Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа № 5

**Создание таблиц истории и возможность «откатится» к заданному времени. Генерация отчетов.**

Выполнил студент гр. 953501

Баяк Е.И.

Проверил

Чащин С.В.

Минск, 2022

Задание

Одной из ключевых возможностей современных баз данных является возможность строить ретроспективные запросы, т.е. выполнение запросов с тем чтобы они выдали результат такой, какой он был в каком-то обозримом прошлом, либо приведение данных к какому-либо моменту в прошлом (откат изменений). В нашу задачу входит обеспечение возможности отмены выполненных изменений к какому-либо моменту в прошлом. Для этого необходимо:

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.
2. Реализовать механизм сохранения изменений данных в этих таблицах (интересуют только DML изменения).
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.
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.

Решение

Для решения поставленной задачи в программе Oracle были написаны триггеры, процедуры, функции и пакеты.

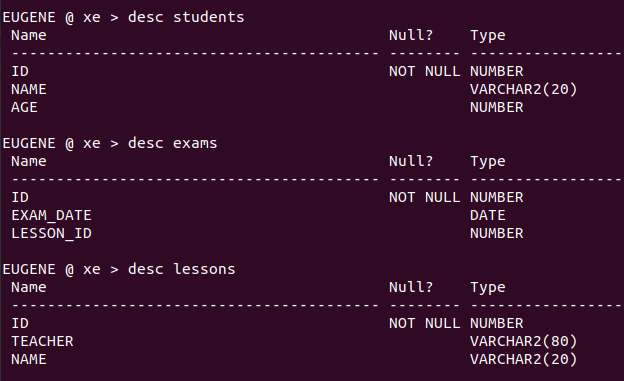


Рисунок 1. Созданные таблицы.

Для каждой таблицы есть дополнительная, которая хранит операции DML.

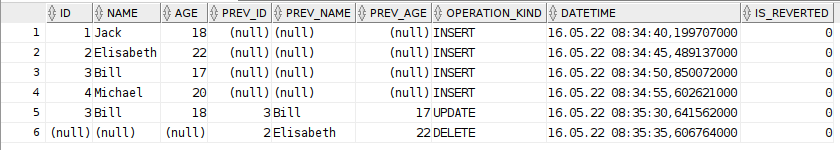


Рисунок 2. Данные таблицы logs\_students.

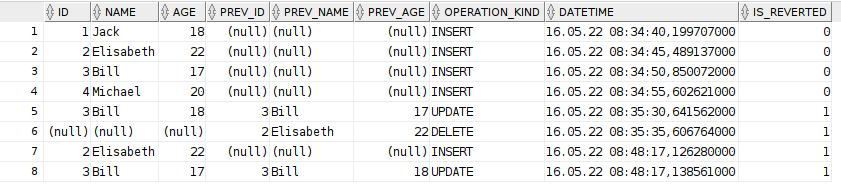


Рисунок 3. Данные таблицы после отката изменений на 08:35:00.

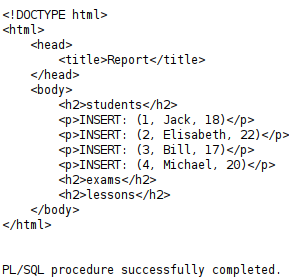


Рисунок 4. Генерация отчета.

Код программы

CREATE TABLE students (

id NUMBER PRIMARY KEY,

name VARCHAR(20),

age NUMBER

);

CREATE TABLE lessons (

id NUMBER PRIMARY KEY,

teacher VARCHAR2(80),

name VARCHAR2(20)

);

CREATE TABLE exams (

id NUMBER PRIMARY KEY,

exam\_date DATE,

lesson\_id NUMBER,

CONSTRAINT fk\_lessons FOREIGN KEY ( lesson\_id )

REFERENCES lessons ( id )

ON DELETE CASCADE

);

CREATE TABLE logs\_students (

id NUMBER,

name VARCHAR(20),

age NUMBER,

prev\_id NUMBER,

prev\_name VARCHAR(20),

prev\_age NUMBER,

operation\_kind VARCHAR(15),

datetime TIMESTAMP,

is\_reverted NUMBER

);

