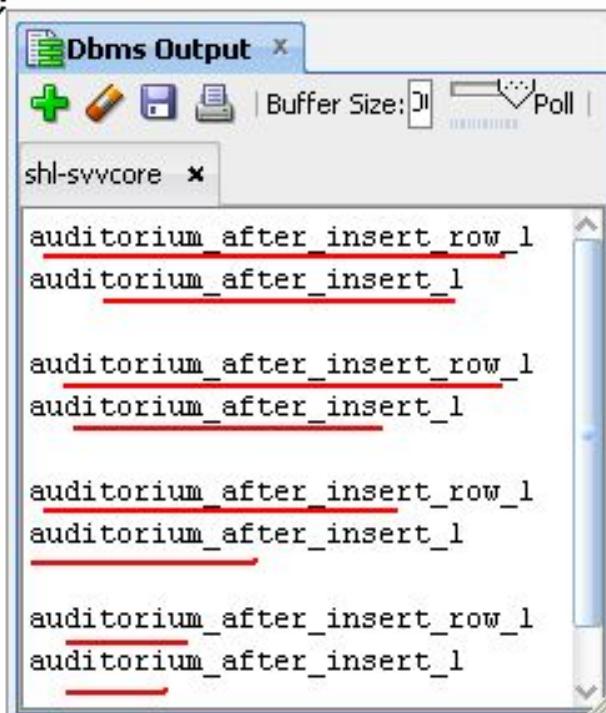
```
drop trigger auditorium_after_update_row;  
drop trigger auditorium_after_delete_row;
```

```
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;
```



