Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Институт информатики и вычислительной техники

Кафедра прикладной математики и кибернетики

Расчётно-графическое задание
по дисциплине
«Сетевые базы данных»

Выполнил:
студент гр.ИП-014
Обухов А.И.

Проверила:
старший преподаватель каф. ПМиК
Дьячкова Ирина Сергеевна

Новосибирск 2024 г.

Оглавление

Задания на РГЗ

1. Создать две таблицы, каждая из которых должна иметь первичный ключ и, по крайней мере, один столбец с ограничением NOT NULL. Таблицы должны быть связаны внешним ключом; тип связи - "один-ко-многим".

2. Создать пакет, содержащий процедуру начального заполнения таблиц данными (по 7-10 записей в таблице) и процедуру очистки таблиц (удаления записей).

3. Для одной из таблиц разработать триггер для обеспечения дополнительных ограничений на изменение данных таблицы **(см. свой вариант задания)**.

4. Создать представление, которое позволяет запрашивать данные из обеих (связанных) таблиц. Представление должно ограничивать доступ к данным по столбцам и строкам.

5. Написать второй пакет, в состав которого включить вызовы процедур из первого пакета.

- В пакет также поместить процедуру изменения данных в таблицах **(см. свой вариант задания)**.
- Значения изменяемых данных должны передаваться в процедуру как параметры.
- В процедурах предусмотреть обработку исключений.
- Обеспечить подтверждение транзакций при их успешном выполнении и откат - в случае возникновения исключительной ситуации.

6. Предоставить привилегии всем пользователям базы данных Oracle на использование представления для просмотра данных.
Предоставить привилегию конкретному пользователю на выполнение процедуры изменения данных.

7. Отчет должен отвечать всем требованиям к оформлению курсовых работ и содержать текст задания, тексты сценариев, пакетов, содержимое таблиц и результаты запросов и выполнения процедур.

Варианты заданий находятся в файлах IP-NNN.txt, где NNN - номер группы.

Свой вариант задания

14. Обухов Артем Игоревич

В таблицах должна содержаться информация об Авиарейсах и Категориях рейсов. Каждый рейс может иметь несколько сервисных категорий. Процедура должна добавлять рейс в таблицу. Триггер должен разрешать добавление только после 20-го числа. Включить в пакет еще одну процедуру, которая выводит количества категорий для каждого рейса, кроме рейса, заданногов параметре. Выборку данных производить в запись, созданную на основе явно определяемого курсора.

Процесс и результаты выполнения работы

```
DROP TABLE categories
```

Table dropped. 0.79 seconds

```
DROP TABLE flights
```

Table dropped. 0.81 seconds

```
CREATE TABLE flights (     id INT PRIMARY KEY,     destination VARCHAR(50) NOT NULL )
```

Table created. 0.06 seconds

```
CREATE TABLE categories (     id INT PRIMARY KEY,     flight_id INT NOT NULL,     name VARCHAR(50) NOT NULL,     FOREIGN KEY (flight_id)
REFERENCES flights(id) )
```

Table created. 0.03 seconds

```
CREATE OR REPLACE PACKAGE DataControlPackage AS     PROCEDURE FillData;     PROCEDURE ClearData; END DataControlPackage;
```

