Лекция 3

Agenda (повестка дня)

- Язык манипулирования данными
- Запросы
- Табличные выражения

DML Commands (Data Manipulation Language)

- SELECT – для запроса данных в базе данных;
- INSERT – вставить данные в таблицу;
- UPDATE – обновить данные в таблице;
- DELETE – удалить данные из таблицы.

DML команда: INSERT

- Основной синтаксис оператора INSERT INTO:

```
INSERT INTO table_name(column1, column2, …)
VALUES (value1, value2, …);
```

- Примеры операторов INSERT PostgreSQL

```
INSERT INTO links (url, name)
VALUES('https://www.postgresqltutorial.com','PostgreSQL Tutorial');
```

Команды DML: INSERT несколько строк

- Чтобы вставить несколько строк в таблицу с помощью одного оператора INSERT, используйте следующий синтаксис:

```
INSERT INTO table_name (column_list)
VALUES
    (value_list_1),
    (value_list_2),
    ...
    (value_list_n);
```

```
INSERT INTO
    links (url, name)
VALUES
    ('https://www.google.com','Google'),
    ('https://www.yahoo.com','Yahoo'),
    ('https://www.bing.com','Bing');
```

DML команда: UPDATE

- Оператор UPDATE позволяет изменять данные в таблице. Следующее иллюстрирует синтаксис оператора UPDATE:

```
UPDATE table_name
SET column1 = value1,
    column2 = value2,
    ...
WHERE condition;
```

- Следующий оператор использует оператор UPDATE для обновления курса с id 3. Он изменяет значение published_date с NULL на «2020-08-01».

```
UPDATE courses
SET published_date = '2020-08-01'
WHERE course_id = 3;
```

DML команда: DELETE

- Оператор PostgreSQL DELETE позволяет удалить одну или несколько строк из таблицы.
- Ниже показан основной синтаксис оператора DELETE:

```
DELETE FROM table_name
WHERE condition;
```

- Следующий оператор использует оператор DELETE для удаления одной строки с id 8 из таблицы ссылок.

```
DELETE FROM links
WHERE id = 8;
```

Запрос данных(Querying Data)

- Процесс извлечения или команда для получения данных из базы данных называется запросом.
- В SQL команда SELECT используется для задания запросов. Общий синтаксис команды SELECT:

```
SELECT column1, column2, ...
FROM table_name;
```

Примеры (SELECT)

- Следующий оператор SQL выбирает все столбцы из таблицы «Customers»:

```
SELECT * FROM Customers;
```

Следующий оператор SQL выбирает столбцы «CustomerName» и «City» из таблицы «Customers»:

```
SELECT CustomerName, City FROM Customers;
```

Примеры (SELECT DISTINCT) (отдельный)

- Инструкция SELECT DISTINCT используется для возврата только отдельных (разных) значений.
- Следующий оператор SQL выбирает только значения DISTINCT из столбца «Country» в таблице «Customers»:

```
SELECT DISTINCT Country FROM Customers;
```

- Следующий оператор SQL перечисляет количество различных (distinct) стран-клиентов:

```
SELECT COUNT(DISTINCT Country) FROM Customers;
```

Column Alias (Псевдоним столбца)

Позволяет присвоить столбцу или выражению в списке выбора оператора SELECT временное имя.

Существует временно во время выполнения запроса

Синтаксис использования псевдонима столбца:

```
SELECT column_name [AS] alias_name FROM table_name;
```

For example:

SELECT a AS value, b + c AS sum FROM customers

Ключевое слово AS является необязательным, но только в том случае, если имя нового столбца не соответствует ни одному ключевому слову PostgreSQL. Чтобы избежать случайного совпадения с ключевым словом, вы можете заключить имя столбца в двойные кавычки.

Для защиты от возможного добавления ключевых слов в будущем рекомендуется всегда либо писать AS, либо заключать имя выходного столбца в двойные кавычки.

Table Expressions (Табличные выражения)

- Табличное выражение вычисляет таблицу.
- Табличное выражение содержит предложение FROM, за которым при необходимости следуют предложения WHERE, GROUP BY и HAVING.
- Необязательные предложения WHERE, GROUP BY и HAVING в табличном выражении определяют конвейер последовательных преобразований, выполняемых над таблицей, полученной в предложении FROM. Все эти преобразования создают виртуальную таблицу, которая предоставляет строки, которые передаются в список выбора для вычисления выходных строк запроса.

WHERE

Основной синтаксис оператора SELECT с предложением WHERE:

```
SELECT <column1_name>, <column2_name>, <columnN_name>
FROM <table_name>
WHERE [search_condition]
```

WHERE [search_condition] — любое выражение значения, которое возвращает значение логического типа.

Операторы в предложении WHERE

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

Пример использования предложения WHERE с оператором равенства (=)

- Следующий оператор использует предложение WHERE для клиентов, чьи имена — Джейми:

```
SELECT
        last_name,
        first_name
FROM
        customer
WHERE
        first_name = 'Jamie';
```

	last_name character varying (45)	first_name character varying (45)
1	Rice	Jamie
2	Waugh	Jamie

Использование предложения WHERE с примером оператора AND

- В следующем примере выполняется поиск клиентов, чьи имя и фамилия — Джейми и Райс, с помощью логического оператора И для объединения двух логических выражений:

```
SELECT
        last_name,
        first_name
FROM
        customer
WHERE
        first_name = 'Jamie' AND
        last_name = 'Rice';
```

	last_name character varying (45)	first_name character varying (45)
1	Rice	Jamie

Использование предложения WHERE с примером оператора OR

- В этом примере выполняется поиск клиентов с фамилией Родригес или именем Адам с помощью оператора OR:

```
SELECT
        first_name,
        last_name
FROM
        customer
WHERE
        last_name = 'Rodriguez' OR
        first_name = 'Adam';
```

	first_name character varying (45)	last_name character varying (45)
1	Laura	Rodriguez
2	Adam	Gooch

