Лекция 9

Модификация данных

Работа с данными

- Добавление данных
- Удаление данных
- Обновление данных

Целостность данных

- **Целостность данных** – это полнота, точность и единообразие данных (соответствие бизнес-требованиям)
- Для поддержания целостности данных в реляционных БД используется ряд инструментов:
  - Ограничения целостности, которые позволяют применять практические правила к данным в таблицах и гарантировать точность и надежность данных
  - Триггера – интегрированный пользовательский код, который выполняется в ответ на определенные операции в БД

Добавление данных

```
INSERT INTO <имя таблицы>[(<имя столбца>,...)]
{DEFAULT VALUES | VALUES (<значение столбца>,…)}
|<выражение запроса>
```

- Столбец **<u>может быть не включен</u>** в список, если он удовлетворяет одному из условий:
  - Является генерируемым:
    - вычисляемый столбец на основе выражения,
    - столбец идентификации
    - столбец SERIAL
  - Имеет заданное свойство DEFAULT
  - Допускает возможность значения NULL

Добавление данных через конструктор VALUES

```
INSERT INTO public.client
       ("Lname" ,"Fname" ,"City" ,"RegDate")
VALUES
      ('Petrov' , 'Ivan' , 'Moscow' , current_date),
      ('Semenov' , 'Ivan' , DEFAULT , DEFAULT);
```

```
INSERT INTO public.client ("Lname" )
VALUES ('Petrova'), ('Semenova');
```

	ClientID	Lname	Fname	City	RegDate
1	1	Petrov	Ivan	Moscow	2022-10-12
2	2	Semenov	Ivan	SPb	2022-10-12
3	3	Petrova	[NULL]	SPb	2022-10-12
4	4	Semenova	[NULL]	SPb	2022-10-12

public.client

Column Name	Properties
ClientID	PK, serial, NOT NULL
Lname	varchar(20), NOT NULL
Fname	varchar(20), NULL
City	varchar(20), DEFAULT 'SPb'
RegDate	date, DEFAULT current_date

Добавление данных на основе выборки

```
INSERT INTO public.client
            ("Lname" ,"Fname" ,"City" ,"RegDate")
SELECT
          lastname, firstname, city, current_date
FROM "HR"."Employees"
WHERE  empid > 4;
```

Results | Messages

	lastname	firstname	city	(No column name)
1	Buck	Sven	London	2022-03-26 13:18:44.300
2	Suurs	Paul	London	2022-03-26 13:18:44.300
3	King	Russell	London	2022-03-26 13:18:44.300
4	Cameron	Maria	Seattle	2022-03-26 13:18:44.300
5	Dolgopyatova	Zoya	London	2022-03-26 13:18:44.300
6	Johnson	Test 1	Ljubljana	2022-03-26 13:18:44.300

	ClientID	Lname	Fname	City	RegDate
1	1	Petrov	Ivan	Moscow	2022-10-12
2	2	Semenov	Ivan	SPb	2022-10-12
3	3	Petrova	[NULL]	SPb	2022-10-12
4	4	Semenova	[NULL]	SPb	2022-10-12
5	5	Cameron	Maria	Seattle	2022-10-12
6	6	Buck	Sven	London	2022-10-12
7	7	Suurs	Paul	London	2022-10-12
8	8	King	Russell	London	2022-10-12
9	9	Dolgopyatova	Zoya	London	2022-10-12
10	10	Johnson	Test 1	Ljubljana	2022-10-12

Добавление данных в столбец **serial**

```
INSERT INTO public.client
("Lname", "RegDate")
VALUES      ('Petrova', DEFAULT);
```

```
INSERT INTO public.client
("ClientID", "Lname", "RegDate")
VALUES  (16, 'Frolova', DEFAULT);
```

```
INSERT INTO public.client
("ClientID", "Lname", "RegDate")
VALUES  (DEFAULT, 'Frolov', DEFAULT);
```

public.client

Column Name	Properties
ClientID	PK, serial, NOT NULL
Lname	varchar(20), NOT NULL
Fname	varchar(20), NULL
City	varchar(20), DEFAULT 'SPb'
RegDate	date, DEFAULT current_date

ClientID	Lname	Fname	City	RegDate
1	Petrov	Ivan	Moscow	2022-10-12
2	Semenov	Ivan	SPb	2022-10-12
3	Petrova	[NULL]	SPb	2022-10-12
10	Johnson	Test 1	Ljubljana	2022-10-12
11	Petrova	[NULL]	SPb	2022-10-12
16	Frolova	[NULL]	SPb	2022-10-12
12	Frolov	[NULL]	SPb	2022-10-13