Package created. 0.02 seconds

```
CREATE OR REPLACE PACKAGE BODY DataControlPackage AS     PROCEDURE FillData AS     BEGIN         INSERT INTO flights (id, destination)
VALUES (1, 'New York');         INSERT INTO flights (id, destination) VALUES (2, 'Paris');         INSERT INTO flights (id, destination)
VALUES (3, 'London');         INSERT INTO flights (id, destination) VALUES (4, 'Moscow');         INSERT INTO flights (id, destination)
VALUES (5, 'Los Angeles');         INSERT INTO flights (id, destination) VALUES (6, 'Berlin');         INSERT INTO flights (id,
destination) VALUES (7, 'Toronto');         INSERT INTO categories (id, flight_id, name) VALUES (1, 3, 'VIP');         INSERT INTO
categories (id, flight_id, name) VALUES (2, 2, 'Eco');         INSERT INTO categories (id, flight_id, name) VALUES (3, 2, 'VIP');
INSERT INTO categories (id, flight_id, name) VALUES (4, 4, 'Business');         INSERT INTO categories (id, flight_id, name) VALUES (5, 5,
'VIP');         INSERT INTO categories (id, flight_id, name) VALUES (6, 5, 'VIP');         INSERT INTO categories (id, flight_id, name)
VALUES (7, 2, 'VIP');         INSERT INTO categories (id, flight_id, name) VALUES (8, 1, 'VIP');         INSERT INTO categories (id,
flight_id, name) VALUES (9, 3, 'VIP');         INSERT INTO categories (id, flight_id, name) VALUES (10, 3, 'VIP');     END FillData;
PROCEDURE ClearData AS     BEGIN         DELETE FROM categories;         DELETE FROM flights;     END ClearData; END DataControlPackage;
```

Package Body created. 0.04 seconds

```
CREATE OR REPLACE TRIGGER AllowAfter20th BEFORE INSERT ON flights FOR EACH ROW BEGIN     IF EXTRACT(DAY FROM SYSDATE) >= 20 THEN
RAISE_APPLICATION_ERROR(-20001, 'Adding flights is only allowed before the 20th of the month.');     END IF; END;
```

Trigger created. 0.02 seconds

```
INSERT INTO flights(id, destination) VALUES (99, 'TEST')
```

1 row(s) inserted. 0.02 seconds

```
CREATE OR REPLACE VIEW FlightsView AS     SELECT flights.id AS flight_id, categories.name AS category     FROM flights     LEFT JOIN
categories ON flights.id = categories.flight_id
```

View created. 0.04 seconds

```
CREATE OR REPLACE PACKAGE TaskPackage AS     PROCEDURE CallFirstPackageProcedures;     PROCEDURE InsertFlight(id IN INT, destination IN
VARCHAR);     PROCEDURE CountCategoriesForFlights(exclude_flight_id IN INT); END TaskPackage;
```

Package created. 0.02 seconds

```
CREATE OR REPLACE PACKAGE BODY TaskPackage AS     PROCEDURE CallFirstPackageProcedures AS     BEGIN         DataControlPackage.FillData();
-- Заполнение данными         COMMIT;  -- Подтверждение транзакции         DBMS_OUTPUT.PUT_LINE('Data populated successfully.');
EXCEPTION         WHEN OTHERS THEN             ROLLBACK;  -- Откат в случае ошибки             DBMS_OUTPUT.PUT_LINE('Error populating data:
' || SQLERRM);     END CallFirstPackageProcedures;          PROCEDURE InsertFlight(id IN INT, destination IN VARCHAR) AS     BEGIN
INSERT INTO flights (id, destination) VALUES (id, destination);         COMMIT;         DBMS_OUTPUT.PUT_LINE('Data modified
successfully.');     EXCEPTION         WHEN OTHERS THEN             ROLLBACK;  -- Откат в случае ошибки
DBMS_OUTPUT.PUT_LINE('Error modifying data: ' || SQLERRM);     END InsertFlight;      PROCEDURE CountCategoriesForFlights(exclude_flight_id
IN INT) AS         CURSOR flights_cursor IS             SELECT f.id AS flight_id, COUNT(c.id) AS category_count             FROM flights f
LEFT JOIN categories c ON f.id = c.flight_id             WHERE f.id != exclude_flight_id             GROUP BY f.id;         flight_row
flights_cursor%ROWTYPE;     BEGIN         OPEN flights_cursor;         LOOP             FETCH flights_cursor INTO flight_row;
EXIT WHEN flights_cursor%NOTFOUND;             DBMS_OUTPUT.PUT_LINE('Flight ID: ' || flight_row.flight_id || ', Category Count: ' ||
flight_row.category_count);         END LOOP;         CLOSE flights_cursor;     EXCEPTION         WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error counting categories for flights: ' || SQLERRM);     END CountCategoriesForFlights; END TaskPackage;
```

Package Body created. 0.03 seconds

```
BEGIN     TaskPackage.CallFirstPackageProcedures(); END;
```

Data populated successfully.