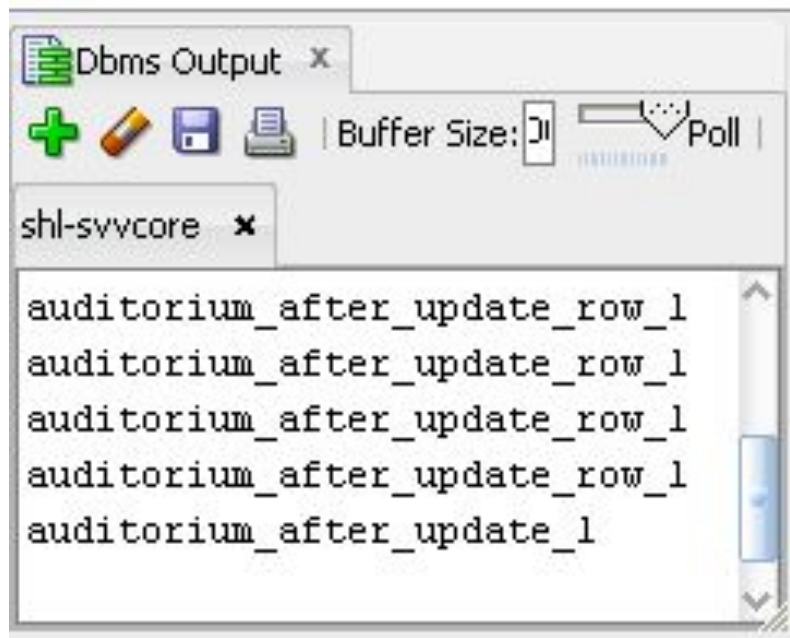
Применение набора триггеров

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



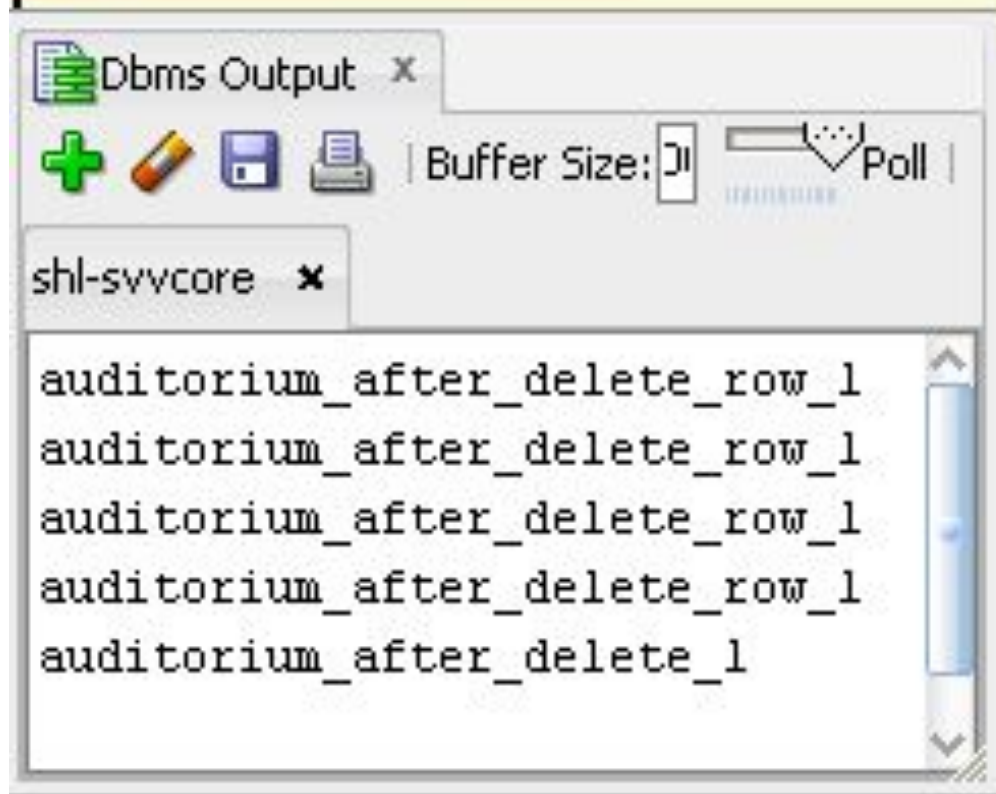
Применение набора триггеров

update auditorium set auditorium_capacity = 120;