Добавление данных в столбец IDENTITY

```
CREATE TABLE cars (
car_id INT GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1),
brand_name VARCHAR NOT NULL
);
```

- ***GENERATED ALWAYS AS IDENTITY***
  - PostgreSQL автоматически генерирует целочисленное значение для столбца идентификаторов
  - Пользователю не разрешено вставлять или обновлять значение в столбце – возникнет ошибка
  - Чтобы исправить эту ошибку, можно использовать **OVERRIDING SYSTEM VALUE**

```
INSERT INTO CARS(car_id, brand_name)
OVERRIDING SYSTEM VALUE
VALUES(5, 'Tesla');
```

Добавление данных в столбец IDENTITY

```
CREATE TABLE cars (
car_id INT GENERATED BY DEFAULT AS IDENTITY,
brand_name VARCHAR NOT NULL
);
```

- ***GENERATED BY DEFAULT***
  - PostgreSQL автоматически генерирует целочисленное значение для столбца идентификаторов.
  - Пользователь может указать значение для вставки или обновления

```
INSERT INTO CARS(car_id, brand_name)
VALUES (DEFAULT, 'Jeep'),
       (5, 'Tesla');
```

Изменение и удаление IDENTITY

- Изменение характеристики столба идентификации – **ALTER TABLE … ALTER COLUMN … SET GENERATED {ALWAYS|BY DEFAULT}**

```
ALTER TABLE CARS
ALTER COLUMN car_id
SET GENERATED BY DEFAULT;
```

- Удаление ограничения GENERATED AS IDENTITY (без удаления столбца) - **ALTER TABLE … DROP IDENTITY**

```
ALTER TABLE CARS
ALTER COLUMN car_id
DROP IDENTITY IF EXISTS;
```

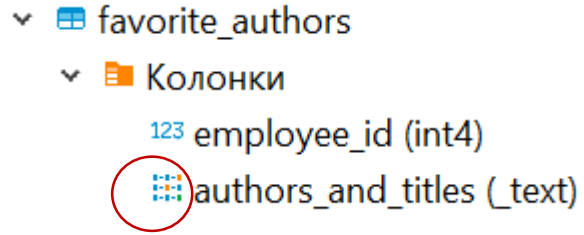
Вставка значений в столбцы массива

- Для вставки нескольких значений в один столбец используется синтаксис описания константы массива:
  - `'{ "text1" [, ...] }'` -- Массив символьных строк
  - `'{ числовой [, ...] }'` -- числовой массив
  - `ARRAY ['эл-т1', 'эл-т2',….]` -- с помощью конструктора ARRAY
- Столбец может быть определен как массив любого произвольного типа (включая типы boolean , date и time)
- ***Внимание***:
  - При работе с массивом символьных строк требуются **двойные кавычки**
  - Если вы обычно используете одинарные кавычки для описания значения в контексте, отличном от массива (например, со строковой константой или значением метки времени), для этого значения в константе массива следует также использовать **двойные кавычки**

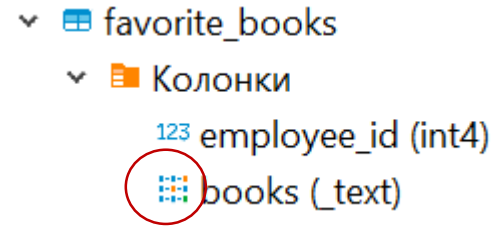
Пример

```
INSERT INTO favorite_books
VALUES (102, E'{"The Hitchhiker\'s Guide to the Galaxy"}'),
       (103, '{"The Hobbit", "Kitten, Squared"}');
```

	employee_id	books
1	103	{The Hobbit,Kitten, Squared}
2	102	{The Hitchhiker's Guide to the Galaxy}



