Лабораторные работы по курсу
Базы данных

Лабораторная работа 5
«Команды модификации базы данных»

Москва, 2023

Оглавление

1. Теоретическая часть

В предыдущей лабораторной работе рассматривались вопросы, связанные с проектированием базы данных. В данной лабораторной работе предложены темы, связанные с изменением данных в таблицах – вставкой, обновлением и удалением. В конце работы рассмотрено понятие транзакции в базе данных.

1.1. Добавление данных в таблицу

Для добавления данных в таблицу существует оператор SQL INSERT. Его сокращенный синтаксис представлен ниже.

```
INSERT INTO имя_таблицы [ AS псевдоним ] [ ( имя_столбца [, ...] ) ]
    { DEFAULT VALUES | VALUES ( { выражение | DEFAULT } [, ...] ) [, ...] |
запрос }
```

Данный оператор добавляет строки в таблицу. С помощью INSERT возможно добавить одну или несколько указанных строк, либо ноль или более строк, возвращенных с помощью дополнительного запроса. После ключевых слов INSERT INTO и названия таблицы, в которую будет производиться добавление данных, возможно в скобках указать порядок и названия атрибутов для добавления. В случае отсутствия подобного списка порядок значений данных должен точно соответствовать порядку столбцов в таблице. Возможно не указывать часть названий столбцов – тогда их значения будут автоматически заполнены значением NULL или значением, заданным по умолчанию при создании таблицы.

Например, для заполнения таблицы «Группа» возможно воспользоваться следующим запросом.

```
INSERT INTO Students_group (students_group_number, enrolment_status,
structural_unit_number) VALUES
('ИВТ-41', 'Очная', 1),
('ИВТ-42', 'Очная', 1),
('ИВТ-43', 'Очная', 1),
('ИВТ-21В', 'Заочная', 1),
('ИБ-21', 'Очная', 3),
('ИТД-31', 'Очная', 4),
('ИТД-32', 'Очная', 4),
('ИТД-33', 'Очная', 4);
```

В качестве строк для добавления в таблицу могут быть использованы значения, сформированные в результате запроса к другой таблице. Для иллюстрации приведем следующий пример. Создадим ещё одну таблицу, которая будет содержать информацию о должниках по предметам (т.е. имеющих хотя бы одну оценку 2).

Для этого создадим еще одну таблицу

```
CREATE TABLE debtor_students
(
    ID SERIAL PRIMARY KEY,
    surname VARCHAR(30) NOT NULL,
    name VARCHAR(30) NOT NULL,
    patronymic VARCHAR(30) NULL,
    group_id VARCHAR(7) NOT NULL,
    debt_number INTEGER NOT NULL
)
```

Данная таблица будет хранить информацию о ФИО и группе студента, а также о числе его долгов.

Для заполнения таблицы составим SQL запрос.

```
INSERT INTO debtor_students (surname, name, patronymic,
students_group_number, debt_number)
```

```
(
      SELECT surname, name, patronymic, students_group_number, COUNT(*) AS
"Number of debts"
      FROM student
      INNER JOIN field_comprehension ON field_comprehension.student_id =
student.student_id
      WHERE field_comprehension.mark = 2
      GROUP BY surname, name, patronymic, students_group_number
);
```

Данный запрос выбирает всех студентов, которые имеют хотя бы одну оценку 2, и записывает данные значения в созданную таблицу с должниками.

1.2. Изменение значений

Для изменения существующих значений в базе данных существует команда UPDATE.

```
UPDATE имя_таблицы [ * ] [ [ AS ] псевдоним ]
    SET { имя_столбца = { выражение | DEFAULT } |
          ( имя_столбца [, ...] ) = ( { выражение | DEFAULT } [, ...] ) |
          ( имя_столбца [, ...] ) = ( вложенный_SELECT )
        } [, ...]
```

После оператора UPDATE указывается целевая таблица, которая должна быть модифицирована. В предложении SET указывается, какие столбцы в выбранных строках таблицы должны быть обновлены, и для них задаются новые значения. Остальные столбцы сохраняют свои предыдущие значения. С помощью ключевого слова WHERE возможно отобрать строки таблицы, подлежащие обновлению. Если предложение WHERE отсутствует в записи оператора UPDATE, обновляются все строки целевой таблицы.

