Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Отчет к лабораторной работе №2

Выполнил:

студент гр. 953501

Кондрашов И.Д.

Проверил:

Чащин C. В.

Минск 2022

Лабораторная самостоятельная работа №2

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

|  |  |  |
| --- | --- | --- |
| Поле | Тип | комментарий |
| **STUDENTS** | | |
| ID | Number | Код студента |
| NAME | VARCHAR2 | Имя студента |
| GROUP\_ID | Number | Код группы |
| **GROUPS** | | |
| ID | Number | Код группы |
| NAME | VARCHAR2 | Название группы |
| C\_VAL | Number | Количество студентов в группе |

CREATE TABLE GROUPS

(ID number primary key not null,

NAME varchar2(100) not null,

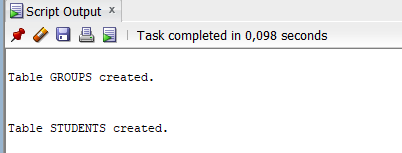
C\_VAL number);

CREATE TABLE STUDENTS

(ID number primary key,

NAME varchar2(100),

GROUP\_ID number REFERENCES GROUPS(ID));



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

CREATE OR REPLACE TRIGGER check\_unique\_groupname

BEFORE INSERT OR UPDATE OF NAME ON GROUPS

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

rowcount NUMBER;

e\_not\_unique\_groupname EXCEPTION;

PRAGMA exception\_init(e\_not\_unique\_groupname, -20001);

BEGIN

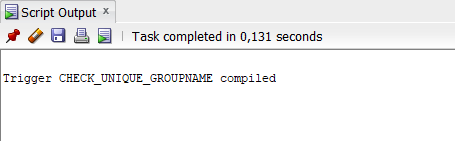
SELECT COUNT(\*) INTO rowcount FROM GROUPS WHERE NAME = :new.NAME;

IF rowcount != 0 THEN

RAISE e\_not\_unique\_groupname;

END IF;

END;



CREATE OR REPLACE TRIGGER check\_unique\_id\_student

BEFORE INSERT OR UPDATE OF NAME ON STUDENTS

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

rowcount NUMBER;

e\_not\_unique\_groupname EXCEPTION;

PRAGMA exception\_init(e\_not\_unique\_groupname, -20001);

BEGIN

SELECT COUNT(\*) INTO rowcount FROM STUDENTS WHERE ID = :new.ID;

IF rowcount != 0 THEN

RAISE e\_not\_unique\_groupname;

END IF;

END;



CREATE OR REPLACE TRIGGER check\_unique\_id\_group

BEFORE INSERT OR UPDATE OF NAME ON GROUPS

FOR EACH ROW

DECLARE

PRAGMA AUTONOMOUS\_TRANSACTION;

rowcount NUMBER;

e\_not\_unique\_groupname EXCEPTION;

PRAGMA exception\_init(e\_not\_unique\_groupname, -20001);

BEGIN

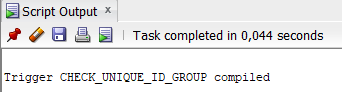
SELECT COUNT(\*) INTO rowcount FROM GROUPS WHERE ID = :new.ID;

IF rowcount != 0 THEN

RAISE e\_not\_unique\_groupname;

END IF;

END;



CREATE OR REPLACE TRIGGER generate\_group\_id

BEFORE INSERT ON GROUPS

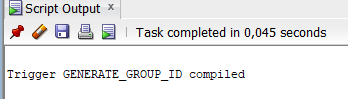
FOR EACH ROW

FOLLOWS check\_unique\_groupname

BEGIN

SELECT groups\_id\_sequence.nextval INTO :new.ID FROM dual;

END;



CREATE OR REPLACE TRIGGER generate\_student\_id

BEFORE INSERT ON STUDENTS

FOR EACH ROW

BEGIN

SELECT students\_id\_sequence.nextval INTO :new.ID FROM dual;

END;



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

CREATE OR REPLACE TRIGGER group\_del

BEFORE DELETE ON GROUPS

FOR EACH ROW

DECLARE

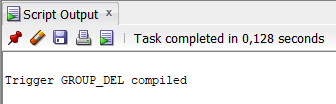
PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