Применение набора триггеров

```
delete auditorium;
```



Порядок выполнения триггеров

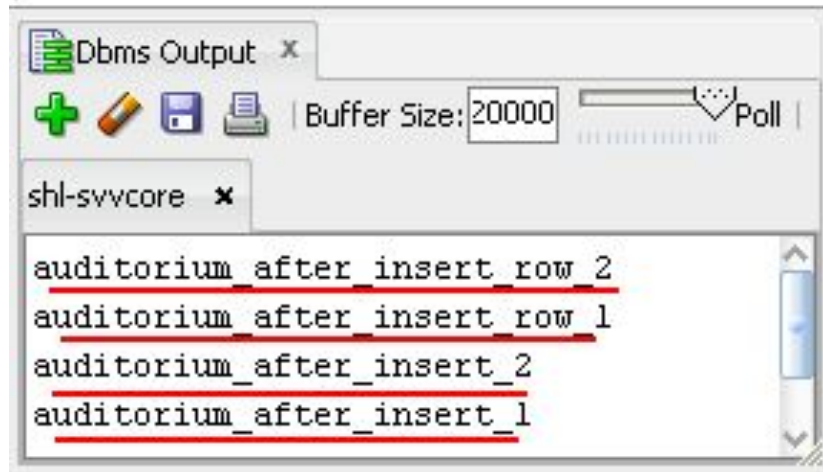
```
create or replace trigger auditorium_after_2  
after insert or update or delete on auditorium  
begin  
  if inserting then  
    dbms_output.put_line('auditorium_after_insert_2');  
  elsif updating then  
    dbms_output.put_line('auditorium_after_update_2');  
  elsif deleting then  
    dbms_output.put_line('auditorium_after_delete_2');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



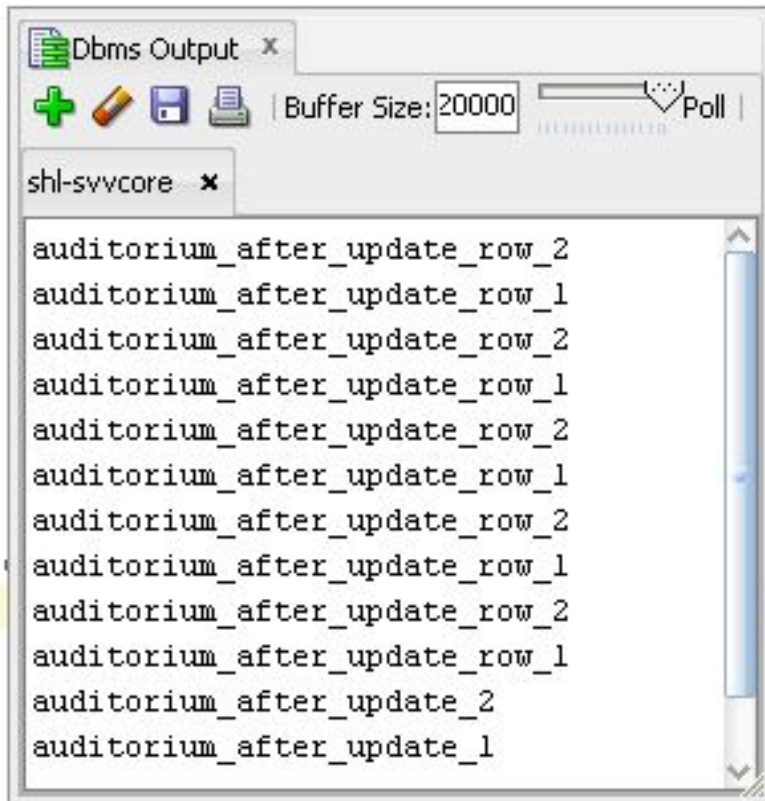
Порядок выполнения триггеров

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



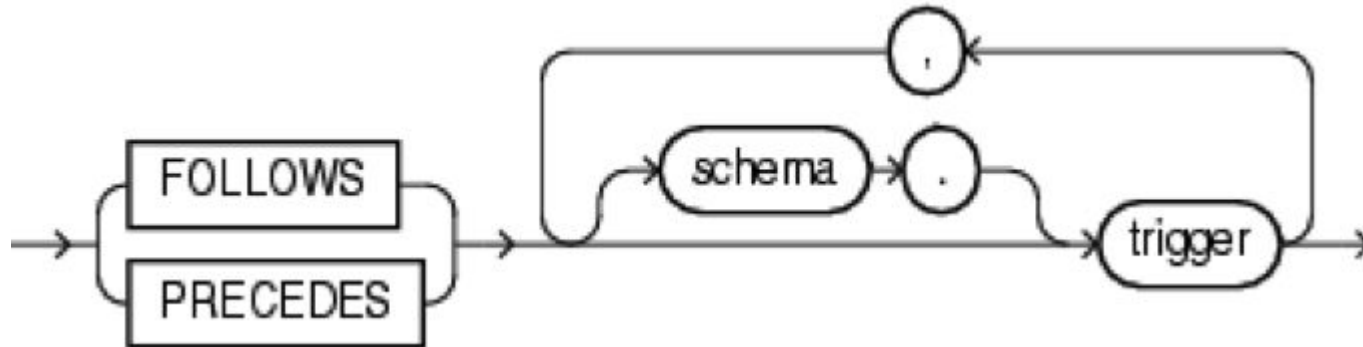
Порядок выполнения триггеров

update auditorium set auditorium_capacity = 120;



Порядок выполнения триггеров

- В каком порядке выполняются триггеры?



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

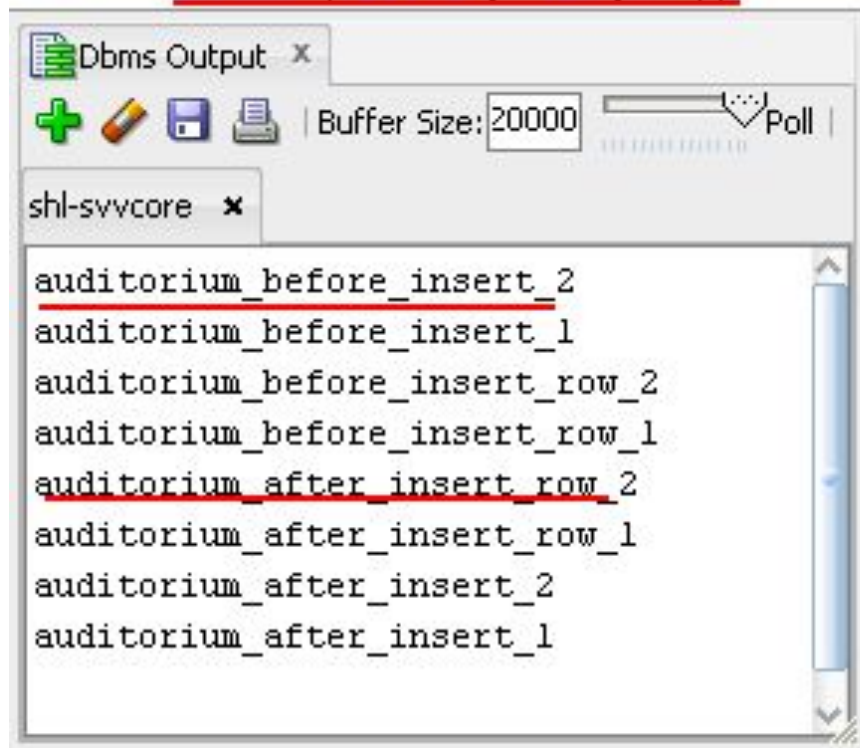
```
create or replace trigger auditorium_before_2  
before insert or update or delete on auditorium  
begin  
    if inserting then  
        dbms_output.put_line('auditorium_before_insert_2');  
    elsif updating then  
        dbms_output.put_line('auditorium_before_update_2');  
    elsif deleting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



