Белорусский государственный университет информатики и радиоэлектроники
Кафедра информатики

Отчет
Лабораторная работа № 2
Модели данных и системы управления базами данных

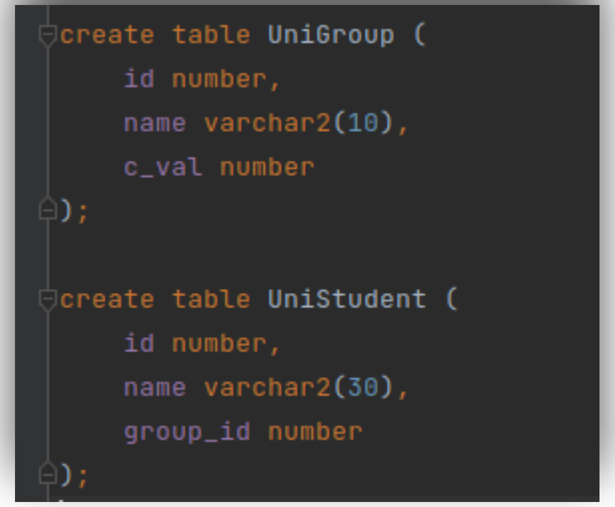
Выполнил студент гр. 953501
Кореневский С. А.

Проверил:
Чащин С. В.

Минск 2022

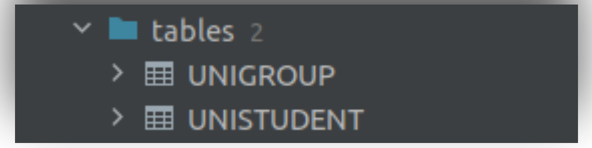
Задание 1

Построить две таблицы STUDENTS и GROUPS реализующих соответственно справочник студентов и справочник групп



```
create table UniGroup (
    id number,
    name varchar2(10),
    c_val number
);

create table UniStudent (
    id number,
    name varchar2(30),
    group_id number
);
```

Результат:

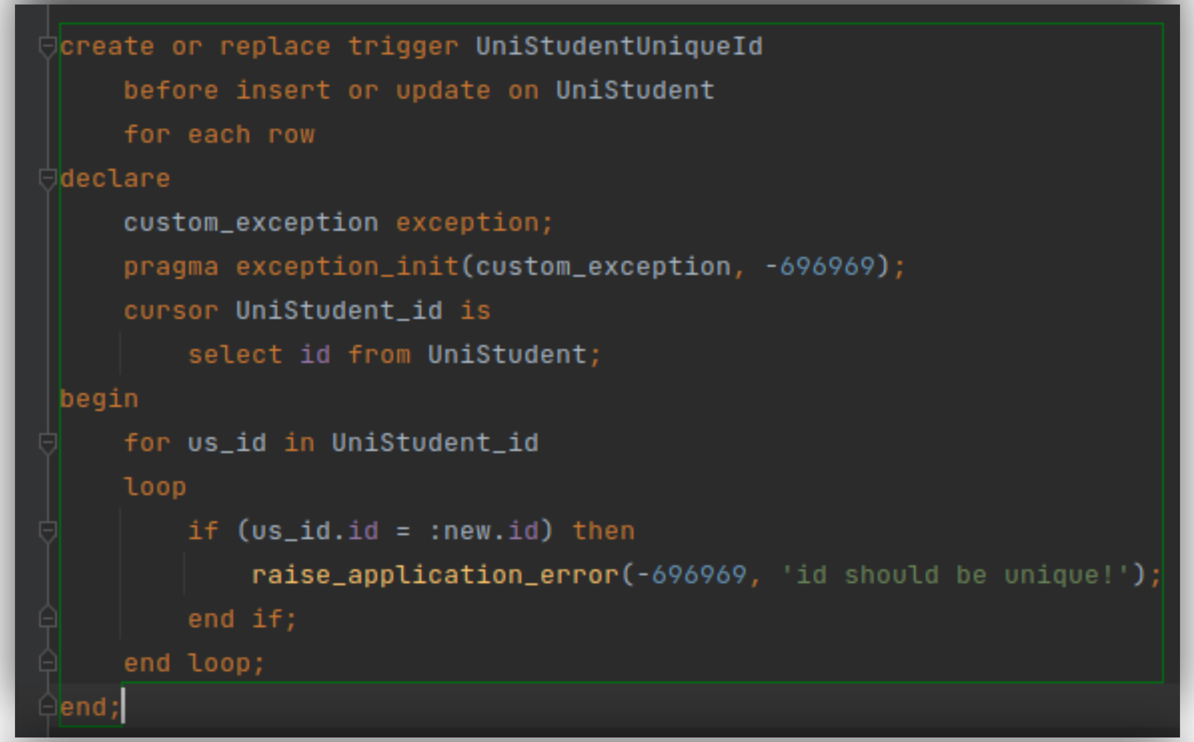


```
tables 2
  UNIGROUP
  UNISTUDENT
```

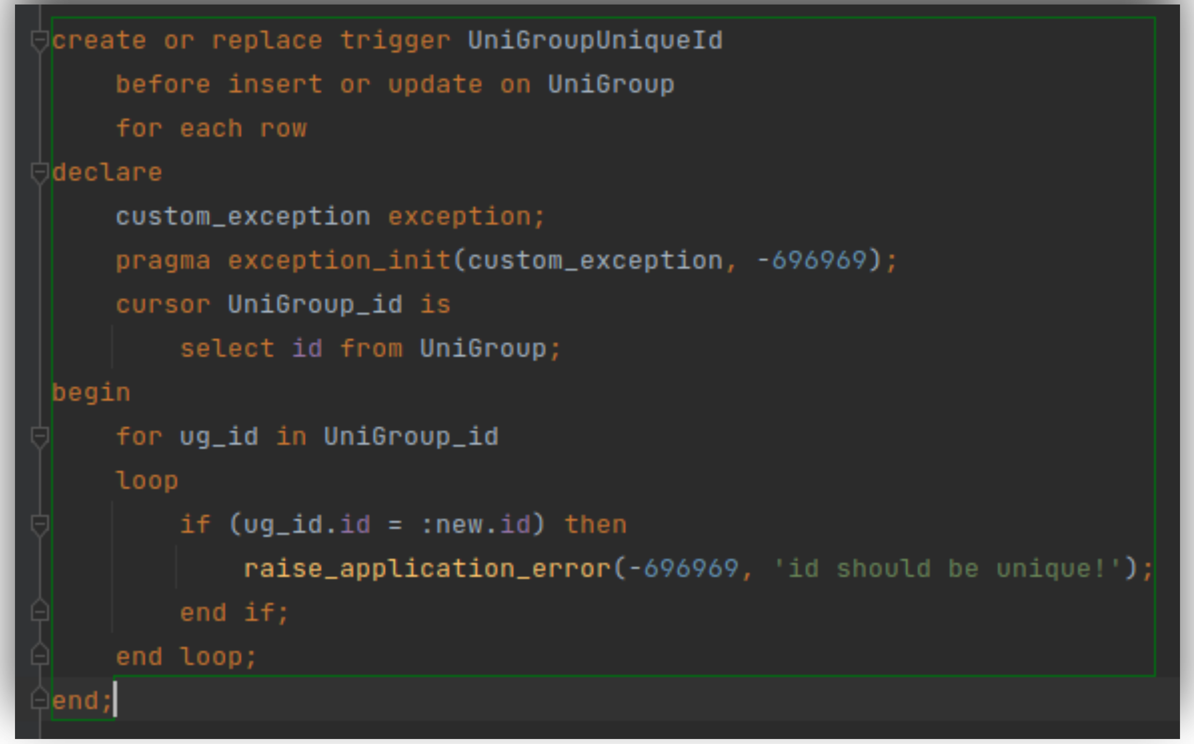
Задание 2

Реализовать триггеры для таблиц задания 1 проверку целостности (проверка на уникальность полей ID), генерацию автоинкрементного ключа и проверку уникальности для поля GROUP.NAME

Проверка уникальности id:

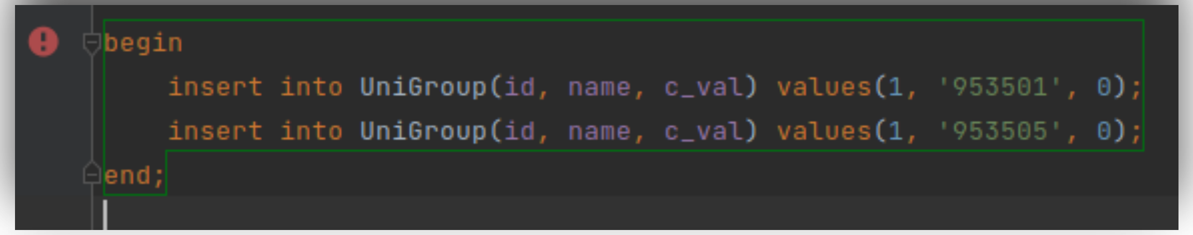


```
create or replace trigger UniStudentUniqueId
    before insert or update on UniStudent
    for each row
declare
    custom_exception exception;
    pragma exception_init(custom_exception, -696969);
    cursor UniStudent_id is
        select id from UniStudent;
begin
    for us_id in UniStudent_id
    loop
        if (us_id.id = :new.id) then
            raise_application_error(-696969, 'id should be unique!');
        end if;
    end loop;
end;
```