DELETE FROM STUDENTS WHERE GROUP\_ID = :old.ID;

COMMIT;

END;



4. Реализовать триггер реализующий журналирование всех действий над данными таблицы STUDENTS

CREATE TABLE LOGS

(TIME TIMESTAMP NOT NULL,

STATEMENT varchar2(100) NOT NULL,

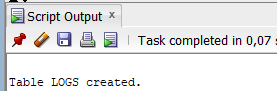
ID\_NEW number,

ID\_OLD number,

NAME varchar2(100),

GROUP\_ID number

);



CREATE OR REPLACE TRIGGER logging

AFTER INSERT OR DELETE OR UPDATE ON STUDENTS

FOR EACH ROW

BEGIN

CASE

WHEN INSERTING THEN

INSERT INTO LOGS VALUES(SYSTIMESTAMP,'INSERT',:new.ID,null,null,null);

WHEN UPDATING THEN

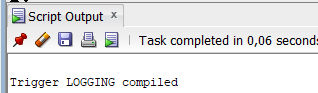
INSERT INTO LOGS VALUES(SYSTIMESTAMP,'UPDATE',:new.ID,:old.ID,:old.NAME,:old.GROUP\_ID);

WHEN DELETING THEN

INSERT INTO LOGS VALUES(SYSTIMESTAMP,'DELETE',null,:old.ID,:old.NAME,:old.GROUP\_ID);

END CASE;

END;



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

CREATE OR REPLACE PROCEDURE Restore(start\_time IN TIMESTAMP)

IS

CURSOR selected\_logs IS

SELECT \* FROM LOGS

WHERE TIME>=start\_time

ORDER BY TIME DESC;

BEGIN

FOR selected\_log IN selected\_logs

LOOP

CASE

WHEN selected\_log.STATEMENT='INSERT' THEN

DELETE FROM STUDENTS WHERE ID=selected\_log.ID\_NEW;

WHEN selected\_log.STATEMENT='UPDATE' THEN

UPDATE STUDENTS

SET ID=selected\_log.ID\_OLD,NAME=selected\_log.NAME,GROUP\_ID =selected\_log.GROUP\_ID

WHERE ID=selected\_log.ID\_NEW;

WHEN selected\_log.STATEMENT='DELETE' THEN

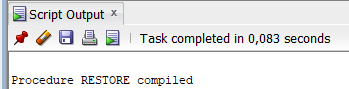
INSERT INTO STUDENTS VALUES(selected\_log.ID\_OLD,selected\_log.NAME,selected\_log.GROUP\_ID);

END CASE;

DELETE FROM LOGS WHERE TIME=selected\_log.TIME;

END LOOP;

END Restore;



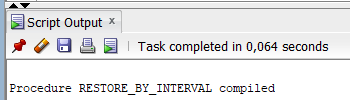
CREATE OR REPLACE PROCEDURE Restore\_by\_interval(interval IN INTERVAL DAY TO SECOND)

IS

BEGIN

Restore(LOCALTIMESTAMP - interval);

END Restore\_by\_interval;



6. Реализовать триггер, который в случае изменения данных в таблице STUDENTS будет соответственно обновлять информацию C\_VAL таблицы GROUPS

CREATE OR REPLACE TRIGGER students\_change

AFTER INSERT OR DELETE OR UPDATE ON STUDENTS

FOR EACH ROW

DECLARE

rowcount NUMBER;

BEGIN

CASE

WHEN INSERTING THEN

UPDATE GROUPS

SET C\_VAL=C\_VAL+1

WHERE ID = :new.GROUP\_ID;

WHEN UPDATING THEN

UPDATE GROUPS

SET C\_VAL=C\_VAL+1

WHERE ID = :new.GROUP\_ID;

UPDATE GROUPS

SET C\_VAL=C\_VAL-1

WHERE ID = :old.GROUP\_ID;

WHEN DELETING THEN

UPDATE GROUPS

SET C\_VAL=C\_VAL-1

WHERE ID = :old.GROUP\_ID;

END CASE;

END;

