Лабораторная работа №8
Управление данными

Техническая документация по управлению данными на языке SQL (модификация данных): https://postgrespro.ru/docs/postgrespro/15/dml

Назначение любой БД заключается в работе с данными, под которой понимается ввод новых данных, их изменение и удаление, а также выборка данных в соответствии с некоторыми критериями.

Большинство БД не хранят статистические данные, которыми приходится непрерывно управлять. PostgreSQL предлагает следующие способы модификации данных в таблицах:

- **INSERT INTO** – добавление одной или нескольких записей в таблицу базы данных.
- **INSERT INTO … SELECT …**– копирование результата выполнения запроса в таблицу.
- **COPY** – закачка исходных данных, хранящихся в файлах.
- **UPDATE** – обновление записей таблицы базы данных.
- **DELETE** – удаление записей таблицы базы данных.

1. Вставка записей

С помощью команды INSERT осуществляется вставка новых записей в таблицу, но не модифицирование уже существующих.

Эта команда имеет несколько синтаксических форм:

- ***однострочная инструкция*** INSERT применяется для добавления одной записи в таблицу и имеет формат:

  INSERT INTO имя_таблицы [(список_полей)]
  VALUES ({ DEFAULT | значение } [,…n]);

  INSERT INTO имя_таблицы DEFAULT VALUES;

- ***многострочная инструкция*** INSERT добавляет в целевую таблицу несколько строк, являющихся результатом выборки данных:

  INSERT INTO имя_таблицы [(список_полей)]

  команда_SELECT;

  Параметры:
  имя_таблицы, в которую будут вставлены данные.
  список_полей указывает имена полей, разделенных запятыми, в которые будет производиться вставка данных. **ВАЖНО**: в списке полей должны присутствовать все поля, для которых запрещены Null-значения и не определены значения по умолчанию. Запрещено указание вычисляемых полей. Если аргумент список_полей опущен, то данные будут вставляться последовательно во все поля таблицы, начиная с первого.

Набор вставляемых данных определяется в параметре VALUES, причем количество аргументов и их значения должны соответствовать количеству полей в списке и их типам, а порядок аргументов должен соответствовать порядку полей.

Зарезервированное слово DEFAULT VALUES указывает на то, что запись будет содержать только значения по умолчанию.

Например, в таблицу Student (StudentID, FirstName, LastName, BirthDate), будет добавлена запись с заданными значениями полей, а поле счетчика – сгенерировано автоматически:

```
INSERT INTO Student (FirstName, LastName, BirthDate)
VALUES ( 'Иван', 'Иванов', '01.01.1997' );
```

Выполнение команды

```
INSERT INTO Student DEFAULT VALUES;
```

приведет к ошибке, т. к. не для всех полей указаны значения по умолчанию.

При использовании команды SELECT в таблицу вставляется несколько записей, являющихся результатом выборки данных других таблиц. Однако если хотя бы одна из вставляемых записей нарушает ограничения целостности таблицы, то вся команда INSERT отменяется.

Одна команда может вставить сразу несколько строк. ***Например,***

```
INSERT INTO products (ProductId, Name, Price) VALUES
    (1, 'Cheese', 220),
    (2, 'Bread', 45),
    (3, 'Milk', 70);
```

Например, следующей командой в таблицу Student вставляются данные из таблицы Employee базы данных AdventureWorks. При этом из таблицы Employee берутся данные только о сотрудниках, родившихся в октябре:

```
INSERT INTO Student (FirstName, LastName, BirthDate)
    SELECT FirstName, LastName, BirthDate
    FROM Employee
    WHERE EXTRACT(month from BirthDate)=10;
```

Задание 1.

А) Внесите не менее четырех записей в таблицу **Группа (SGroup)** БД «Факультет» с помощью команды INSERT. Проверьте результат.

Б) Внесите сведения о шести студентах в таблицу **Студент (Student)** БД «Факультет» с помощью команды INSERT. Среди этих студентов должны быть три студента 4 курса. Проверьте результат.

В) Аналогичным образом с помощью команды INSERT внесите по четыре и более записей во **ВСЕ** остальные таблицы БД «Факультет». Самостоятельно продумайте порядок заполнения таблиц.

Г) Проверьте результат и выведите на экран содержимое всех таблиц БД «Факультет».

Указание: для того чтобы посмотреть результаты выполнения заданий, к скрипту добавьте команду выборки всех значений из таблицы:

```
SELECT * FROM имя_просматриваемой_таблицы;
```

Задание 2.

В таблицу **Выпускник (Graduate)** Вашей базы данных выгрузите сведения **о студентах 4 курса** из таблицы Студент (Student), при этом поле e-mail таблицы Выпускник оставьте пустым, а вместо трех полей Фамилия, Имя, Отчество получите новое поле, содержащее фамилию, имя и отчество студента через пробел.

