Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Отчет к лабораторной работе №5

Выполнил:

студент гр. 953501

Кондрашов И.Д.

Проверил:

Чащин C. В.

Минск 2022

# 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;