CREATE TABLE logs\_lessons (

id NUMBER,

teacher VARCHAR2(80),

name VARCHAR2(20),

prev\_id NUMBER,

prev\_teacher VARCHAR2(80),

prev\_name VARCHAR(20),

operation\_kind VARCHAR(15),

datetime TIMESTAMP,

is\_reverted NUMBER

);

CREATE TABLE logs\_exams (

id NUMBER,

exam\_date DATE,

lesson\_id NUMBER,

prev\_id NUMBER,

prev\_exam\_date DATE,

prev\_lesson\_id NUMBER,

operation\_kind VARCHAR(15),

datetime TIMESTAMP,

is\_reverted NUMBER

);

CREATE OR REPLACE TRIGGER arch\_students BEFORE

DELETE OR INSERT OR UPDATE ON students

FOR EACH ROW

BEGIN

IF inserting THEN

INSERT INTO logs\_students (

id,

name,

age,

operation\_kind,

datetime,

is\_reverted

) VALUES (

:new.id,

:new.name,

:new.age,

'INSERT',

current\_timestamp,

0

);

ELSIF updating THEN

INSERT INTO logs\_students (

id,

name,

age,

prev\_id,

prev\_name,

prev\_age,

operation\_kind,

datetime,

is\_reverted

) VALUES (

:new.id,

:new.name,

:new.age,

:old.id,

:old.name,

:old.age,

'UPDATE',

current\_timestamp,

0

);

ELSIF deleting THEN

INSERT INTO logs\_students (

prev\_id,

prev\_name,

prev\_age,

operation\_kind,

datetime,

is\_reverted

) VALUES (

:old.id,

:old.name,

:old.age,

'DELETE',

current\_timestamp,

0

);

END IF;

END;

/

CREATE OR REPLACE TRIGGER arch\_lessons BEFORE

DELETE OR INSERT OR UPDATE ON lessons

FOR EACH ROW

BEGIN

IF inserting THEN

INSERT INTO logs\_lessons (

id,

name,

teacher,

operation\_kind,

datetime,

is\_reverted

) VALUES (

:new.id,

:new.name,

:new.teacher,

'INSERT',

current\_date,

0

);

ELSIF updating THEN

INSERT INTO logs\_lessons (

id,

name,

teacher,

prev\_id,

prev\_name,

prev\_teacher,

operation\_kind,

datetime,

is\_reverted

) VALUES (

:new.id,

:new.name,

:new.teacher,

:old.id,

:old.name,

:old.teacher,

'UPDATE',

current\_date,

0

);

ELSIF deleting THEN

INSERT INTO logs\_lessons (

prev\_id,

prev\_name,

prev\_teacher,

operation\_kind,

datetime,

is\_reverted

) VALUES (

:old.id,

:old.name,

:old.teacher,

'DELETE',

current\_date,

0

);

END IF;

END;

/

CREATE OR REPLACE TRIGGER arch\_exams BEFORE

DELETE OR INSERT OR UPDATE ON exams

FOR EACH ROW

BEGIN

IF inserting THEN

INSERT INTO logs\_exams (

id,

exam\_date,

lesson\_id,

operation\_kind,

datetime,

is\_reverted

) VALUES (

:new.id,

:new.exam\_date,

:new.lesson\_id,

'INSERT',

current\_date,

0

);

ELSIF updating THEN

INSERT INTO logs\_exams (

id,

exam\_date,

lesson\_id,

prev\_id,

prev\_exam\_date,

prev\_lesson\_id,

operation\_kind,

datetime,

is\_reverted

) VALUES (

:new.id,

:new.exam\_date,

:new.lesson\_id,

:old.id,

:old.exam\_date,

:old.lesson\_id,

'UPDATE',

current\_date,

0

);

ELSIF deleting THEN

INSERT INTO logs\_exams (

prev\_id,

prev\_exam\_date,

prev\_lesson\_id,

operation\_kind,

datetime,

is\_reverted

) VALUES (

:old.id,

:old.exam\_date,

:old.lesson\_id,

'DELETE',

current\_date,

0

);

END IF;

END;

/

SET SERVEROUTPUT ON SIZE UNLIMITED

CREATE OR REPLACE TYPE string\_array AS

VARRAY(3) OF VARCHAR2(10);