Указание: предварительно в Вашей базе данных **с помощью скрипта на языке SQL** создайте пустую таблицу **Выпускник (Graduate)** по структуре схожую с таблицей Студент (Student). При этом вместо первичного ключа Номер зачетки определите первичный ключ ВыпускникID, заполняемый автоматически целыми значениями, вместо трех полей Фамилия, Имя, Отчество создайте поле ФИО. Никакие другие ограничения на поля новой таблицы накладывать **НЕ** нужно. Для объединения нескольких строк в одну используйте оператор конкатенации «| |»,

2. Загрузка данных из файла. Выгрузка данных в файл

Команда COPY используется для копирования данных между файлом и таблицей, как правило, применяется для загрузки / выгрузки больших объемов данных.

COPY перемещает данные между таблицами Postgres и обычными файлами в файловой системе. COPY TO копирует содержимое таблицы **в файл**, а COPY FROM – **из файла** в таблицу (добавляет данные к тем, что уже содержались в таблице). COPY TO может также скопировать результаты запроса SELECT.

Если указывается список столбцов, COPY TO копирует в файл только данные указанных столбцов, а COPY FROM вставляет каждое поле из файла в соответствующий ему по порядку столбец из указанного списка. В случае отсутствия в этом списке каких-либо столбцов таблицы при COPY FROM они получают значения по умолчанию.

Имеет следующий синтаксис:

```
COPY имя_таблицы [ ( имя_столбца [, ...] ) ]
    FROM 'имя_файла'
    [ [ WITH ] ( параметр [, ...] ) ]
    [ WHERE условие ]

COPY { имя_таблицы [ ( имя_столбца [, ...] ) ] | ( запрос ) }
    TO 'имя_файла'
    [ [ WITH ] ( параметр [, ...] ) ]
```

имя_таблицы, возможно, дополненное схемой.

имя_столбца – необязательный список столбцов, данные которых будут копироваться.

запрос – команда SELECT, VALUES, INSERT, UPDATE или DELETE, результаты которой будут скопированы. Запрос должен заключаться в скобки.

имя_файла – путь входного или выходного файла. Путь входного файла может быть абсолютным или относительным, но путь выходного должен быть только абсолютным. Пользователям Windows следует использовать формат E'' и продублировать каждую обратную черту в пути файла.

Здесь допускается параметр:

FORMAT имя_формата – выбирает формат чтения или записи данных: text (текстовый), csv (значения, разделённые запятыми, Comma Separated Values) или binary (двоичный). По умолчанию выбирается формат text.

DELIMITER 'символ_разделитель' – разделяющий столбцы в строках файла. По умолчанию это символ табуляции в текстовом формате и запятая в формате CSV. Задаваемый символ должен быть однобайтовым. Для формата binary этот параметр не допускается.

HEADER [boolean | MATCH] – указывает, что файл содержит строку заголовка с именами каждого столбца в файле. При выводе первая строка будет содержать имена столбцов из таблицы. При вводе первая строка отбрасывается, если для этого параметра установлено значение true (или равнозначное логическое значение). Если для этого параметра установлено значение MATCH, имена столбцов (и их количество) в строке заголовка должны совпадать с фактическими именами столбцов таблицы по порядку, иначе возникнет ошибка. Этот параметр не допускается при использовании формата binary. Параметр MATCH действителен только для команд COPY FROM.

WHERE условие – любое выражение, выдающее результат типа boolean. Строки, не удовлетворяющие этому условию, добавляться в таблицу не будут.

Про особенности работы с CSV-файлами и текстовыми файлами более подробно можно ознакомиться здесь: https://postgrespro.ru/docs/postgrespro/15/sql-copy

В примере ниже показано, копирование внешних данных в некоторую временную таблицу с указанием полей для заполнения, пути к файлу со списком студентов. CSV файл в качестве разделителя использует запятую, а также содержит заголовок, поэтому первая строка файла с данными не загружается.

```
COPY Tmp(LastName,FirstName,Patronymic,BDate)
FROM 'd:\Student1.csv'
DELIMITER ','
CSV HEADER;
```

Задание 3.

А) Заполните таблицу **Аудитория (Classroom)** БД «Факультет» из CSV файла, содержащего данные не менее чем о четырех аудиториях, расположенных в двух разных корпусах.

Б) Заполните таблицу **Дисциплина (Subject)** БД «Факультет» из CSV файла (не менее 6 записей), первая строка в котором содержит строку заголовка с именами каждого столбца в файле.

Указание: не забывайте, что значения для поля Код дисциплины генерируются автоматически, а общий объем часов рассчитывается по формуле. При сохранении файла в формате CSV обязательно указывайте расширение .csv в имени файла, а также выбирайте кодировку UTF-8, так как она поддерживает разнообразные символы и языки.