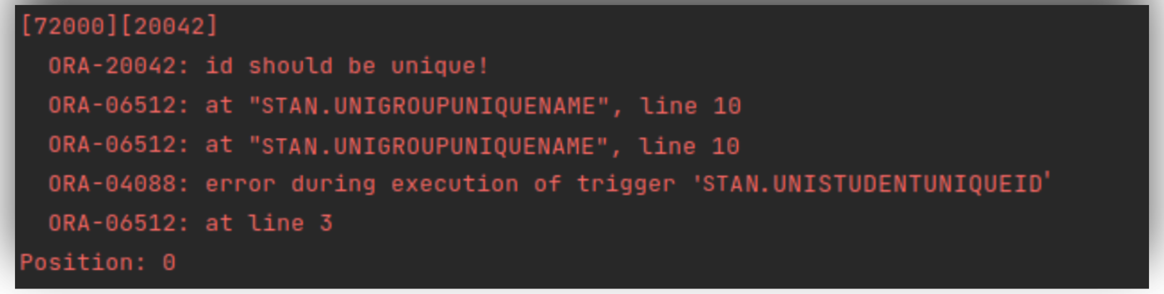


```
create or replace trigger UniGroupUniqueId
    before insert or update on UniGroup
    for each row
declare
    custom_exception exception;
    pragma exception_init(custom_exception, -696969);
    cursor UniGroup_id is
        select id from UniGroup;
begin
    for ug_id in UniGroup_id
    loop
        if (ug_id.id = :new.id) then
            raise_application_error(-696969, 'id should be unique!');
        end if;
    end loop;
end;
```

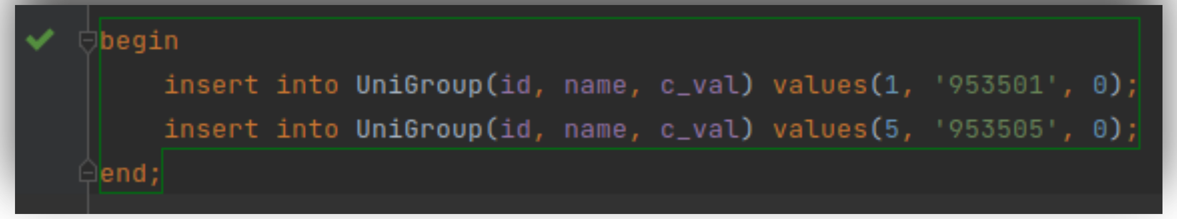
Результат:



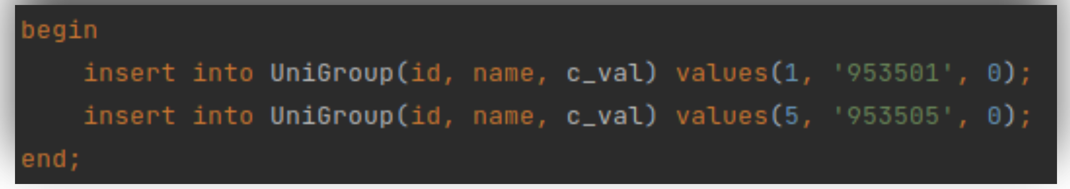
```
begin
    insert into UniGroup(id, name, c_val) values(1, '953501', 0);
    insert into UniGroup(id, name, c_val) values(1, '953505', 0);
end;
```



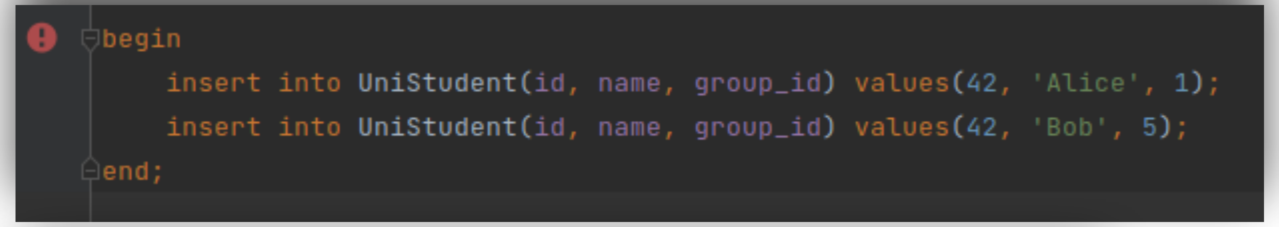
```
[72000][20042]
  ORA-20042: id should be unique!
  ORA-06512: at "STAN.UNIGROUPUNIQUENAME", line 10
  ORA-06512: at "STAN.UNIGROUPUNIQUENAME", line 10
  ORA-04088: error during execution of trigger 'STAN.UNISTUDENTUNIQUEID'
  ORA-06512: at line 3
Position: 0
```



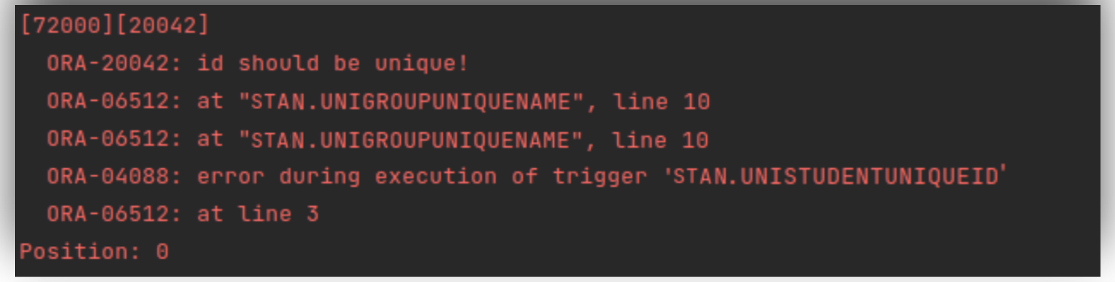
```
begin
    insert into UniGroup(id, name, c_val) values(1, '953501', 0);
    insert into UniGroup(id, name, c_val) values(5, '953505', 0);
end;
```



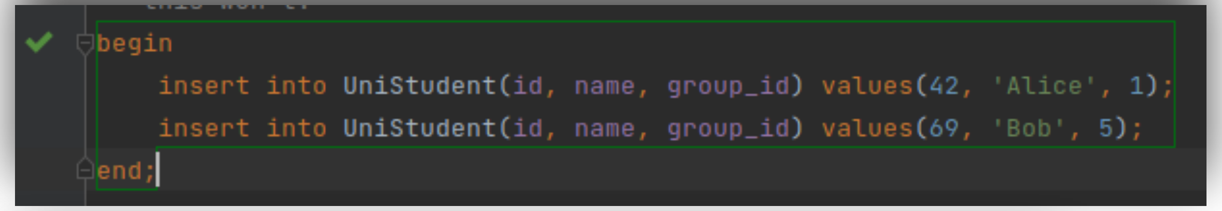
```
begin
    insert into UniGroup(id, name, c_val) values(1, '953501', 0);
    insert into UniGroup(id, name, c_val) values(5, '953505', 0);
end;
```



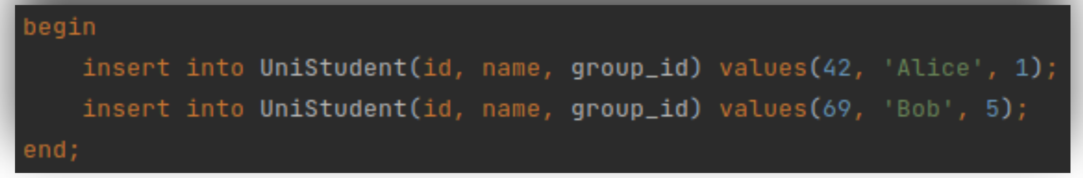
```
begin
    insert into UniStudent(id, name, group_id) values(42, 'Alice', 1);
    insert into UniStudent(id, name, group_id) values(42, 'Bob', 5);
end;
```



```
[72000][20042]
  ORA-20042: id should be unique!
  ORA-06512: at "STAN.UNIGROUPUNIQUENAME", line 10
  ORA-06512: at "STAN.UNIGROUPUNIQUENAME", line 10
  ORA-04088: error during execution of trigger 'STAN.UNISTUDENTUNIQUEID'
  ORA-06512: at line 3
Position: 0
```