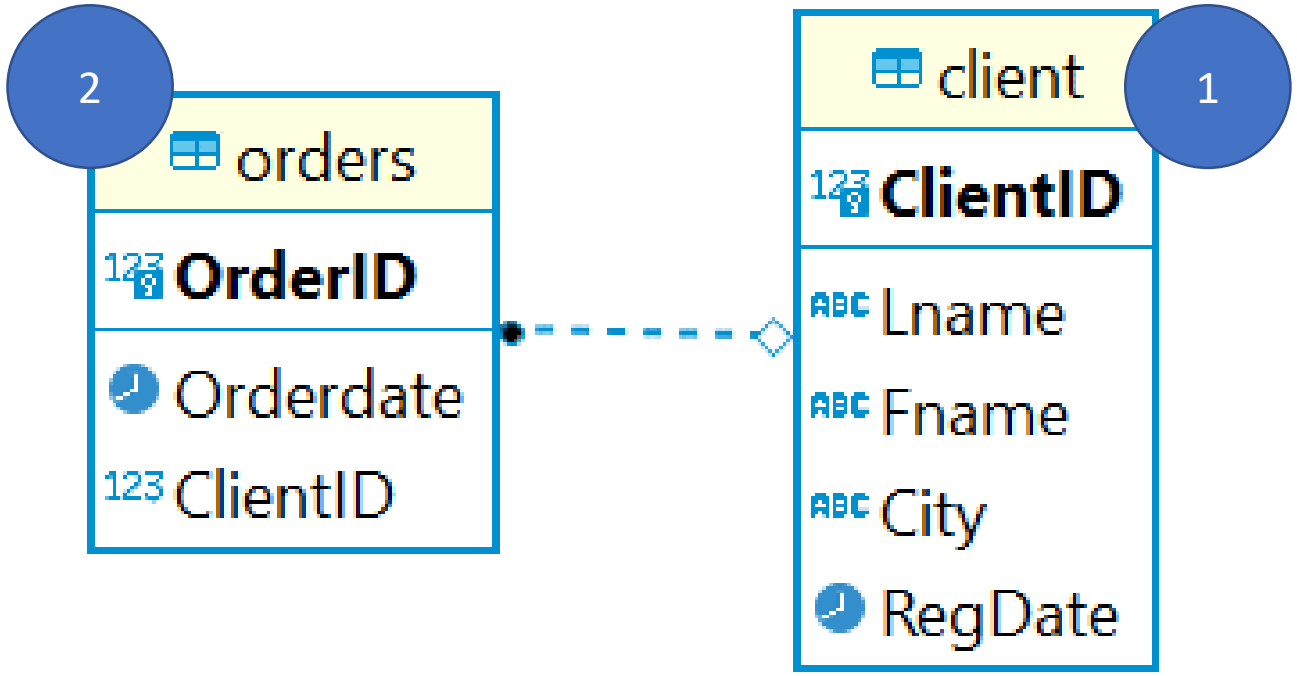

```
INSERT INTO favorite_authors
VALUES (102,
        '{{"J.R.R. Tolkien", "The Silmarillion"},
          {"Charles Dickens", "Great Expectations"},
          {"Ariel Denham", "Attic Lives"}}'
);
```



	employee_id	authors_and_titles
1	102	{{J.R.R. Tolkien,The Silmarillion},{Charles Dickens,Great Expectations},{Ariel Denham,Attic Lives}}

Последовательность добавления

- Если между таблицами определена связь:
  - Сначала добавляются записи в таблицу **PK**
  - Затем связанные записи могут быть добавлены в таблицу **FK**



Ошибки добавления

- При нарушении ограничений:
  - PRIMARY KEY
  - UNIQUE
  - CHECK
  - NOT NULL
  - FOREIGN KEY

Разрешение конфликта вставки

```
INSERT INTO <имя таблицы>[(<имя столбца>,...)]
{DEFAULT VALUES | VALUES (<значение столбца>,…)}
|<выражение запроса>
[ ON CONFLICT [ объект_конфликта ] действие_при_конфликте ]
```

- **ON CONFLICT** позволяет задать действие, заменяющее возникновение ошибки при нарушении ограничения уникальности или ограничения-исключения
  - проверка выполняется для каждой отдельной добавляемой строки
- Возможные действия:
  - ***DO NOTHING*** - не делать ничего
  - ***DO UPDATE*** - произвести изменение. Необходимо указать точные детали операции UPDATE, выполняемой в случае конфликта

Разрешение конфликта вставки

- Добавить клиентов или изменить существующие данные должным образом.
- Для обращения к значениям, изначально предлагаемым для добавления, используется специальная таблица **excluded**

```
INSERT INTO public.client ("ClientID","Lname")
    VALUES (5, 'Gizmo'), (6, 'Sergeev')
    ON CONFLICT ("ClientID")
    DO UPDATE SET "Lname" = EXCLUDED."Lname";
```