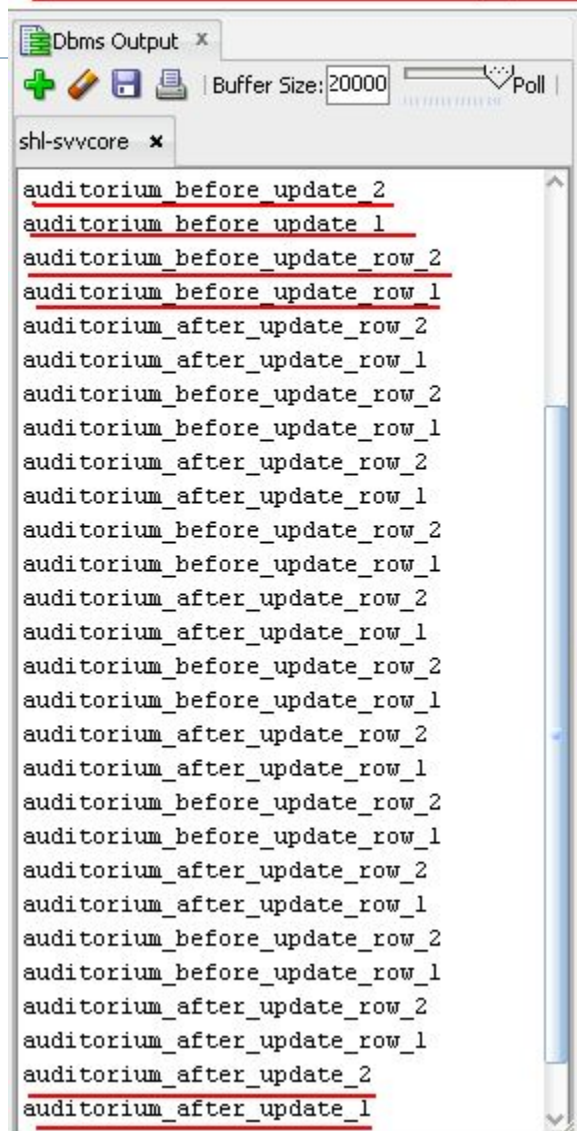
Before - триггеры

```
insert into auditorium(auditorium, auditorium type,auditorium capacity)  
values ('137-4', 'JK', 60);
```