/

CREATE OR REPLACE FUNCTION get\_dependent\_tables (

in\_table\_name IN VARCHAR2

) RETURN string\_array AS

dependent\_tables string\_array := string\_array();

i 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;

i := i + 1;

dependent\_tables(i) := relation.child;

END LOOP;

RETURN dependent\_tables;

END;

/

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, 'YYYY/MM/DD HH24:MI:SS')

|| ''', ''YYYY/MM/DDHH24:MI:SS''));

END;

';

END LOOP;

END;

/

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;

/

CREATE TABLE logs\_scripts (

canceled\_operation VARCHAR(10),

executed\_script VARCHAR(500)

);

CREATE OR REPLACE PROCEDURE restore\_students (

restore\_until TIMESTAMP

) AS

stmt VARCHAR(500);

BEGIN

FOR logs\_row IN (

SELECT

id,

name,

age,

prev\_id,

prev\_name,

prev\_age,

operation\_kind,

datetime

FROM

logs\_students

WHERE

datetime > restore\_until

AND is\_reverted = 0

ORDER BY

datetime DESC

) LOOP

CASE logs\_row.operation\_kind

WHEN 'UPDATE' THEN

stmt := 'UPDATE students SET name = '''

|| logs\_row.prev\_name

|| ''', age = '

|| logs\_row.prev\_age

|| ' WHERE ID = '

|| logs\_row.id;

WHEN 'DELETE' THEN

stmt := 'INSERT INTO students(id, name, age) VALUES ('

|| logs\_row.prev\_id

|| ', '''

|| logs\_row.prev\_name

|| ''', '

|| logs\_row.prev\_age

|| ')';

WHEN 'INSERT' THEN

stmt := 'DELETE FROM students WHERE ID=' || logs\_row.id;

END CASE;

EXECUTE IMMEDIATE stmt;

INSERT INTO logs\_scripts (

canceled\_operation,

executed\_script

) VALUES (

logs\_row.operation\_kind,

stmt

);

restore\_child('students', logs\_row.datetime);

END LOOP;

UPDATE logs\_students

SET

is\_reverted = 1

WHERE

datetime > restore\_until;

END;

/

CREATE OR REPLACE PROCEDURE restore\_lessons (

restore\_until TIMESTAMP

) AS

stmt VARCHAR(500);

BEGIN

FOR logs\_row IN (

SELECT

id,

teacher,

name,

prev\_id,

prev\_teacher,

prev\_name,

operation\_kind,

datetime

FROM

logs\_lessons

WHERE

datetime > restore\_until

AND is\_reverted = 0

ORDER BY

datetime DESC

) LOOP

CASE logs\_row.operation\_kind

WHEN 'UPDATE' THEN

stmt := 'UPDATE lessons SET teacher = '''

|| logs\_row.prev\_teacher

|| ''', name = '''

|| logs\_row.prev\_name

|| ''' WHERE ID = '

|| logs\_row.id;

WHEN 'DELETE' THEN

stmt := 'INSERT INTO lessons(id, tescher, name) VALUES ('

|| logs\_row.prev\_id

|| ', '''

|| logs\_row.prev\_teacher

|| ''', '''

|| logs\_row.prev\_name

|| ''')';

WHEN 'INSERT' THEN

stmt := 'DELETE FROM lessons WHERE ID=' || logs\_row.id;

END CASE;

EXECUTE IMMEDIATE stmt;

INSERT INTO logs\_scripts (

canceled\_operation,

executed\_script

) VALUES (

logs\_row.operation\_kind,

stmt

);

restore\_child('lessons', logs\_row.datetime);

END LOOP;

UPDATE logs\_lessons

SET

is\_reverted = 1

WHERE

datetime > restore\_until;

END;

/

CREATE OR REPLACE PROCEDURE restore\_exams (

restore\_until TIMESTAMP

) AS

stmt VARCHAR(500);

BEGIN

FOR logs\_row IN (

SELECT

id,

exam\_date,

lesson\_id,

prev\_id,

prev\_exam\_date,

prev\_lesson\_id,

operation\_kind,

datetime

FROM

logs\_exams

WHERE

datetime > restore\_until

AND is\_reverted = 0

ORDER BY

datetime DESC

) LOOP

CASE logs\_row.operation\_kind

WHEN 'UPDATE' THEN

stmt := 'UPDATE exams SET exam\_date = (TO\_DATE('''

|| logs\_row.prev\_exam\_date

|| ''', ''DD.MM.YY''), lesson\_id = '

|| logs\_row.prev\_lesson\_id

|| ' WHERE ID = '

|| logs\_row.id;