	ClientID	Lname	Fname	City	RegDate
1	1	Petrov	Ivan	Moscow	2022-05-12
2	2	Semenov	Ivan	SPb	2022-05-12
3	3	Petrova	Alice	SPb	2022-05-12
4	4	Semenova	[NULL]	SPb	2022-05-12
5	6	Buck	Sven	London	2022-05-12

	ClientID	Lname	Fname	City	RegDate
1	1	Petrov	Ivan	Moscow	2022-05-12
2	2	Semenov	Ivan	SPb	2022-05-12
3	3	Petrova	Alice	SPb	2022-05-12
4	4	Semenova	[NULL]	SPb	2022-05-12
5	5	Gizmo	[NULL]	SPb	2022-10-12
6	6	Sergeev	Sven	London	2022-05-12

Удаление данных

```
DELETE FROM public.client;
```

SQL Error [23503]: ERROR: update or delete on table "client" violates foreign key constraint "orders_ClientID_fkey" on table "orders"
Подробности: Key (ClientID)=(1) is still referenced from table "orders".

```
DELETE FROM public.client
WHERE "ClientID" = 5;
```

```
DELETE FROM public.client
WHERE "ClientID" IN
(SELECT "ClientID"
 FROM public.orders
 WHERE "Orderdate"<'20220327');
```

SQL Error [23503]: ERROR: update or delete on table "client" violates foreign key constraint "orders_ClientID_fkey" on table "orders"
Подробности: Key (ClientID)=(2) is still referenced from table "orders".

public.orders

OrderID	Orderdate	ClientID
1	2022-03-26	2
2	2022-03-27	3
3	2022-03-28	3
4	2022-03-28	1

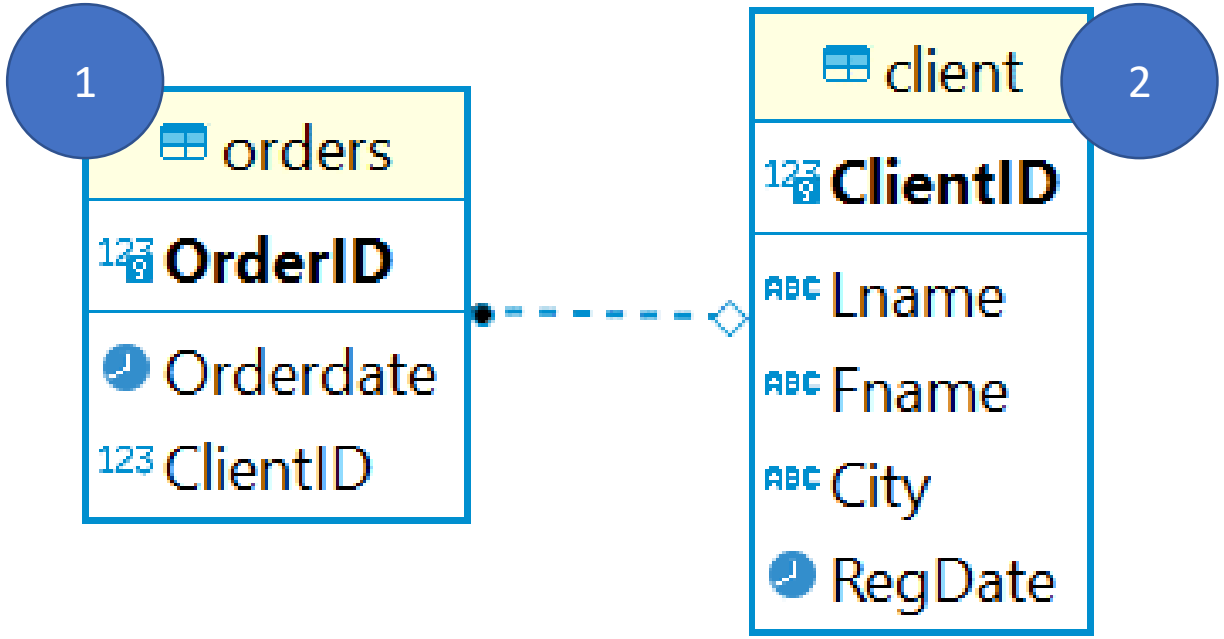
?

public.client

ClientID	Lname	Fname	City	RegDate
1	Petrov	Ivan	Moscow	26.03.2022
2	Semenov	Ivan	SPb	26.03.2022
3	Petrova	NULL	SPb	26.03.2022
4	Semenova	NULL	SPb	26.03.2022
5	Buck	Sven	London	26.03.2022
6	Suurs	Paul	London	26.03.2022
7	King	Russell	London	26.03.2022
8	Cameron	Maria	Seattle	26.03.2022
9	Dolgopyatova	Zoya	London	26.03.2022
10	Johnson	Test 1	Ljubljana	26.03.2022