Использование предложения WHERE с примером оператора IN

- Если вы хотите сопоставить строку с любой строкой в списке, вы можете использовать оператор IN.
- Например, следующий оператор возвращает клиентов, чье имя — Ann или Anne или Annie:

```
SELECT
        first_name,
        last_name
FROM
        customer
WHERE
        first_name IN ('Ann','Anne','Annie');
```

	first_name character varying (45)	last_name character varying (45)
1	Ann	Evans
2	Anne	Powell
3	Annie	Russell

Использование предложения WHERE с примером оператора LIKE

- Чтобы найти строку, соответствующую заданному шаблону, вы используете оператор LIKE. Следующий пример возвращает всех клиентов, чьи имена начинаются со строки Ann:
- % называется подстановочным знаком, который соответствует любой строке. Шаблон «Ann%» соответствует л нающейс

```
SELECT
        first_name,
        last_name
FROM
        customer
WHERE
        first_name LIKE 'Ann%'
```

	first_name character varying (45)	last_name character varying (45)
1	Anna	Hill
2	Ann	Evans
3	Anne	Powell
4	Annie	Russell
5	Annette	Olson

Использование предложения WHERE с примером оператора BETWEEN

- В следующем примере клиенты, чьи имена начинаются с буквы A и содержат от 3 до 5 символов, находятся с помощью оператора BETWEEN. Оператор BETWEEN возвращает true, если значение находится в диапазоне значений.

```
SELECT
        first_name,
        LENGTH(first_name) name_length
FROM
        customer
WHERE
        first_name LIKE 'A%' AND
        LENGTH(first_name) BETWEEN 3 AND 5
ORDER BY
        name_length;
```

	first_name character varying (45)	name_length integer
1	Amy	3
2	Ann	3
3	Ana	3
4	Andy	4
5	Anna	4
6	Anne	4
7	Alma	4
8	Adam	4
9	Alan	4
10	Alex	4
11	Angel	5
12	Agnes	5
13	Andre	5

Пример использования предложения WHERE с оператором неравенства (<>)

- В этом примере выполняется поиск клиентов, чьи имена начинаются с «Bra», а фамилии не являются «Motley»:
- Обратите внимание, что вы можете использовать операторы != и <> как взаимозаменяемые, поскольку они эквивалентны.

```
SELECT
        first_name,
        last_name
FROM
        customer
WHERE
        first_name LIKE 'Bra%' AND
        last_name <> 'Motley';
```

	first_name character varying (45)	last_name character varying (45)
1	Brandy	Graves
2	Brandon	Huey
3	Brad	Mccurdy

ORDER BY

- Предложение ORDER BY позволяет сортировать строки, возвращаемые предложением SELECT, в порядке возрастания или убывания на основе выражения сортировки.
- Ниже показан синтаксис предложения ORDER BY:

```
SELECT
        select_list
FROM
        table_name
ORDER BY
        sort_expression1 [ASC | DESC],
        ...
        sort_expressionN [ASC | DESC];
```

Использование предложения ORDER BY для сортировки строк по одному примеру столбца

- В следующем запросе используется предложение ORDER BY для сортировки клиентов по именам в порядке возрастания:

```
SELECT
        first_name,
        last_name
FROM
        customer
ORDER BY
        first_name ASC;
```

	first_name character varying (45)	last_name character varying (45)
1	Aaron	Selby
2	Adam	Gooch
3	Adrian	Clary
4	Agnes	Bishop
5	Alan	Kahn
6	Albert	Crouse
7	Alberto	Henning
8	Alex	Gresham
9	Alexander	Fennell
10	Alfred	Casillas
11	Alfredo	Mcadams
12	Alice	Stewart
13	Alicia	Mills

Использование предложения ORDER BY для сортировки строк по одному столбцу в порядке убывания.

- Следующий оператор выбирает имя и фамилию из таблицы клиентов и сортирует строки по значениям в столбце фамилии в порядке убывания:

```
SELECT
	first_name,
	last_name
FROM
	customer
ORDER BY
	last_name DESC;
```

	first_name character varying (45)	last_name character varying (45)
1	Cynthia	Young
2	Marvin	Yee
3	Luis	Yanez
4	Brian	Wyman
5	Brenda	Wright
6	Tyler	Wren
7	Florence	Woods
8	Lori	Wood
9	Virgil	Wofford
10	Darren	Windham

Использование предложения ORDER BY для сортировки строк по нескольким столбцам

- Следующий оператор выбирает имя и фамилию из таблицы клиентов и сортирует строки по имени в порядке возрастания и фамилии в порядке убывания:

```
SELECT
        first_name,
        last_name
FROM
        customer
ORDER BY
        first_name ASC,
        last_name DESC;
```

	first_name character varying (45)	last_name character varying (45)
321	Kathleen	Adams
322	Kathryn	Coleman
323	Kathy	James
324	Katie	Elliott
325	Kay	Caldwell
326	Keith	Rico
327	Kelly	Torres
328	Kelly	Knott
329	Ken	Prewitt
330	Kenneth	Gooden
331	Kent	Arsenault
332	Kevin	Schuler