WHEN 'DELETE' THEN

stmt := 'INSERT INTO exams(id, exam\_date, lesson\_id) VALUES ('

|| logs\_row.prev\_id

|| ', TO\_DATE('''

|| logs\_row.prev\_exam\_date

|| ''', ''DD.MM.YY''), '''

|| logs\_row.prev\_lesson\_id

|| ''')';

WHEN 'INSERT' THEN

stmt := 'DELETE FROM exams WHERE ID=' || logs\_row.id;

END CASE;

EXECUTE IMMEDIATE stmt;

INSERT INTO logs\_scripts (

canceled\_operation,

executed\_script

) VALUES (

logs\_row.operation\_kind,

stmt

);

restore\_child('exams', logs\_row.datetime);

END LOOP;

UPDATE logs\_exams

SET

is\_reverted = 1

WHERE

datetime > restore\_until;

END;

/

CREATE OR REPLACE PACKAGE restore\_pkg AS

PROCEDURE db\_rollback (

rollback\_timestamp IN TIMESTAMP,

table\_names string\_array

);

PROCEDURE db\_rollback (

rollback\_millisecond IN NUMBER,

table\_names string\_array

);

END;

/

CREATE OR REPLACE PACKAGE BODY restore\_pkg AS

PROCEDURE db\_rollback (

rollback\_timestamp IN TIMESTAMP,

table\_names string\_array

) AS

BEGIN

restore\_data(table\_names, rollback\_timestamp);

END;

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;

restore\_data(table\_names, rollback\_timestamp);

END;

END;

/

CREATE OR REPLACE FUNCTION create\_html\_report (

table\_names IN string\_array,

ts IN TIMESTAMP

) RETURN VARCHAR2 AS

html\_document VARCHAR2(10000) := '<!DOCTYPE html>'

|| chr(10)

|| '<html>'

|| chr(10)

|| ' <head>'

|| chr(10)

|| ' <title>Report</title>'

|| chr(10)

|| ' </head>'

|| chr(10)

|| ' <body>'

|| chr(10);

sys\_ref\_c SYS\_REFCURSOR;

logs\_student\_row logs\_students%rowtype;

logs\_lesson\_row logs\_lessons%rowtype;

logs\_exam\_row logs\_exams%rowtype;

BEGIN

FOR i IN 1..table\_names.count LOOP

html\_document := html\_document

|| ' <h2>'

|| table\_names(i)

|| '</h2>'

|| chr(10);

CASE table\_names(i)

WHEN 'students' THEN

OPEN sys\_ref\_c FOR 'SELECT \* FROM logs\_students WHERE is\_reverted=0 AND datetime > TO\_TIMESTAMP('''

|| to\_char(ts, 'DD-MM-YYYY HH24:MI:SS')

|| ''', ''DD-MM-YYYYHH24:MI:SS'')';

LOOP

FETCH sys\_ref\_c INTO logs\_student\_row;

EXIT WHEN sys\_ref\_c%notfound;

CASE logs\_student\_row.operation\_kind

WHEN 'INSERT' THEN

html\_document := html\_document

|| ' <p>'

|| logs\_student\_row.operation\_kind

|| ': ('

|| logs\_student\_row.id

|| ', '

|| logs\_student\_row.name

|| ', '

|| logs\_student\_row.age

|| ')</p>'

|| chr(10);

WHEN 'DELETE' THEN

html\_document := html\_document

|| ' <p>'

|| logs\_student\_row.operation\_kind

|| ': ('

|| logs\_student\_row.prev\_id

|| ', '

|| logs\_student\_row.prev\_name

|| ', '

|| logs\_student\_row.prev\_age

|| ')</p>'

|| chr(10);

ELSE

html\_document := html\_document

|| ' <p>'

|| logs\_student\_row.operation\_kind

|| ': ('

|| logs\_student\_row.id

|| ', '

|| logs\_student\_row.name

|| ', '

|| logs\_student\_row.age

|| ') -> ('

|| logs\_student\_row.prev\_id

|| ', '

|| logs\_student\_row.prev\_name

|| ', '

|| logs\_student\_row.prev\_age

|| ')</p>'

|| chr(10);

