Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра информатики

# Отчёт

Лабораторная работа №5

| | |
|---|---|
| Выполнил: | Проверил: |
| студент группы №853504 | Чащин С.В. |
| Кузьма В.В. | |

Минск 2021

);

## ЗАДАНИЕ 1.

Создать три таблицы произвольной структуры, необходимые условия: в каждой таблице необходим первичный ключ. В таблицах как минимум 3 столбца. Предусмотреть наличие внешних ключей и наличия столбцов символьного типа, цифрового типа и типа дата-время.

```
CREATE TABLE TABLE1
(
        ID NUMBER PRIMARY KEY,
        COLUMN1 VARCHAR(20)
);

CREATE TABLE TABLE2
(
        ID NUMBER PRIMARY KEY,
        COLUMN1 DATE,
        TABLE1_FK NUMBER,
        CONSTRAINT fk_Table2_Table1 FOREIGN KEY(TABLE1_FK) REFERENCES
TABLE1(ID)
        ON DELETE CASCADE
);

CREATE TABLE TABLE3
(
        ID NUMBER PRIMARY KEY,
        COLUMN1 NUMBER,
        TABLE2_FK NUMBER,
        CONSTRAINT fk_TABLE3_TABLE2 FOREIGN KEY (TABLE2_FK) REFERENCES
TABLE2(ID) ON DELETE CASCADE
);
```

## ЗАДАНИЕ 2.

Реализовать механизм сохранения изменений данных в этих таблицах

(интересуют только DML изменения).

```
create or replace trigger table1_audit_trigger
before delete or insert or update on table1
FOR EACH ROW
begin
  CASE
    WHEN INSERTING THEN
        INSERT INTO table1_audit(operation, change_time, is_reverted, testcolumn, id_row)
            VALUES ('INSERT' ,CURRENT_TIMESTAMP, 0, :NEW.testcolumn, :NEW.id);
    WHEN DELETING THEN
        INSERT INTO table1_audit(operation, change_time, is_reverted, testcolumn, id_row)
            VALUES ('DELETE',CURRENT_TIMESTAMP,0,:OLD.testcolumn,:OLD.id);
    WHEN UPDATING THEN
        INSERT INTO table1_audit(operation, change_time ,is_reverted, testcolumn, id_row)
            VALUES ('UPDATE',CURRENT_TIMESTAMP,0,:OLD.testcolumn,:OLD.id);
    END CASE;
end;
/


create or replace trigger table2_audit_trigger
before delete or insert or update on table2
FOR EACH ROW
begin
  CASE
    WHEN INSERTING THEN
        INSERT INTO table2_audit(operation, change_time, is_reverted, testcolumn, id_row)
            VALUES ('INSERT' ,CURRENT_TIMESTAMP, 0, :NEW.testcolumn, :NEW.id);
    WHEN DELETING THEN
```

```
        INSERT INTO table2_audit(operation, change_time, is_reverted, testcolumn, id_row)
              VALUES ('DELETE',CURRENT_TIMESTAMP,0,:OLD.testcolumn,:OLD.id);
    WHEN UPDATING THEN
        INSERT INTO table2_audit(operation, change_time ,is_reverted, testcolumn, id_row)
              VALUES ('UPDATE',CURRENT_TIMESTAMP,0,:OLD.testcolumn,:OLD.id);
    END CASE;
end;
/

create or replace trigger table3_audit_trigger
before delete or insert or update on table3
FOR EACH ROW
begin
  CASE
    WHEN INSERTING THEN
        INSERT INTO table3_audit(operation, change_time, is_reverted, testcolumn, id_row)
              VALUES ('INSERT' ,CURRENT_TIMESTAMP, 0, :NEW.testcolumn, :NEW.id);
    WHEN DELETING THEN
        INSERT INTO table3_audit(operation, change_time, is_reverted, testcolumn, id_row)
              VALUES ('DELETE',CURRENT_TIMESTAMP,0,:OLD.testcolumn,:OLD.id);
    WHEN UPDATING THEN
        INSERT INTO table3_audit(operation, change_time ,is_reverted, testcolumn, id_row)
              VALUES ('UPDATE',CURRENT_TIMESTAMP,0,:OLD.testcolumn,:OLD.id);
    END CASE;
end;
```