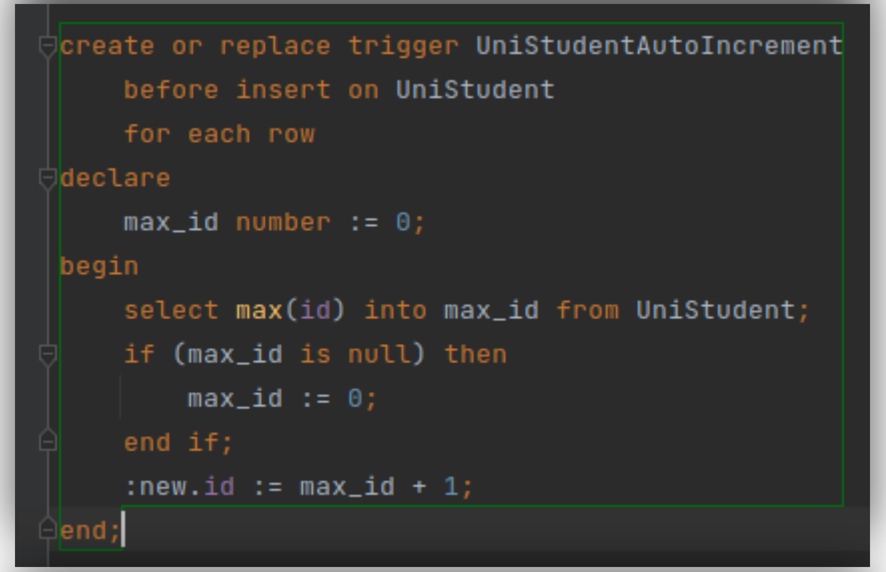


```
begin
    insert into UniStudent(id, name, group_id) values(42, 'Alice', 1);
    insert into UniStudent(id, name, group_id) values(69, 'Bob', 5);
end;
```

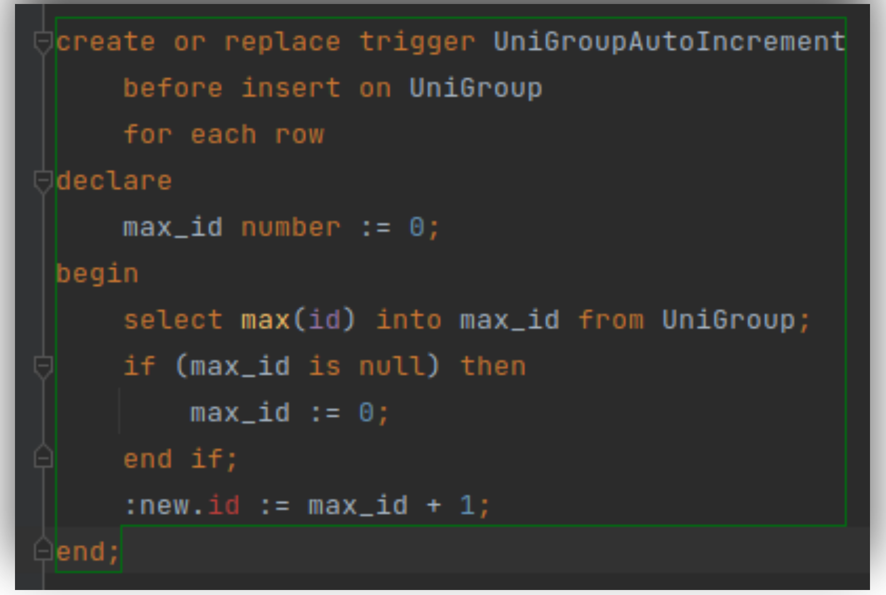


```
begin
    insert into UniStudent(id, name, group_id) values(42, 'Alice', 1);
    insert into UniStudent(id, name, group_id) values(69, 'Bob', 5);
end;
```

Автоинкрементирование id:

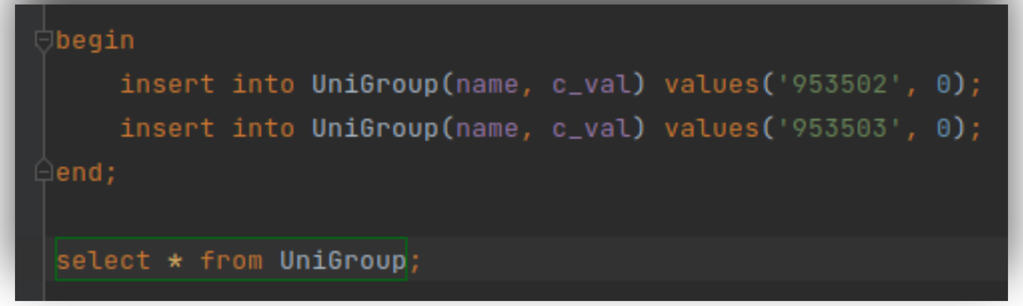


```
create or replace trigger UniStudentAutoIncrement
    before insert on UniStudent
    for each row
declare
    max_id number := 0;
begin
    select max(id) into max_id from UniStudent;
    if (max_id is null) then
        max_id := 0;
    end if;
    :new.id := max_id + 1;
end;
```

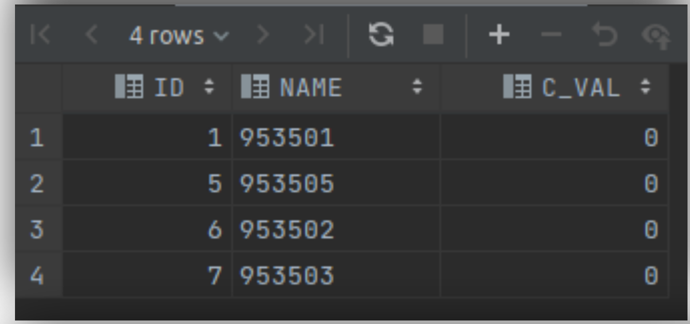


```
create or replace trigger UniGroupAutoIncrement
    before insert on UniGroup
    for each row
declare
    max_id number := 0;
begin
    select max(id) into max_id from UniGroup;
    if (max_id is null) then
        max_id := 0;
    end if;
    :new.id := max_id + 1;
end;
```

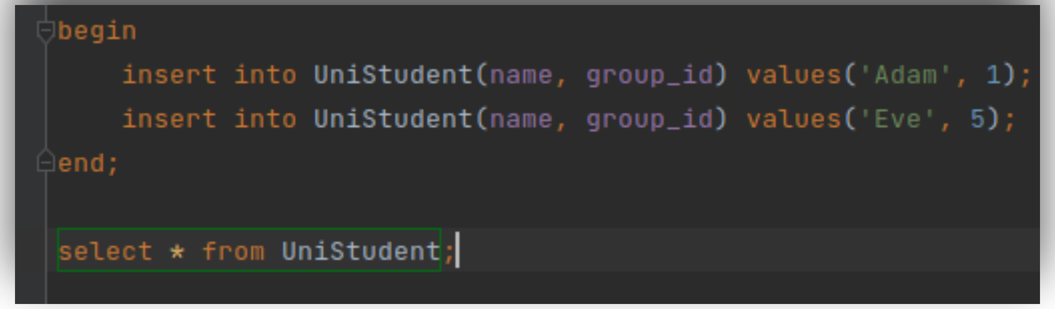
Результат:



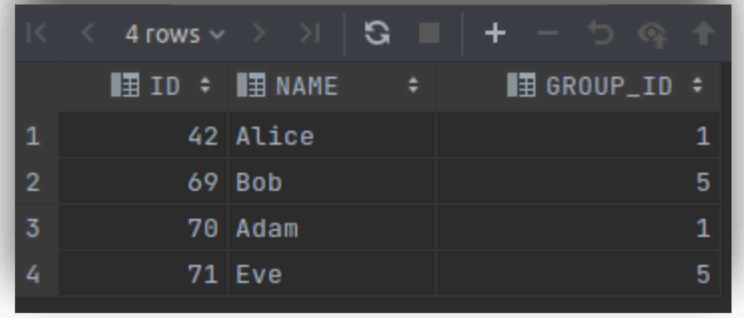
```
begin
    insert into UniGroup(name, c_val) values('953502', 0);
    insert into UniGroup(name, c_val) values('953503', 0);
end;

select * from UniGroup;
```