Например, составим запрос, убирающий один долг у всех студентов групп ИВТ.

```
UPDATE debtor_students
SET debt_number = debt_number - 1
WHERE students_group_number LIKE 'ИВТ%';
```

1.3. Удаление значений

Для удаления значений из базы данных существует команда DELETE.

```
DELETE FROM имя_таблицы [ * ] [ [ AS ] псевдоним ]
    [ WHERE условие ]
```

Команда DELETE удаляет из указанной таблицы строки, удовлетворяющие условию WHERE. Если предложение WHERE отсутствует, она удаляет из таблицы все строки.

Составим запрос на удаление из списка должников всех студентов, не имеющих задолженностей (число долгов = 0).

```
DELETE FROM debtor_students
WHERE debt_number = 0;
```

В случае отсутствия оператора WHERE из таблицы будут удалены все значения. Аналогичного результата возможно добиться, используя команду TRUNCATE.

```
TRUNCATE debtor_students;
```

2. Практическая часть

2.1. Задание 1.

Придумайте и создайте еще одну группу. В группе должно быть не менее 25 студентов, оценки и ФИО студентов должны быть осмысленные.

2.2. Задание 2.

Запустите программу «Симулятор ОРИОКС» и убедитесь, что созданные в предыдущем задании студенты могут получить доступ к программе, а преподаватели могут выставлять им оценки.

2.3. Задание 3.

Напишите запрос по варианту

№	Условие запроса
1	Переведите всех учащихся 3-го курса на 4-й, изменив номер группы
2	Предположим, что одна из групп была расформирована. Составьте запрос таким образом, чтобы оставшиеся студенты были распределены по другим группам потока.
3	Увеличьте заработную плату до прожиточного минимума у тех преподавателей, у которых она меньше данного значения.
4	Составьте запрос, увеличивающий значение стажа на 1 год
5	Удалите преподавателей со ставкой меньше 0.25, средний балл за предметы которых ниже среднего значения по всем дисциплинам кафедры, на котором он преподает
6	Пришла пора отчислений. Удалите всех студентов, у которых число долгов более 4х

2.4. Задание 4.

Создайте временную таблицу, содержащую 10 лучших студентов 3-го курса института МПСУ (имеющих самую высокую среднюю оценку), окончивших учебу без троек. Отсортируйте ее по убыванию среднего балла. Таблица должна содержать ФИО, номер студенческого билета и средний балл.

2.5. Задание 5.

В зависимости от варианта, добавьте значения в исправленную вами базу данных в прошлой лабораторной. В каждую из таблиц необходимо добавить не менее 10 осмысленных значений.

Список литературы

[1] Е. П. Моргунов, PostgreSQL. Основы языка SQL, 1-е ред., Санкт-Петербург: БХВ-Петербург, 2018, p. 336.

[2] «PL/pgSQL — процедурный язык SQL,» [В Интернете]. Available: https://postgrespro.ru/docs/postgresql/15/plpgsql. [Дата обращения: 09 03 2023].

[3] «Исходный код СУБД postgres,» [В Интернете]. Available: https://github.com/postgres/postgres. [Дата обращения: 30 01 2023].

[4] Документация к PostgreSQL 15.1, 2022.

[5] Е. Рогов, PostgreSQL изнутри, 1-е ред., Москва: ДМК Пресс, 2023, p. 662 .

[6] Б. А. Новиков, Е. А. Горшкова и Н. Г. Графеева, Основы технологии баз данных, 2-е ред., Москва: ДМК пресс, 2020, p. 582.