## ЗАДАНИЕ 3.

Реализовать перегруженную пакетную процедуру на вход которой подается либо дата-время либо интервал в миллисекундах в первом случае должен происходить откат всех изменений на заданную дату-время, во втором на указанное количество миллисекунд назад.

```
CREATE Or REPLACE TYPE string_array AS VARRAY(3) OF VARCHAR2(10);
```

```
/

create or replace procedure restore_child
(
  table_name in varchar2 ,
  restore_until TIMESTAMP
) as
    child_array string_array;
begin
    child_array := get_dependent_tables(table_name);
    restore_data(child_array,restore_until);
end restore_child;
/

create or replace function get_dependent_tables
(
  in_table_name in varchar2
) return string_array as
    dependent_tables string_array:=string_array();
    indx NUMBER;
begin
   FOR relation IN (SELECT p.table_name,ch.table_name child
        FROM user_cons_columns p
        JOIN user_constraints ch ON p.constraint_name = ch.r_constraint_name
        WHERE p.table_name= in_table_name) LOOP
    dependent_tables.extend;
    indx := indx +1;
    dependent_tables(indx):=relation.child;
    END LOOP;
    return dependent_tables;
end get_dependent_tables;
/
```

```
create or replace procedure restore_table1(restore_until TIMESTAMP) as
begin
  FOR  audit_row in (SELECT id, operation, testcolumn, id_row, change_time FROM TABLE1_AUDIT
                                              WHERE change_time > restore_until
                                              AND is_reverted = 0  ) LOOP
          CASE audit_row.operation
           WHEN 'UPDATE' THEN
                     DBMS_OUTPUT.put_line(  'UPDATE  TABLE1   SET  COLUUMN1  =  '  || audit_row.testcolumn ||  ' WHERE ID = ' || audit_row.id_row);
                    INSERT INTO audit_scripts(operation,script) VALUES ('UPDATE', 'UPDATE TABLE1 SET COLUUMN1 = ' || audit_row.testcolumn || ' WHERE ID = ' || audit_row.id_row);
          WHEN 'DELETE' THEN
                     DBMS_OUTPUT.put_line('INSERT INTO TABLE1(testcolumn) VALUES (' || audit_row.testcolumn || ')');
                    INSERT INTO audit_scripts(operation,script) VALUES  ('DELETE','INSERT INTO TABLE1(testcolumn) VALUES (' || audit_row.testcolumn || ')');
                 restore_child('table1',audit_row.change_time);
          WHEN 'INSERT' THEN
                    DBMS_OUTPUT.put_line('DELETE  FROM  TABLE1  WHERE  ID='  || audit_row.id_row );
                    INSERT INTO  audit_scripts(operation,script) VALUES ('INSERT','DELETE FROM TABLE1 WHERE ID=' || audit_row.id_row  );
                 restore_child('table1',audit_row.change_time);
          END CASE;
  END LOOP;



  UPDATE TABLE1_AUDIT
  SET is_reverted = 1
  WHERE change_time > restore_until;
end restore_table1;
/
```

```
create or replace procedure restore_table2(restore_until TIMESTAMP) as
begin
  FOR  audit_row in (SELECT id, operation, testcolumn, id_row, change_time FROM TABLE2_AUDIT
                    WHERE change_time > restore_until
                    AND is_reverted = 0  ) LOOP
        CASE audit_row.operation
         WHEN 'UPDATE' THEN
               DBMS_OUTPUT.put_line( 'UPDATE  TABLE2   SET  COLUUMN1  =  '  || audit_row.testcolumn ||  ' WHERE ID = ' || audit_row.id_row);