	ID	NAME	C_VAL
1	1	953501	0
2	5	953505	0
3	6	953502	0
4	7	953503	0

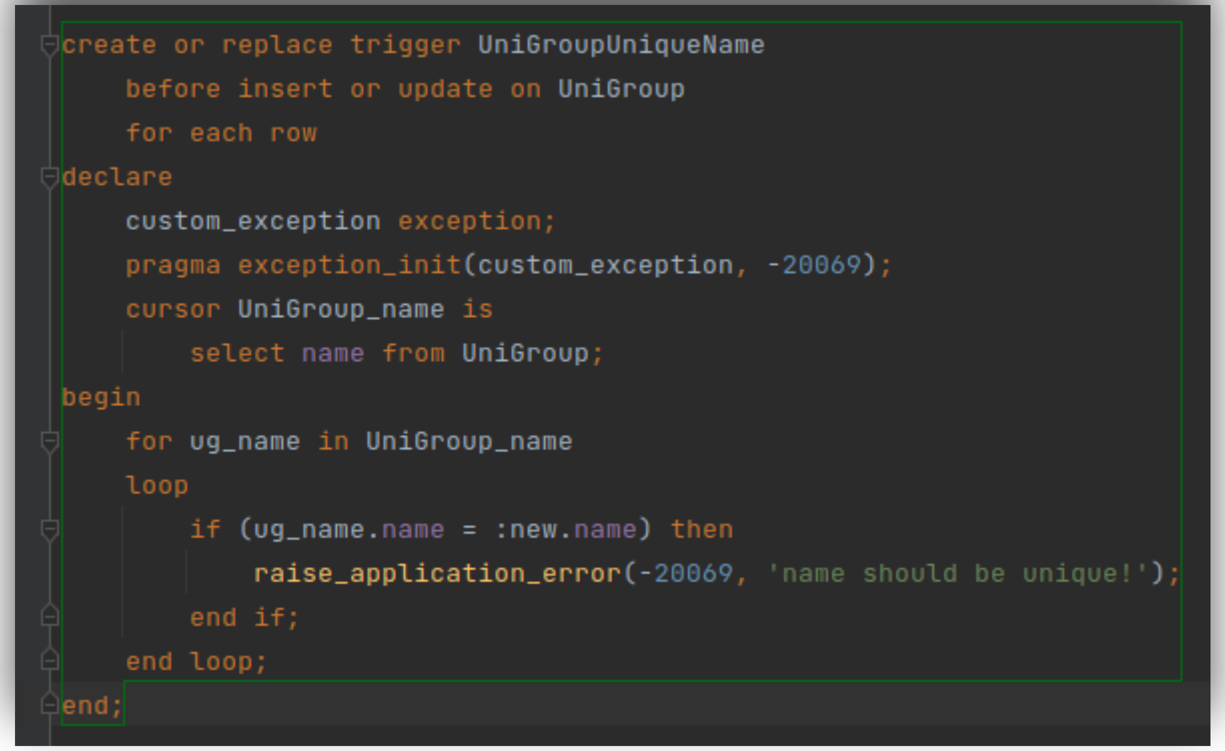


```
begin
    insert into UniStudent(name, group_id) values('Adam', 1);
    insert into UniStudent(name, group_id) values('Eve', 5);
end;

select * from UniStudent;
```



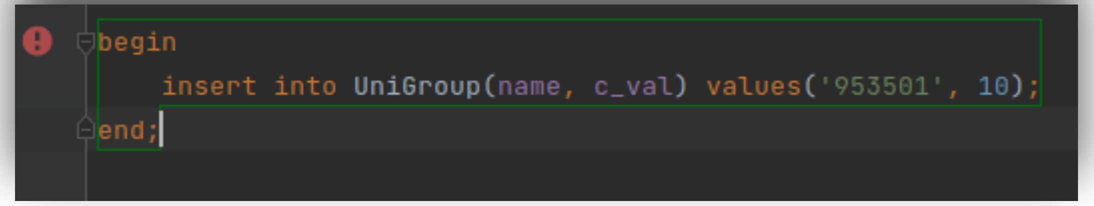
	ID	NAME	GROUP_ID
1	42	Alice	1
2	69	Bob	5
3	70	Adam	1
4	71	Eve	5

Проверка уникальности UniGroup.name:

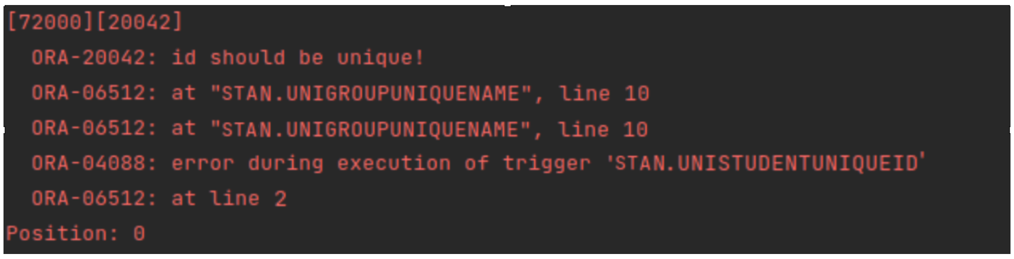


```
create or replace trigger UniGroupUniqueName
    before insert or update on UniGroup
    for each row
declare
    custom_exception exception;
    pragma exception_init(custom_exception, -20069);
    cursor UniGroup_name is
        select name from UniGroup;
begin
    for ug_name in UniGroup_name
    loop
        if (ug_name.name = :new.name) then
            raise_application_error(-20069, 'name should be unique!');
        end if;
    end loop;
end;
```

Результат:



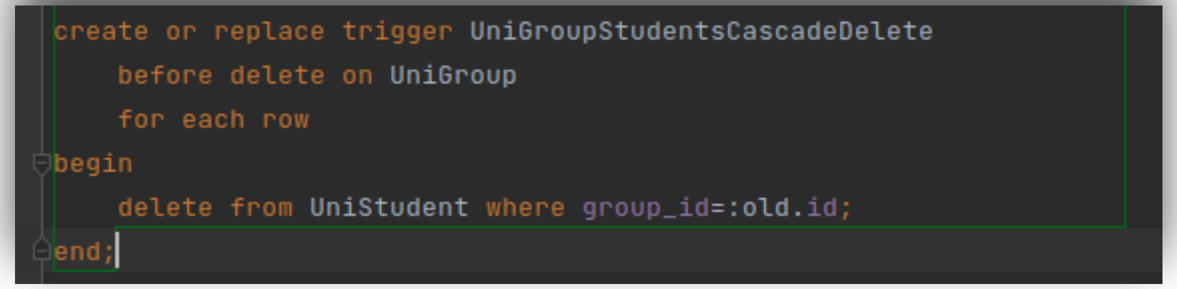
```
begin
    insert into UniGroup(name, c_val) values('953501', 10);
end;
```



```
[72000][20042]
  ORA-20042: id should be unique!
  ORA-06512: at "STAN.UNIGROUPUNIQUENAME", line 10
  ORA-06512: at "STAN.UNIGROUPUNIQUENAME", line 10
  ORA-04088: error during execution of trigger 'STAN.UNISTUDENTUNIQUEID'
  ORA-06512: at line 2
Position: 0
```

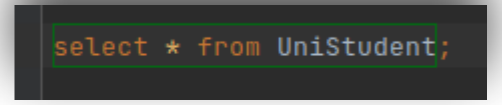
Задание 3

Реализовать триггер реализующий Foreign Key с каскадным удалением между таблицами STUDENTS и GROUPS

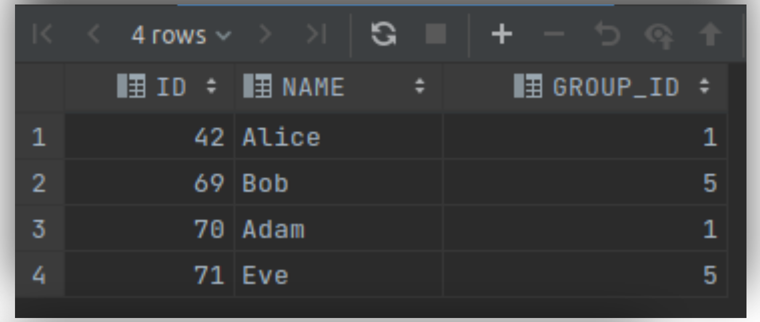


```
create or replace trigger UniGroupStudentsCascadeDelete
    before delete on UniGroup
    for each row
begin
    delete from UniStudent where group_id=:old.id;
end;
```

Результат:

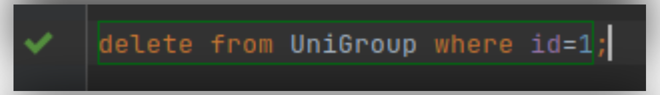


```
select * from UniStudent;
```

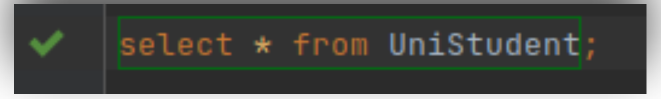


4 rows

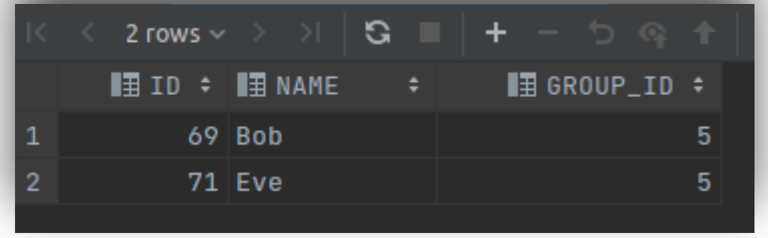
	ID	NAME	GROUP_ID
1	42	Alice	1
2	69	Bob	5
3	70	Adam	1
4	71	Eve	5



```
delete from UniGroup where id=1;
```



```
select * from UniStudent;
```

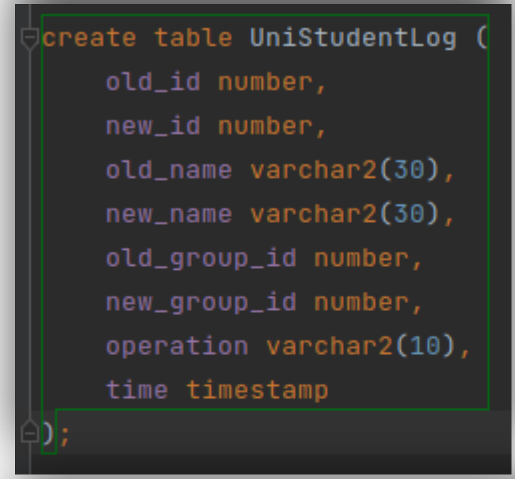


2 rows

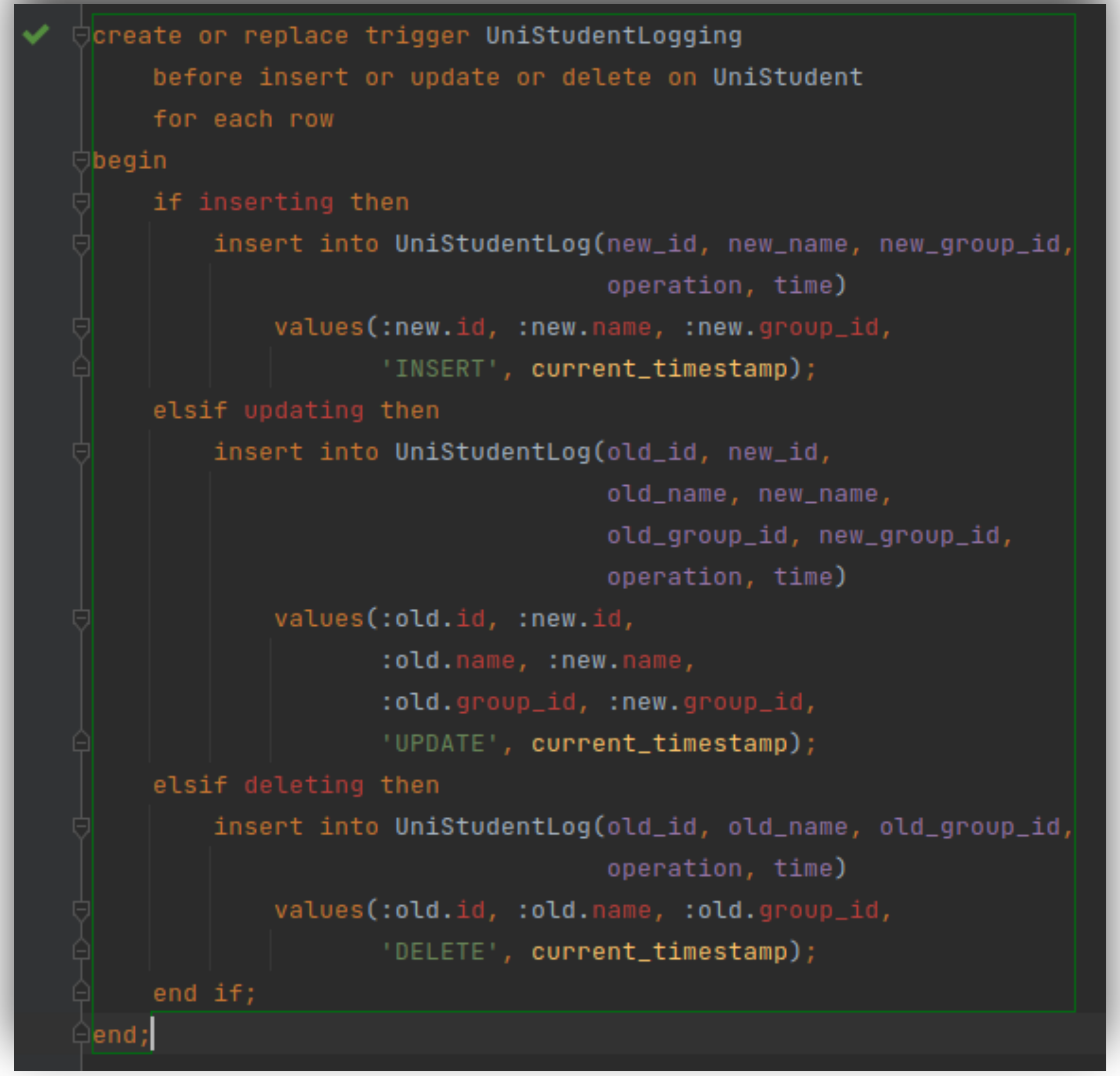
	ID	NAME	GROUP_ID
1	69	Bob	5
2	71	Eve	5

Задание 4

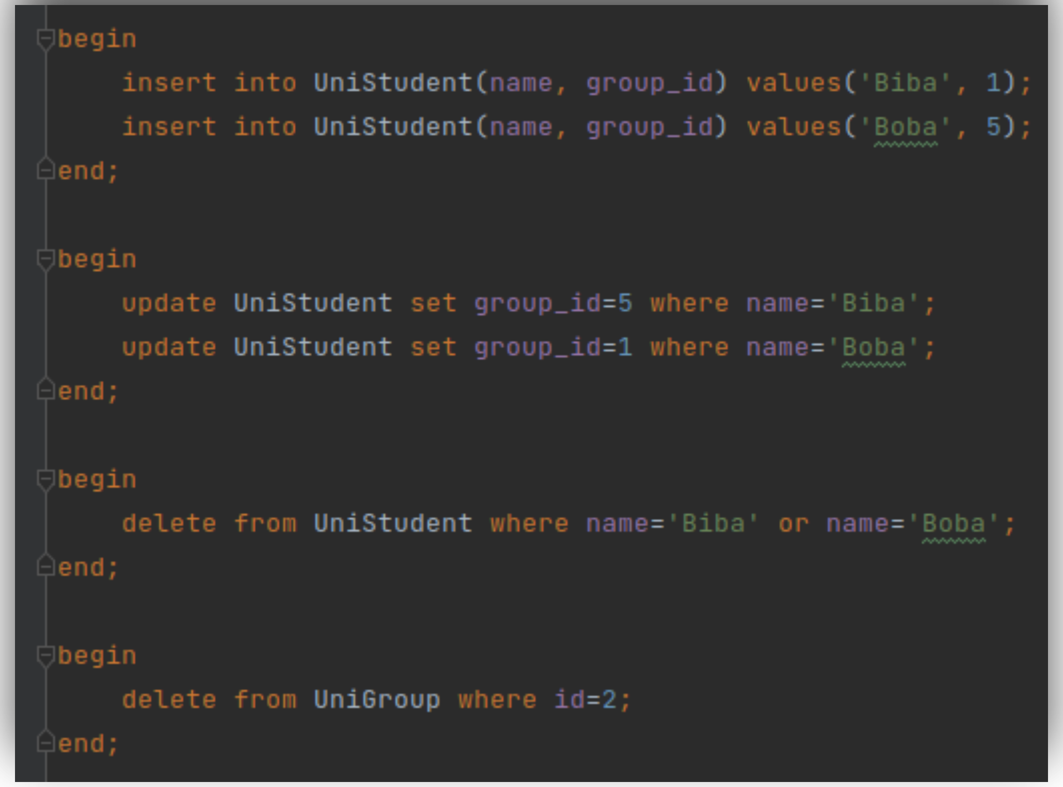
Реализовать триггер реализующий журналирование всех действий над данными таблицы STUDENTS



```sql
create table UniStudentLog (
    old_id number,
    new_id number,
    old_name varchar2(30),
    new_name varchar2(30),
    old_group_id number,
    new_group_id number,
    operation varchar2(10),
    time timestamp
);
```