Before - триггеры

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



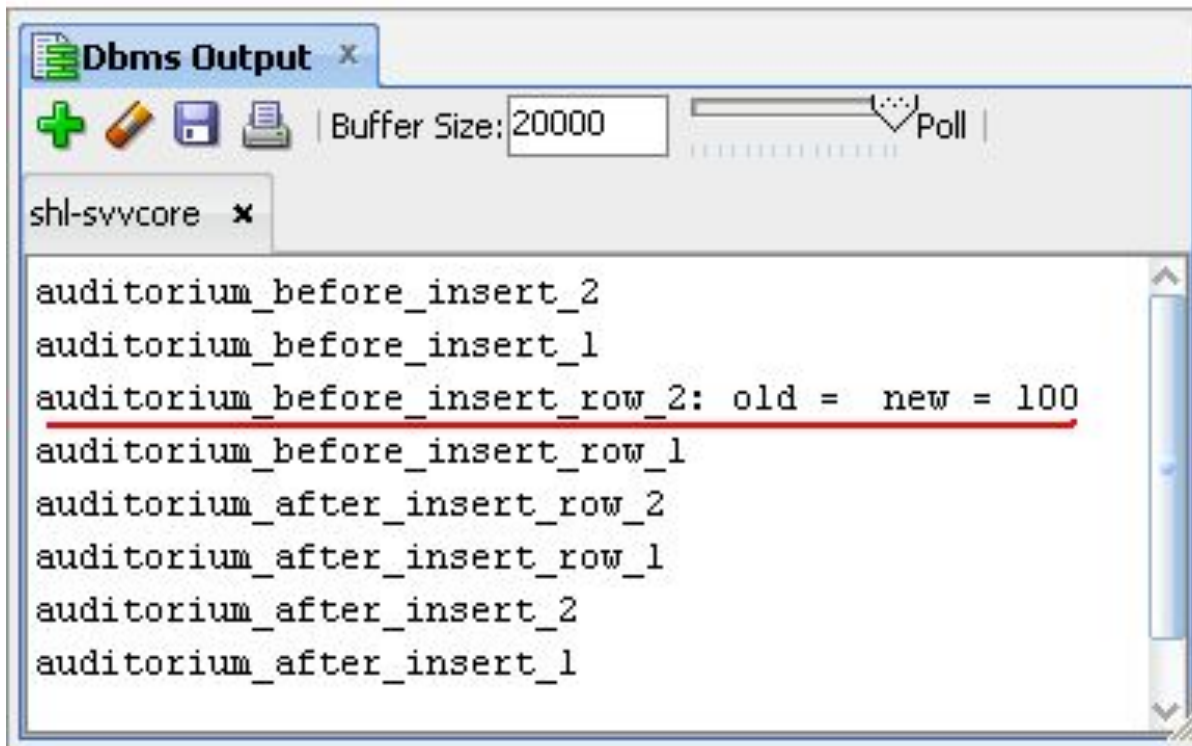
```
shl-svvcare x
auditorium_before_update_2
auditorium_before_update_1
auditorium_before_update_row_2
auditorium_before_update_row_1
auditorium_after_update_row_2
auditorium_after_update_row_1
auditorium_before_update_row_2
auditorium_before_update_row_1
auditorium_after_update_row_2
auditorium_after_update_row_1
auditorium_before_update_row_2
auditorium_before_update_row_1
auditorium_after_update_row_2
auditorium_after_update_row_1
auditorium_before_update_row_2
auditorium_before_update_row_1
auditorium_after_update_row_2
auditorium_after_update_row_1
auditorium_before_update_row_2
auditorium_before_update_row_1
auditorium_after_update_row_2
auditorium_after_update_row_1
auditorium_before_update_row_2
auditorium_before_update_row_1
auditorium_after_update_row_2
auditorium_after_update_row_1
auditorium_after_update_2
auditorium_after_update_1
```

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', 'ЖК', 100);
```



Псевдозаписи new, old

```
delete auditorium;
```

Dbms Output x



Buffer Size: 20000



Poll

shl-svvcare x