Statement processed. 0.04 seconds

Package Body created. 0.03 seconds

```
BEGIN     TaskPackage.CallFirstPackageProcedures(); END;
```

Data populated successfully.

Statement processed. 0.04 seconds

```
SELECT * FROM flights
```

ID	DESTINATION
99	TEST
1	New York
2	Paris
3	London
4	Moscow
5	Los Angeles
6	Berlin
7	Toronto

8 rows selected. 0.01 seconds

```
SELECT * FROM categories
```

ID	FLIGHT_ID	NAME
1	3	VIP
2	2	Eco
3	2	VIP
4	4	Business
5	5	VIP
6	5	VIP
7	2	VIP
8	1	VIP
9	3	VIP
10	3	VIP

```
SELECT * FROM FlightsView
```

FLIGHT_ID	CATEGORY
3	VIP
2	Eco
2	VIP
4	Business
5	VIP
5	VIP
2	VIP
1	VIP
3	VIP
3	VIP
6	-
7	-
99	-

13 rows selected. 0.00 seconds

```
BEGIN     TaskPackage.CountCategoriesForFlights(3); END;
```

```
Flight ID: 1, Category Count: 1
Flight ID: 2, Category Count: 3
Flight ID: 4, Category Count: 1
Flight ID: 5, Category Count: 2
Flight ID: 6, Category Count: 0
Flight ID: 7, Category Count: 0
Flight ID: 99, Category Count: 0
```

Statement processed. 0.02 seconds

```
GRANT SELECT ON FlightsView TO PUBLIC
```

Statement processed. 0.01 seconds

```
GRANT EXECUTE ON TaskPackage TO PUBLIC
```

Statement processed. 0.01 seconds

```
GRANT UPDATE, INSERT, DELETE ON flights TO USER
```