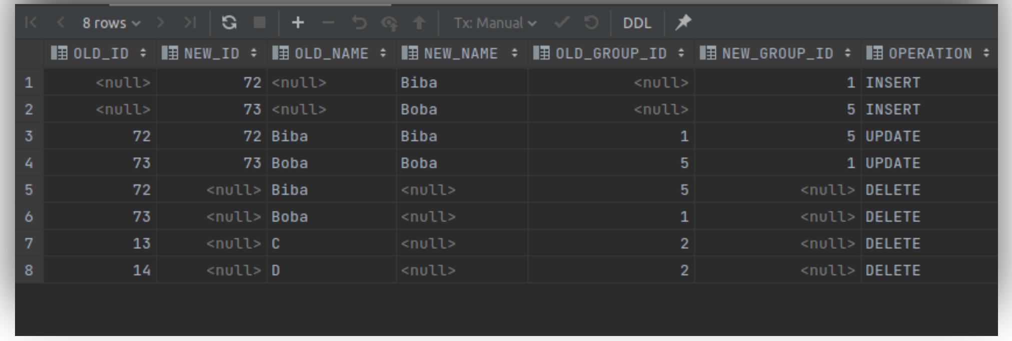


```sql
create or replace trigger UniStudentLogging
    before insert or update or delete on UniStudent
    for each row
begin
    if inserting then
        insert into UniStudentLog(new_id, new_name, new_group_id,
                                  operation, time)
            values(:new.id, :new.name, :new.group_id,
                   'INSERT', current_timestamp);
    elsif updating then
        insert into UniStudentLog(old_id, new_id,
                                  old_name, new_name,
                                  old_group_id, new_group_id,
                                  operation, time)
            values(:old.id, :new.id,
                   :old.name, :new.name,
                   :old.group_id, :new.group_id,
                   'UPDATE', current_timestamp);
    elsif deleting then
        insert into UniStudentLog(old_id, old_name, old_group_id,
                                  operation, time)
            values(:old.id, :old.name, :old.group_id,
                   'DELETE', current_timestamp);
    end if;
end;
```



```
begin
    insert into UniStudent(name, group_id) values('Biba', 1);
    insert into UniStudent(name, group_id) values('Boba', 5);
end;

begin
    update UniStudent set group_id=5 where name='Biba';
    update UniStudent set group_id=1 where name='Boba';
end;

begin
    delete from UniStudent where name='Biba' or name='Boba';
end;

begin
    delete from UniGroup where id=2;
end;
```

Результат:

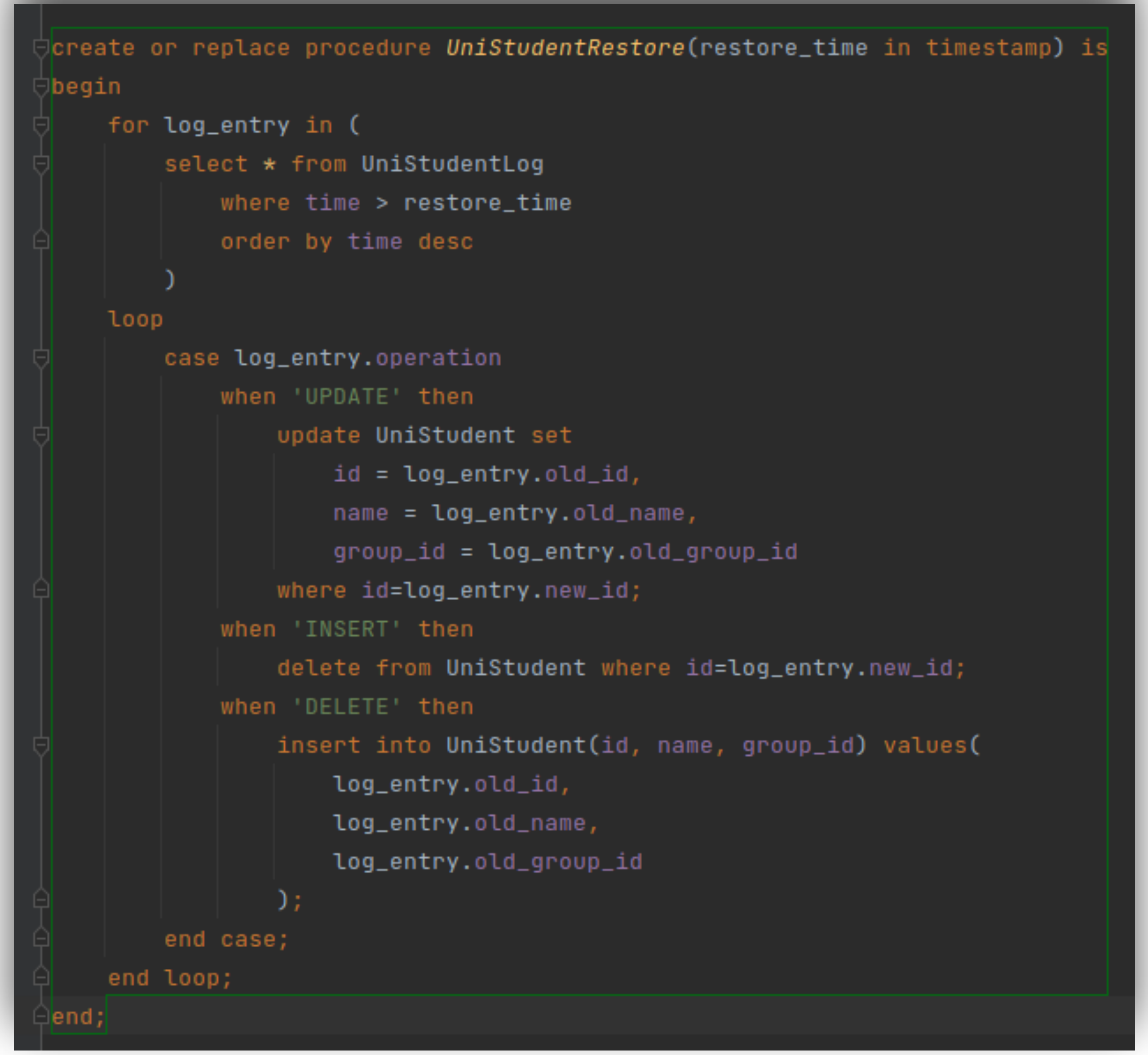


8 rows Tx: Manual DDL

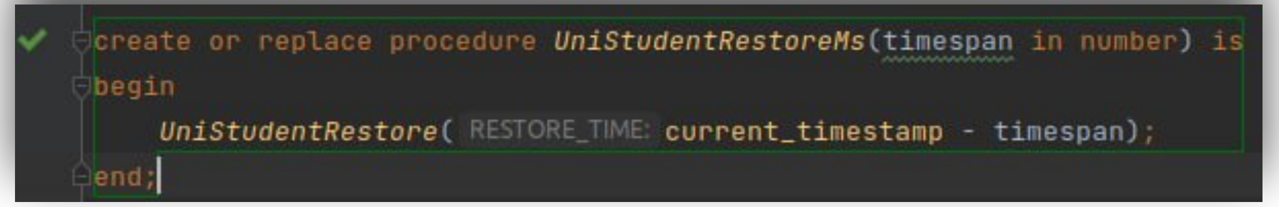
	OLD_ID	NEW_ID	OLD_NAME	NEW_NAME	OLD_GROUP_ID	NEW_GROUP_ID	OPERATION
1	<null>	72	<null>	Biba	<null>	1	INSERT
2	<null>	73	<null>	Boba	<null>	5	INSERT
3	72	72	Biba	Biba	1	5	UPDATE
4	73	73	Boba	Boba	5	1	UPDATE
5	72	<null>	Biba	<null>	5	<null>	DELETE
6	73	<null>	Boba	<null>	1	<null>	DELETE
7	13	<null>	C	<null>	2	<null>	DELETE
8	14	<null>	D	<null>	2	<null>	DELETE

Задание 5

Исходя из данных предыдущей задачи, реализовать процедуру для восстановления информации на указанный временной момент и на временное смещение