              INSERT INTO audit_scripts(operation,script) VALUES ('UPDATE', 'UPDATE TABLE2 SET COLUUMN1 = ' || audit_row.testcolumn || ' WHERE ID = ' || audit_row.id_row);
        WHEN 'DELETE' THEN
               DBMS_OUTPUT.put_line('INSERT INTO TABLE2(testcolumn) VALUES (' || audit_row.testcolumn || ')');
              INSERT INTO audit_scripts(operation,script) VALUES  ('DELETE','INSERT INTO TABLE2(testcolumn) VALUES (' || audit_row.testcolumn || ')');
            restore_child('table2',audit_row.change_time);
        WHEN 'INSERT' THEN
              DBMS_OUTPUT.put_line('DELETE  FROM  TABLE2  WHERE  ID='  || audit_row.id_row );
              INSERT INTO  audit_scripts(operation,script) VALUES ('INSERT','DELETE FROM TABLE2 WHERE ID=' || audit_row.id_row  );
            restore_child('table2',audit_row.change_time);
        END CASE;
  END LOOP;



  UPDATE TABLE2_AUDIT
  SET is_reverted = 1
  WHERE change_time > restore_until;
end restore_table2;
/
```

```
create or replace procedure restore_table3(restore_until TIMESTAMP) as
begin
  FOR  audit_row in (SELECT id, operation, testcolumn, id_row, change_time FROM TABLE3_AUDIT
                     WHERE change_time > restore_until
                     AND is_reverted = 0  ) LOOP
        CASE audit_row.operation
         WHEN 'UPDATE' THEN
               DBMS_OUTPUT.put_line( 'UPDATE TABLE3   SET COLUUMN1 = ' || audit_row.testcolumn ||  ' WHERE ID = ' || audit_row.id_row);
               INSERT INTO audit_scripts(operation,script) VALUES ('UPDATE', 'UPDATE TABLE3 SET COLUUMN1 = ' || audit_row.testcolumn || ' WHERE ID = ' || audit_row.id_row);
        WHEN 'DELETE' THEN
               DBMS_OUTPUT.put_line('INSERT INTO TABLE3(testcolumn) VALUES (' || audit_row.testcolumn || ')');
               INSERT INTO audit_scripts(operation,script) VALUES  ('DELETE','INSERT INTO TABLE3(testcolumn) VALUES (' || audit_row.testcolumn || ')');
             restore_child('table3',audit_row.change_time);
        WHEN 'INSERT' THEN
               DBMS_OUTPUT.put_line('DELETE FROM TABLE3 WHERE ID=' || audit_row.id_row );
               INSERT INTO  audit_scripts(operation,script) VALUES ('INSERT','DELETE FROM TABLE3 WHERE ID=' || audit_row.id_row  );
             restore_child('table3',audit_row.change_time);
        END CASE;
  END LOOP;



  UPDATE TABLE3_AUDIT
  SET is_reverted = 1
  WHERE change_time > restore_until;
end restore_table3;
/
```

```
create or replace package body restore_pkg as
procedure db_back(rollback_timestamp in timestamp, table_names string_array) as
        begin
                restore_data(table_names, rollback_timestamp);
        end db_rollback;
        procedure db_rollback(rollback_millisecond in number, table_names string_array) as
rollback_timestamp timestamp;
        begin
                SELECT current_timestamp - interval '0.001' second * rollback_millisecond INTO
rollback_timestamp FROM dual;
        end db_back;
end restore_pkg;
/


create or replace procedure restore_data
(
  input_tables in string_array ,
  input_ts in TIMESTAMP
) as
begin
  FOR i in 1..input_tables.count LOOP
     EXECUTE IMMEDIATE '
      BEGIN
           RESTORE_' || input_tables(i)|| '( TO_TIMESTAMP(''' || TO_CHAR(input_ts,'DD-MM-
YYYY HH:MI:SS') || ''', ''DD-MM-YYYYHH:MI:SS''));
      END;';
  END LOOP;
end restore_data;
```