```
auditorium_before_delete_2  
auditorium_before_delete_1  
auditorium_before_delete_row_2: old = 100 new =  
auditorium_before_delete_row_1  
auditorium_after_delete_row_2  
auditorium_after_delete_row_1  
auditorium_before_delete_row_2: old = 100 new =  
auditorium_before_delete_row_1  
auditorium_after_delete_row_2  
auditorium_after_delete_row_1  
auditorium_before_delete_row_2: old = 100 new =  
auditorium_before_delete_row_1  
auditorium_after_delete_row_2  
auditorium_after_delete_row_1
```

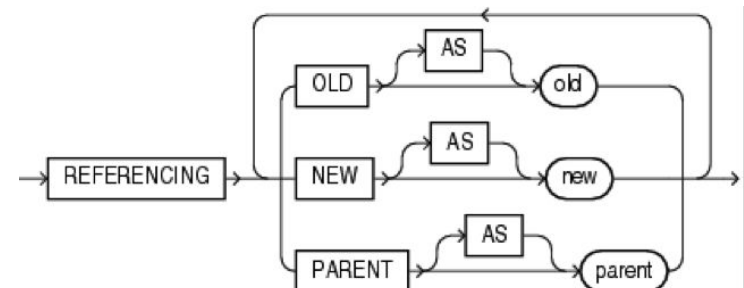
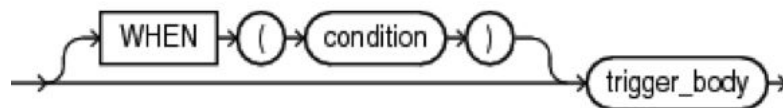