Листинг

```
DROP TABLE categories;
DROP TABLE flights;

CREATE TABLE flights (
    id INT PRIMARY KEY,
    destination VARCHAR(50) NOT NULL
);

CREATE TABLE categories (
    id INT PRIMARY KEY,
    flight_id INT NOT NULL,
    name VARCHAR(50) NOT NULL,
    FOREIGN KEY (flight_id) REFERENCES flights(id)
);
```

```
CREATE OR REPLACE PACKAGE DataControlPackage AS
    PROCEDURE FillData;
    PROCEDURE ClearData;
END DataControlPackage;
/

CREATE OR REPLACE PACKAGE BODY DataControlPackage AS
    PROCEDURE FillData AS
    BEGIN
        INSERT INTO flights (id, destination) VALUES (1, 'New York');
        INSERT INTO flights (id, destination) VALUES (2, 'Paris');
        INSERT INTO flights (id, destination) VALUES (3, 'London');
        INSERT INTO flights (id, destination) VALUES (4, 'Moscow');
        INSERT INTO flights (id, destination) VALUES (5, 'Los Angeles');
        INSERT INTO flights (id, destination) VALUES (6, 'Berlin');
        INSERT INTO flights (id, destination) VALUES (7, 'Toronto');

        INSERT INTO categories (id, flight_id, name) VALUES (1, 3, 'VIP');
        INSERT INTO categories (id, flight_id, name) VALUES (2, 2, 'Eco');
        INSERT INTO categories (id, flight_id, name) VALUES (3, 2, 'VIP');
        INSERT INTO categories (id, flight_id, name) VALUES (4, 4, 'Business');
        INSERT INTO categories (id, flight_id, name) VALUES (5, 5, 'VIP');
        INSERT INTO categories (id, flight_id, name) VALUES (6, 5, 'VIP');
        INSERT INTO categories (id, flight_id, name) VALUES (7, 2, 'VIP');
        INSERT INTO categories (id, flight_id, name) VALUES (8, 1, 'VIP');
        INSERT INTO categories (id, flight_id, name) VALUES (9, 3, 'VIP');
        INSERT INTO categories (id, flight_id, name) VALUES (10, 3, 'VIP');

    END FillData;

    PROCEDURE ClearData AS
    BEGIN
        DELETE FROM categories;
        DELETE FROM flights;
    END ClearData;
END DataControlPackage;
/

CREATE OR REPLACE TRIGGER AllowAfter20th
BEFORE INSERT ON flights
FOR EACH ROW
BEGIN
```

```
    IF EXTRACT(DAY FROM SYSDATE) >= 20 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Adding flights is only allowed before the 20th of the month.');
    END IF;
END;
/
INSERT INTO flights(id, destination) VALUES (99, 'TEST');
/
CREATE OR REPLACE VIEW FlightsView AS
    SELECT flights.id AS flight_id, categories.name AS category
    FROM flights
    LEFT JOIN categories ON flights.id = categories.flight_id;
/

CREATE OR REPLACE PACKAGE TaskPackage AS
    PROCEDURE CallFirstPackageProcedures;
    PROCEDURE InsertFlight(id IN INT, destination IN VARCHAR);
    PROCEDURE CountCategoriesForFlights(exclude_flight_id IN INT);
END TaskPackage;
/

CREATE OR REPLACE PACKAGE BODY TaskPackage AS
    PROCEDURE CallFirstPackageProcedures AS
    BEGIN
        DataControlPackage.FillData();   -- Заполнение данными
        COMMIT;  -- Подтверждение транзакции
        DBMS_OUTPUT.PUT_LINE('Data populated successfully.');
    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;  -- Откат в случае ошибки
            DBMS_OUTPUT.PUT_LINE('Error populating data: ' || SQLERRM);
    END CallFirstPackageProcedures;

    PROCEDURE InsertFlight(id IN INT, destination IN VARCHAR) AS
    BEGIN
        INSERT INTO flights (id, destination) VALUES (id, destination);
        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Data modified successfully.');
    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;  -- Откат в случае ошибки
            DBMS_OUTPUT.PUT_LINE('Error modifying data: ' || SQLERRM);
    END InsertFlight;
```

```
    PROCEDURE CountCategoriesForFlights(exclude_flight_id IN INT) AS
        CURSOR flights_cursor IS
            SELECT f.id AS flight_id, COUNT(c.id) AS category_count
            FROM flights f
            LEFT JOIN categories c ON f.id = c.flight_id
            WHERE f.id != exclude_flight_id
            GROUP BY f.id;
        flight_row flights_cursor%ROWTYPE;
    BEGIN
        OPEN flights_cursor;
        LOOP
            FETCH flights_cursor INTO flight_row;
            EXIT WHEN flights_cursor%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE('Flight ID: ' || flight_row.flight_id || ', Category Count: ' ||
flight_row.category_count);
        END LOOP;
        CLOSE flights_cursor;
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error counting categories for flights: ' || SQLERRM);
    END CountCategoriesForFlights;
END TaskPackage;

/
BEGIN
    TaskPackage.CallFirstPackageProcedures();
END;
/
SELECT * FROM flights;
SELECT * FROM categories;
SELECT * FROM FlightsView;
/
BEGIN
    TaskPackage.CountCategoriesForFlights(3);
END;
/
GRANT SELECT ON FlightsView TO PUBLIC;
GRANT EXECUTE ON TaskPackage TO PUBLIC;
GRANT UPDATE, INSERT, DELETE ON flights TO USER;
```