В) Создайте CSV-файл с оценками студентов (не менее 15 записей). Заполните таблицу **Оценка (Mark)** БД «Факультет CSV файла» из подготовленного CSV файла.

Г) Выгрузите в CSV файл все данные из таблицы **Преподаватель (Teacher).**

Д) Выгрузите в CSV файл данные по аудиториям с проектором.

3. Изменение записей

Обновление записей осуществляется с помощью команды UPDATE, которая позволяет выполнять как простое обновление данных в поле, так и сложные модификации данных во множестве записей таблицы.

При выполнении данной команды обновляются записи только одной таблицы базы данных, если же ни одна запись не обновлена, то генерируется сообщение об ошибке.

Данная команда имеет следующий синтаксис:

```
UPDATE имя_таблицы
    SET { имя_поля = новое_значение | DEFAULT } [,…n]
    [ FROM имя_таблицы ]
    [ WHERE условие_отбора ];
```

В результате выполнения команды происходит обновление одного или нескольких полей, указанных в разделе SET.

Значение поля является значением некоторого выражения, содержащего стандартные функции, переменные и поля таблиц, заданных в разделе FROM.

С помощью зарезервированного слова DEFAULT полю присваивается значение, определенное по умолчанию.

Изменению подвергаются все записи таблицы, удовлетворяющие условию отбора, указанному в разделе WHERE, а его правила записи аналогичны правилам в команде SELECT.

Например, чтобы задать новый номер телефона у студента по фамилии *Иванов* необходимо выполнить следующую команду:

```
UPDATE Student
    SET Phone = '+79231353535'
    WHERE LastName = 'Иванов' and StudentId = 123321;
```

Задание 4. В БД «Факультет» в таблице **Преподаватель (Teacher)** для поля «Ученая степень» установите значение «без степени» для всех записей, у которых в этом поле содержатся неопределенные значения или значения отличные от «кандидат наук» или «доктор наук».

Задание 5. В БД «Факультет» в таблице **Дисциплина (Subject)** увеличьте Количество лекционных часов и Количество лабораторных часов на 10% для дисциплины с определенным наименованием (например, Базы данных).

Указание: новое количество часов округляйте в большую сторону до целого значения.

4. Удаление записей

Удаление данных из таблицы осуществляется построчно, при этом можно выполнить удаление как одной записи, так и нескольких, удовлетворяющих некоторому условию.

Синтаксис данной команды:

```
DELETE [ FROM ] имя_таблицы
    [ USING элемент_FROM [ , … ] ]
    [ WHERE условие_отбора ];
```

Параметр условие_отбора определяет набор записей, которые необходимо удалять. Правила записи данного раздела аналогичны правилам этого раздела команды SELECT.

PostgreSQL позволяет ссылаться на столбцы других таблиц в условии WHERE, когда эти таблицы перечисляются в предложении USING. Например, удалить все фильмы определённого продюсера можно так:

```
DELETE FROM films USING producers
WHERE producer_id = producers.id AND producers.name = 'foo';
```

Для удаления всех записей некоторой таблицы используется команда:

```
TRUNCATE [ TABLE ] имя_таблицы [ , … ]
[ CASCADE | RESTRICT ];
```

Команда TRUNCATE быстро удаляет все записи, однако структура таблицы сохраняется, как и связанные с ней объекты. Она действует так же, как безусловная команда DELETE для каждой таблицы, но гораздо быстрее, так как она фактически не сканирует таблицы. Более того, она немедленно высвобождает дисковое пространство. Наиболее полезна она для больших таблиц.

Параметры:

имя_таблицы, возможно, дополненное схемой.

CASCADE – автоматически опустошать все таблицы, ссылающиеся по внешнему ключу на заданные таблицы, или на таблицы, затронутые в результате действия CASCADE.

RESTRICT – отказать в опустошении любых таблиц, на которые по внешнему ключу ссылаются другие таблицы, не перечисленные в этой команде. Это поведение по умолчанию.

Задание 6. Удалите из таблицы **Преподаватель (Teacher)** тех, кому в этом году исполнилось более 65 лет.

Указание: если таких преподавателей нет, то сначала предусмотрите добавление записей, удовлетворяющих указанному условию. Также предусмотрите случай, чтобы преподаватель, удаляемый в последствии, изначально проводил занятия у групп и / или проставлял оценки студентам.

Задание 7. Удалите **все** записи из таблицы **Выпускник (Graduate)**.

Задание 8. Удалите все занятия у групп 4-го курса.

Задание 9. Удалите из таблицы **Студент (Student)** записи о студентах, у которых одновременно не указаны реальные Адрес, Сотовый телефон, e-mail и Год поступления.

Пояснение: под реальными данными имеются в виду данные отличные Null-значений и от значений по умолчанию.