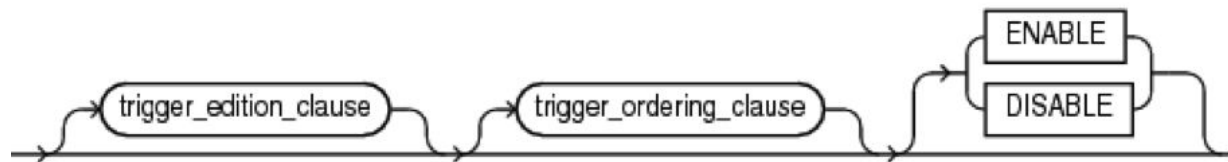
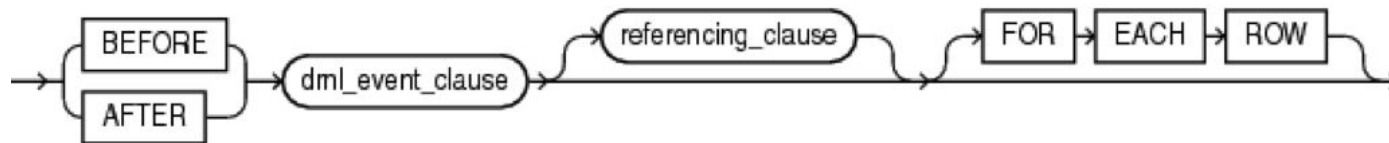
Псевдозаписи new, old

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



Выражение REFERENCING

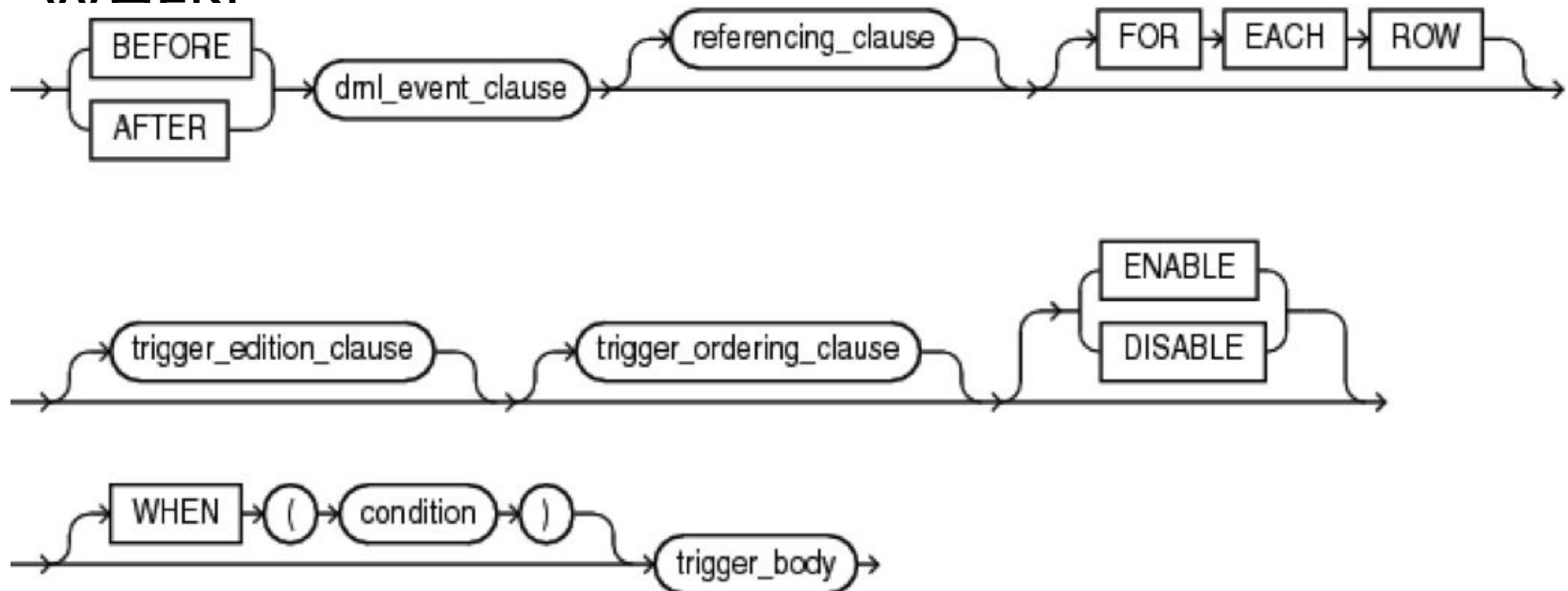
- **REFERENCING** позволяет определить имена для триггерных записей, отличные от имен по



Выражение WHEN

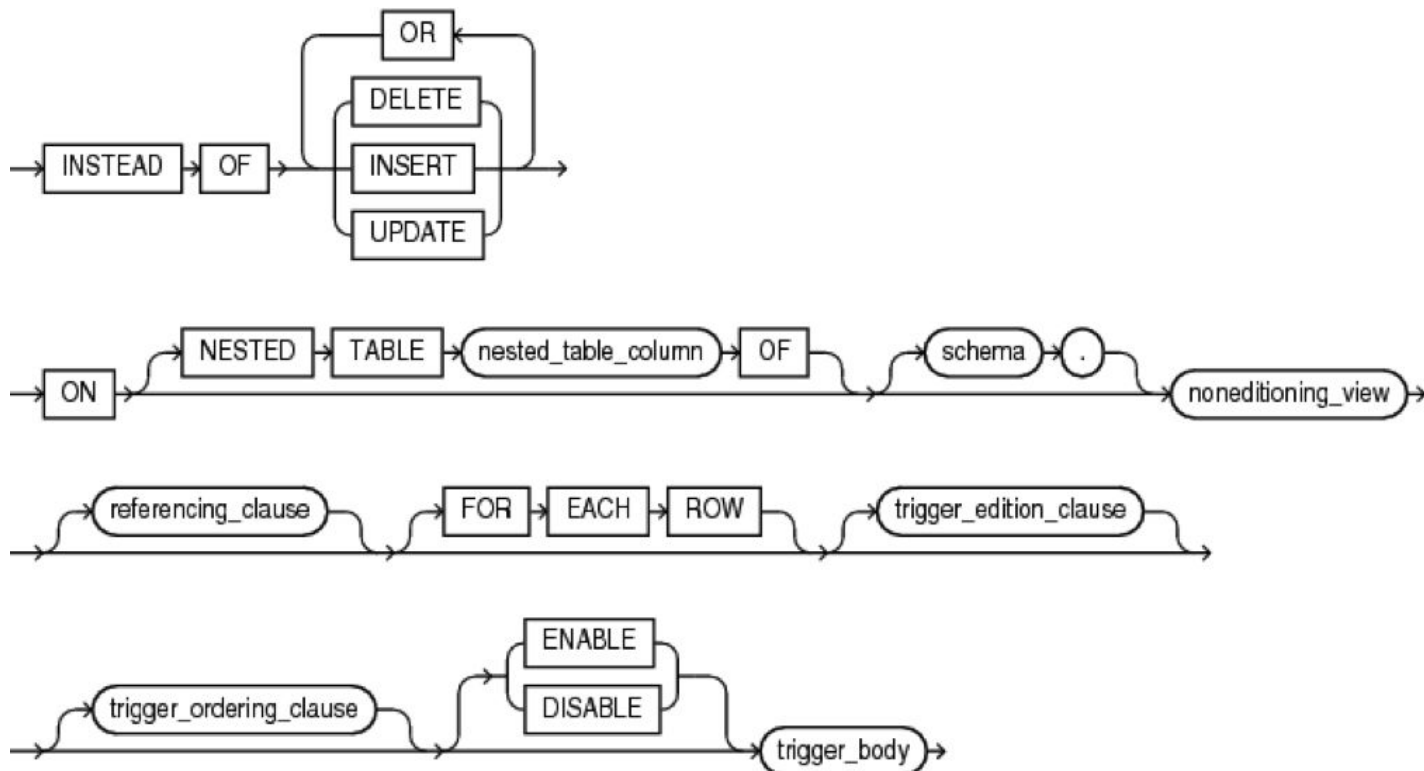
- Выражение **WHEN** определяет условия, при которых срабатывает триггер.
- Хранимые функции и объектные методы не разрешены для использования в выражении

WHEN



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
```