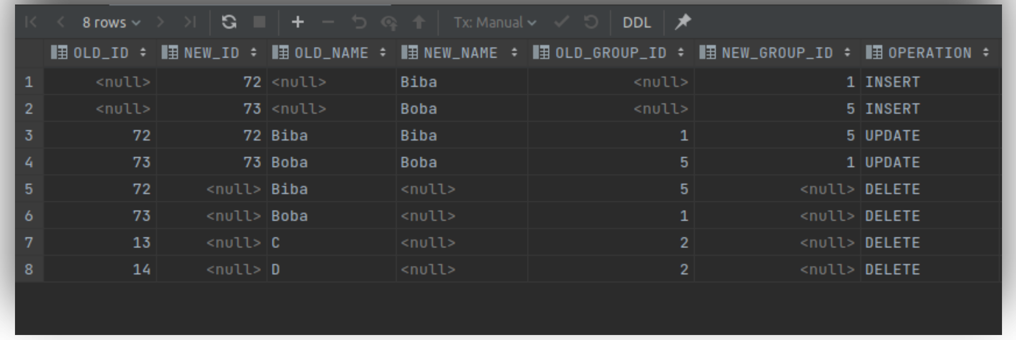


```
create or replace procedure UniStudentRestore(restore_time in timestamp) is
begin
    for log_entry in (
        select * from UniStudentLog
            where time > restore_time
            order by time desc
        )
    loop
        case log_entry.operation
            when 'UPDATE' then
                update UniStudent set
                    id = log_entry.old_id,
                    name = log_entry.old_name,
                    group_id = log_entry.old_group_id
                where id=log_entry.new_id;
            when 'INSERT' then
                delete from UniStudent where id=log_entry.new_id;
            when 'DELETE' then
                insert into UniStudent(id, name, group_id) values(
                    log_entry.old_id,
                    log_entry.old_name,
                    log_entry.old_group_id
                );
        end case;
    end loop;
end;
```



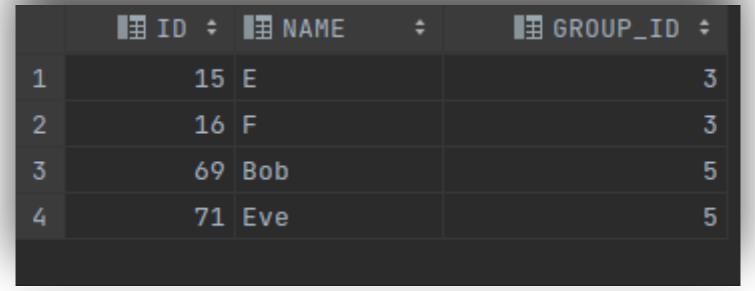
```
create or replace procedure UniStudentRestoreMs(timespan in number) is
begin
    UniStudentRestore( RESTORE_TIME: current_timestamp - timespan);
end;
```

Результат:



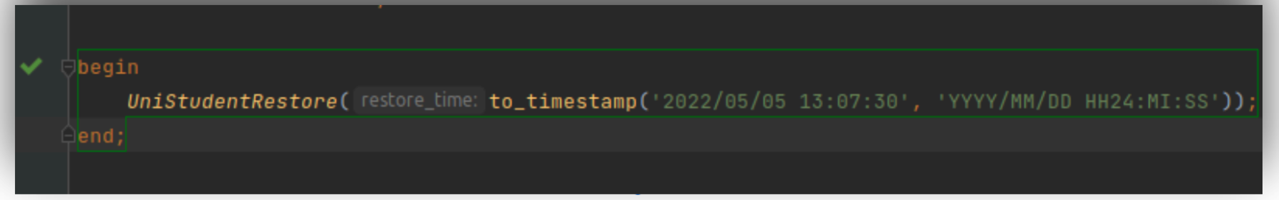
	OLD_ID	NEW_ID	OLD_NAME	NEW_NAME	OLD_GROUP_ID	NEW_GROUP_ID	OPERATION
1	<null>	72	<null>	Biba	<null>	1	INSERT
2	<null>	73	<null>	Boba	<null>	5	INSERT
3	72	72	Biba	Biba	1	5	UPDATE
4	73	73	Boba	Boba	5	1	UPDATE
5	72	<null>	Biba	<null>	5	<null>	DELETE
6	73	<null>	Boba	<null>	1	<null>	DELETE
7	13	<null>	C	<null>	2	<null>	DELETE
8	14	<null>	D	<null>	2	<null>	DELETE

Состояние UniStudent:



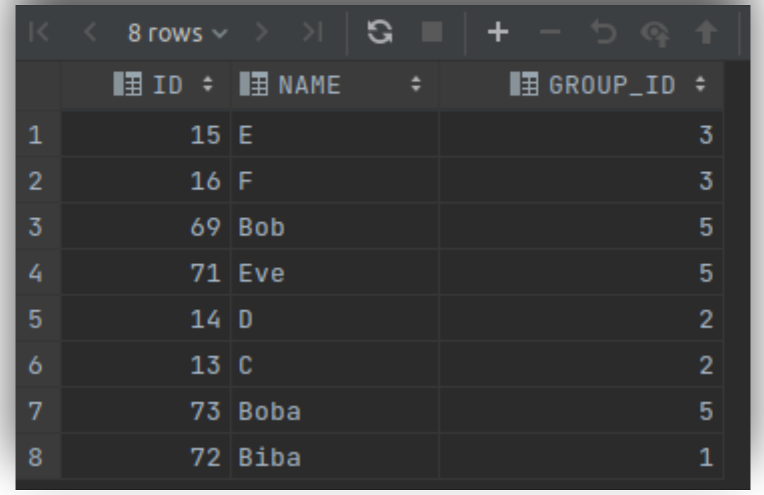
	ID	NAME	GROUP_ID
1	15	E	3
2	16	F	3
3	69	Bob	5
4	71	Eve	5

Вызов процедуры:



```
begin
    UniStudentRestore( restore_time: to_timestamp('2022/05/05 13:07:30', 'YYYY/MM/DD HH24:MI:SS'));
end;
```

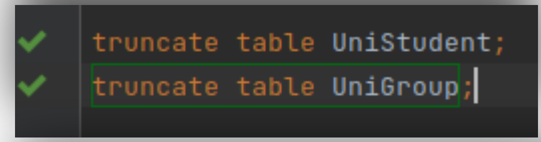
Состояние UniStudent:



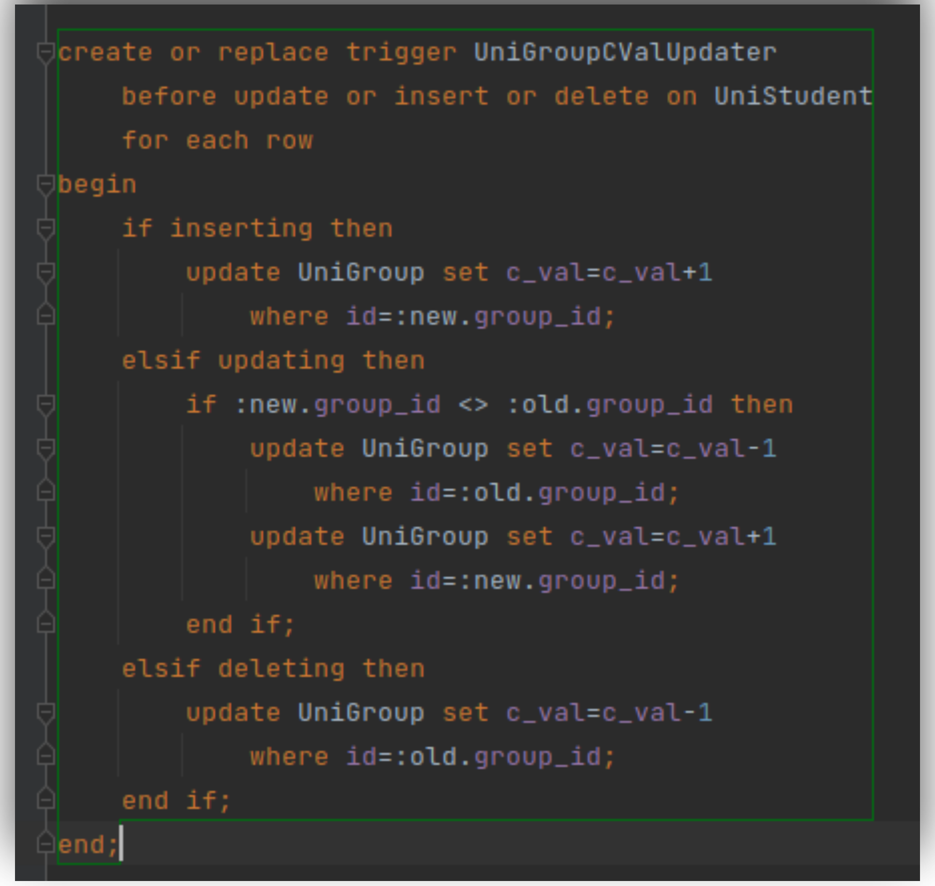
	ID	NAME	GROUP_ID
1	15	E	3
2	16	F	3
3	69	Bob	5
4	71	Eve	5
5	14	D	2
6	13	C	2
7	73	Boba	5
8	72	Biba	1