END CASE;

END LOOP;

CLOSE sys\_ref\_c;

WHEN 'lessons' THEN

OPEN sys\_ref\_c FOR 'SELECT \* FROM logs\_lessons WHERE is\_reverted=0 AND datetime > TO\_TIMESTAMP('''

|| to\_char(ts, 'DD-MM-YYYY HH24:MI:SS')

|| ''', ''DD-MM-YYYYHH24:MI:SS'')';

LOOP

FETCH sys\_ref\_c INTO logs\_lesson\_row;

EXIT WHEN sys\_ref\_c%notfound;

CASE logs\_lesson\_row.operation\_kind

WHEN 'INSERT' THEN

html\_document := html\_document

|| ' <p>'

|| logs\_lesson\_row.operation\_kind

|| ': ('

|| logs\_lesson\_row.id

|| ', '

|| logs\_lesson\_row.teacher

|| ', '

|| logs\_lesson\_row.name

|| ')</p>'

|| chr(10);

WHEN 'DELETE' THEN

html\_document := html\_document

|| ' <p>'

|| logs\_lesson\_row.operation\_kind

|| ': ('

|| logs\_lesson\_row.prev\_id

|| ', '

|| logs\_lesson\_row.prev\_teacher

|| ', '

|| logs\_lesson\_row.prev\_name

|| ')</p>'

|| chr(10);

ELSE

html\_document := html\_document

|| ' <p>'

|| logs\_lesson\_row.operation\_kind

|| ': ('

|| logs\_lesson\_row.id

|| ', '

|| logs\_lesson\_row.teacher

|| ', '

|| logs\_lesson\_row.name

|| ') -> ('

|| logs\_lesson\_row.prev\_id

|| ', '

|| logs\_lesson\_row.prev\_teacher

|| ', '

|| logs\_lesson\_row.prev\_name

|| ')</p>'

|| chr(10);

END CASE;

END LOOP;

CLOSE sys\_ref\_c;

WHEN 'exams' THEN

OPEN sys\_ref\_c FOR 'SELECT \* FROM logs\_exams WHERE is\_reverted=0 AND datetime > TO\_TIMESTAMP('''

|| to\_char(ts, 'DD-MM-YYYY HH24:MI:SS')

|| ''', ''DD-MM-YYYYHH24:MI:SS'')';

LOOP

FETCH sys\_ref\_c INTO logs\_exam\_row;

EXIT WHEN sys\_ref\_c%notfound;

CASE logs\_exam\_row.operation\_kind

WHEN 'INSERT' THEN

html\_document := html\_document

|| ' <p>'

|| logs\_exam\_row.operation\_kind

|| ': ('

|| logs\_exam\_row.id

|| ', '

|| logs\_exam\_row.exam\_date

|| ', '

|| logs\_exam\_row.lesson\_id

|| ')</p>'

|| chr(10);

WHEN 'DELETE' THEN

html\_document := html\_document

|| ' <p>'

|| logs\_exam\_row.operation\_kind

|| ': ('

|| logs\_exam\_row.prev\_id

|| ', '

|| logs\_exam\_row.prev\_exam\_date

|| ', '

|| logs\_exam\_row.prev\_lesson\_id

|| ')</p>'

|| chr(10);

ELSE

html\_document := html\_document

|| ' <p>'

|| logs\_exam\_row.operation\_kind

|| ': ('

|| logs\_exam\_row.id

|| ', '

|| logs\_exam\_row.exam\_date

|| ', '

|| logs\_exam\_row.lesson\_id

|| ') -> ('

|| logs\_exam\_row.prev\_id

|| ', '

|| logs\_exam\_row.prev\_exam\_date

|| ', '

|| logs\_exam\_row.prev\_lesson\_id

|| ')</p>'

|| chr(10);

END CASE;

END LOOP;

CLOSE sys\_ref\_c;

ELSE

dbms\_output.put\_line('undefined table');

END CASE;

END LOOP;

html\_document := html\_document

|| ' </body>'

|| chr(10)

|| '</html>';

RETURN html\_document;

END;

/