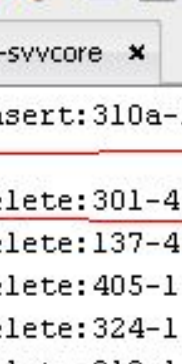
```
select * from vauditorium
```

```
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



The screenshot shows a 'Dbms Output' window with a toolbar containing icons for adding, editing, saving, and printing. The 'Buffer Size' is set to 1000. The output text is as follows:

```

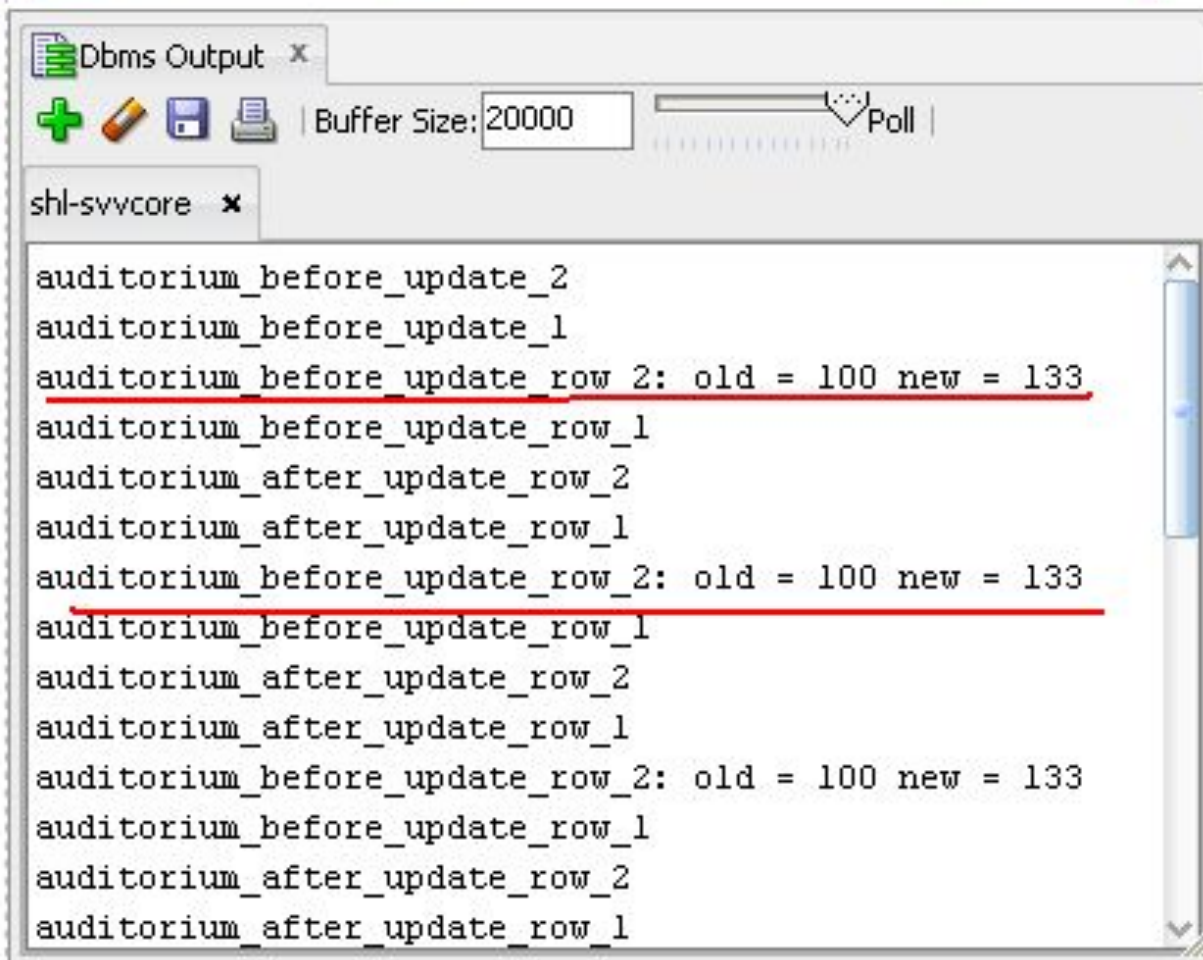
shl-svvcore x
insert: 310a-1
delete: 301-4
delete: 137-4
delete: 405-1
delete: 324-1
delete: 313-1
delete: 222-4
delete: 114-4

update: 100->300
update: 100->300
update: 100->300
update: 100->300
update: 100->300
update: 100->300
update: 100->300

```

Псевдозаписи new, old

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



The screenshot shows a Dbms Output window with a tab labeled 'shl-svvcare'. The window has a toolbar with icons for adding, editing, saving, and printing, along with a 'Buffer Size' field set to 20000 and a 'Poll' button. The output text is as follows:

```
auditorium_before_update_2  
auditorium_before_update_1  
auditorium_before_update_row_2: old = 100 new = 133  
auditorium_before_update_row_1  
auditorium_after_update_row_2  
auditorium_after_update_row_1  
auditorium_before_update_row_2: old = 100 new = 133  
auditorium_before_update_row_1  
auditorium_after_update_row_2  
auditorium_after_update_row_1  
auditorium_before_update_row_2: old = 100 new = 133  
auditorium_before_update_row_1  
auditorium_after_update_row_2  
auditorium_after_update_row_1
```

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



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



Триггеры - словарь

- ▣ **dba_triggers** – информация о триггерах
- ▣ **dba_source** – код тела триггера
- ▣ **dba_objects** – валидность триггера



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:
 - DATABASE, SCHEMA
- По виду события:
 - 1)серверные события;
 - 2) DDL-события;
 - 3)события сбора статистики;
 - 4) события аудита;
 - 5) DCL-события.



Триггерные события DDL

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 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.

Триггерные события базы данных

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



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?