Задание 6

Реализовать триггер, который в случае изменения данных в таблице STUDENTS будет соответственно обновлять информацию C_VAL таблицы GROUPS

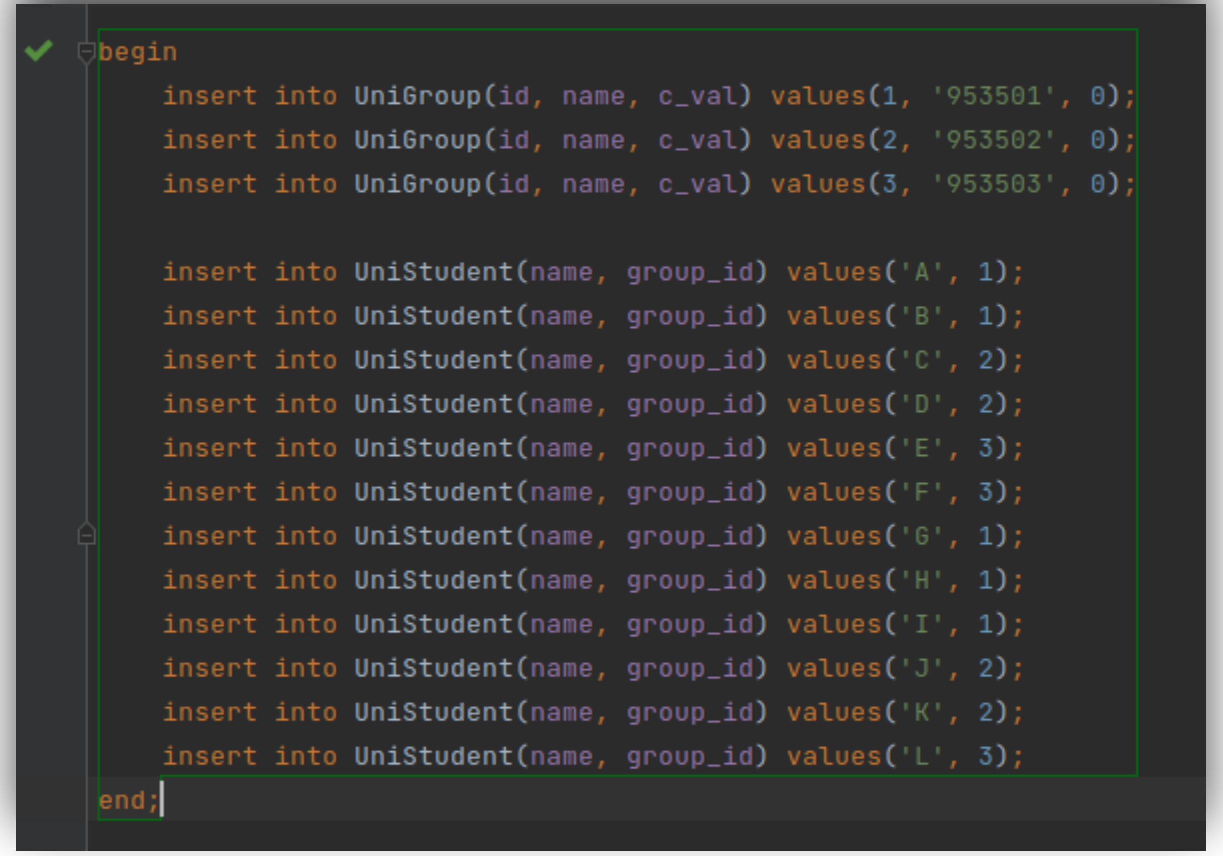


```
truncate table UniStudent;
truncate table UniGroup;
```

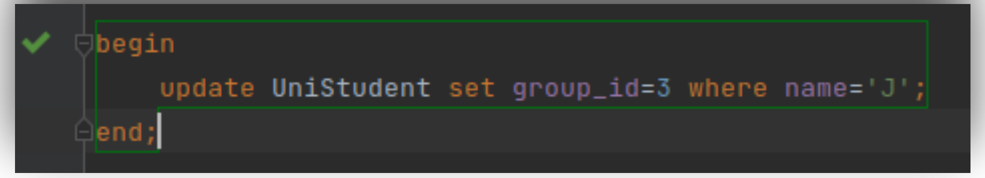


```
create or replace trigger UniGroupCValUpdater
    before update or insert or delete on UniStudent
    for each row
begin
    if inserting then
        update UniGroup set c_val=c_val+1
            where id=:new.group_id;
    elsif updating then
        if :new.group_id <> :old.group_id then
            update UniGroup set c_val=c_val-1
                where id=:old.group_id;
            update UniGroup set c_val=c_val+1
                where id=:new.group_id;
        end if;
    elsif deleting then
        update UniGroup set c_val=c_val-1
            where id=:old.group_id;
    end if;
end;
```

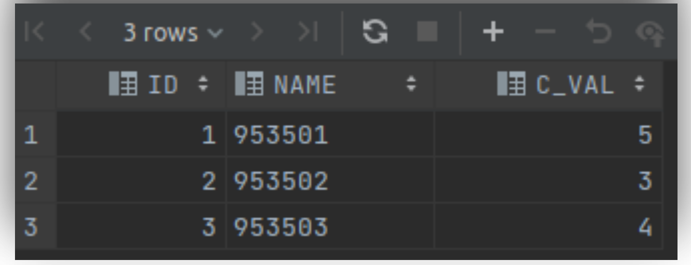
Результат:



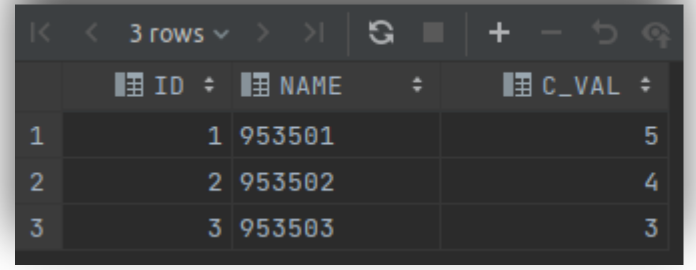
```
begin
    insert into UniGroup(id, name, c_val) values(1, '953501', 0);
    insert into UniGroup(id, name, c_val) values(2, '953502', 0);
    insert into UniGroup(id, name, c_val) values(3, '953503', 0);

    insert into UniStudent(name, group_id) values('A', 1);
    insert into UniStudent(name, group_id) values('B', 1);
    insert into UniStudent(name, group_id) values('C', 2);
    insert into UniStudent(name, group_id) values('D', 2);
    insert into UniStudent(name, group_id) values('E', 3);
    insert into UniStudent(name, group_id) values('F', 3);
    insert into UniStudent(name, group_id) values('G', 1);
    insert into UniStudent(name, group_id) values('H', 1);
    insert into UniStudent(name, group_id) values('I', 1);
    insert into UniStudent(name, group_id) values('J', 2);
    insert into UniStudent(name, group_id) values('K', 2);
    insert into UniStudent(name, group_id) values('L', 3);
end;
```



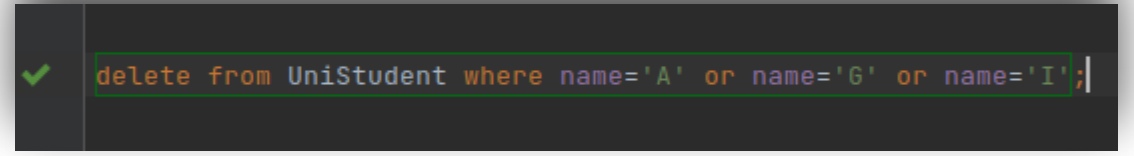
```
begin
    update UniStudent set group_id=3 where name='J';
end;
```



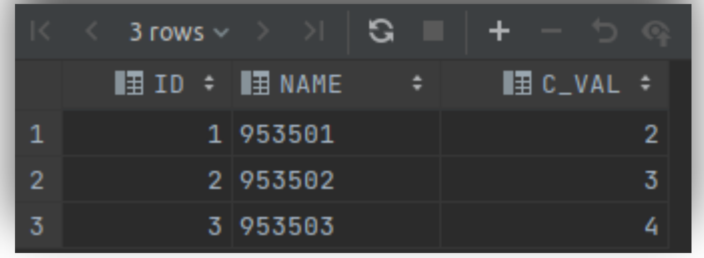
	ID	NAME	C_VAL
1	1	953501	5
2	2	953502	3
3	3	953503	4



	ID	NAME	C_VAL
1	1	953501	5
2	2	953502	4
3	3	953503	3



```sql
delete from UniStudent where name='A' or name='G' or name='I';
```



	ID	NAME	C_VAL
1	1	953501	2
2	2	953502	3
3	3	953503	4