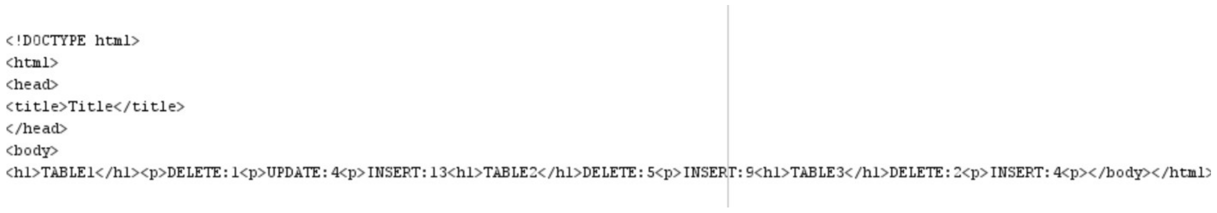
## ЗАДАНИЕ 4.

Предусмотреть процедуру создания отчета об изменениях произошедших либо с момента последнего отчета либо начиная с указанной даты-времени. В отчет должна попасть информация по каждой таблице о количестве проделанных INSERT, UPDATE, DELETE, изменения которые отменены в отчете не должны быть указаны. Отчет необходимо формировать в формате HTML.

```
create or replace procedure create_audit(table_names in string_array) as
begin
   FOR i in 1..table_names.count LOOP
      EXECUTE IMMEDIATE 'ALTER TRIGGER ' || table_names(i) || '_AUDIT_TRIGGER' || ' DISABLE';
   END LOOP;
  FOR audit_script_row IN (SELECT script FROM audit_scripts ORDER BY ID DESC) LOOP
   DBMS_OUTPUT.put_line('EXECUTING:' ||  audit_script_row.script);
   EXECUTE IMMEDIATE audit_script_row.script;
  END LOOP;
  DELETE FROM audit_scripts;
  FOR i in 1..table_names.count LOOP
      EXECUTE IMMEDIATE 'ALTER TRIGGER ' || table_names(i)|| '_AUDIT_TRIGGER' || ' ENABLE';
   END LOOP;
   DELETE FROM AUDIT_SCRIPTS;
end create_audit;


create or replace function html_create(table_names IN string_array,ts IN TIMESTAMP) return varchar2 as
html_document VARCHAR2(500):='<!DOCTYPE html>
<html>
<head>
<title>Title</title>
</head>
<body>
';
operation_count NUMBER;
sys_ref_c SYS_REFCURSOR;
```

```
operation_name VARCHAR(20);
begin
  FOR i in 1..table_names.count LOOP
  html_document := html_document || '<h1>' || table_names(i) || '</h1>';
    OPEN sys_ref_c FOR 'SELECT operation,COUNT(*) FROM ' || table_names(i) || '_AUDIT ' || 'WHERE   is_reverted=0 AND change_time > TO_TIMESTAMP(''' || TO_CHAR(ts,'DD-MM-YYYY HH:MI:SS') || ''', ''DD-MM-YYYYHH:MI:SS'') GROUP BY operation';
        LOOP
            FETCH sys_ref_c INTO operation_name,operation_count;
            EXIT WHEN sys_ref_c%NOTFOUND;
            html_document := html_document || operation_name || ':' || operation_count || '<p>';
        END LOOP;
    CLOSE sys_ref_c;
  END LOOP;
  html_document := html_document || '</body></html>';
  return html_document;
end html_create;
```



```
<!DOCTYPE html>
<html>
<head>
<title>Title</title>
</head>
<body>
<h1>TABLE1</h1><p>DELETE:1<p>UPDATE:4<p>INSERT:13<h1>TABLE2</h1>DELETE:5<p>INSERT:9<h1>TABLE3</h1>DELETE:2<p>INSERT:4<p></body></html>
```