Последовательность удаления

- Сначала удаляются связанные записи из таблицы FK
- Затем могут быть удалены записи из таблицы PK
  - Если FK поддерживает **<u>каскадное удаление</u>** при удалении записи в таблице PK , связанные записи в таблице FK будут удалены **АВТОМАТИЧЕСКИ**



«Опустошить» таблицу

```
TRUNCATE [ TABLE ] [ ONLY ] имя [ * ] [, ... ]
    [ RESTART IDENTITY | CONTINUE IDENTITY ]
        [ CASCADE | RESTRICT ]
```

- Команда TRUNCATE быстро удаляет все строки из набора таблиц
  - не сканирует таблицы
  - немедленно высвобождает дисковое пространство (не нужно выполнять операцию VACUUM)
  - Наиболее полезна для больших таблиц

```
TRUNCATE TABLE public.client;
```

SQL Error [0A000]: ERROR: cannot truncate a table referenced in a foreign key constraint
Подробности: Table "orders" references "client".
Подсказка: Truncate table "orders" at the same time, or use TRUNCATE ... CASCADE.

```
TRUNCATE TABLE public.client CASCADE;
```

public.orders

OrderID	Orderdate	ClientID
1	2022-03-26	2
2	2022-03-27	3
3	2022-03-28	3
4	2022-03-28	1

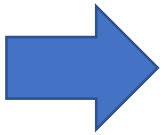
public.client

ClientID	Lname	Fname	City	RegDate
1	Petrov	Ivan	Moscow	26.03.2022
2	Semenov	Ivan	SPb	26.03.2022
3	Petrova	NULL	SPb	26.03.2022
4	Semenova	NULL	SPb	26.03.2022
5	Buck	Sven	London	26.03.2022
6	Suurs	Paul	London	26.03.2022
7	King	Russell	London	26.03.2022
8	Cameron	Maria	Seattle	26.03.2022
9	Dolgopyatova	Zoya	London	26.03.2022
10	Johnson	Test 1	Ljubljana	26.03.2022

Обновление данных

```
UPDATE public.client
SET "RegDate" = "RegDate" - interval '5 mons';
```

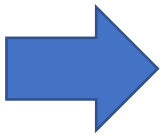
	ClientID	Lname	Fname	City	RegDate
1	1	Petrov	Ivan	Moscow	2022-10-12
2	2	Semenov	Ivan	SPb	2022-10-12
3	3	Petrova	[NULL]	SPb	2022-10-12
4	4	Semenova	[NULL]	SPb	2022-10-12
5	6	Buck	Sven	London	2022-10-12



	ClientID	Lname	Fname	City	RegDate
1	1	Petrov	Ivan	Moscow	2022-05-12
2	2	Semenov	Ivan	SPb	2022-05-12
3	3	Petrova	[NULL]	SPb	2022-05-12
4	4	Semenova	[NULL]	SPb	2022-05-12
5	6	Buck	Sven	London	2022-05-12

```
UPDATE public.client
SET "Fname" = 'Alice'
WHERE "ClientID" = 3;
```

	ClientID	Lname	Fname	City	RegDate
1	1	Petrov	Ivan	Moscow	2022-05-12
2	2	Semenov	Ivan	SPb	2022-05-12
3	3	Petrova	[NULL]	SPb	2022-05-12
4	4	Semenova	[NULL]	SPb	2022-05-12
5	6	Buck	Sven	London	2022-05-12



	ClientID	Lname	Fname	City	RegDate
1	1	Petrov	Ivan	Moscow	2022-05-12
2	2	Semenov	Ivan	SPb	2022-05-12
3	3	Petrova	Alice	SPb	2022-05-12
4	4	Semenova	[NULL]	SPb	2022-05-12
5	6	Buck	Sven	London	2022-05-12

Обновление значений в столбцах массива

- Значения в столбцах массива можно изменить одним из трех способов:
  - **Полная модификация**
    - весь массив может быть заменен новой константой массива
    - в новом массиве не обязательно должно быть столько же значений, сколько в существующем
  - **Модификация среза**
    - набор элементов массива (диапазон между двумя значениями) может быть заменен новой константой массива
    - количество элементов новой константы массива должно совпадать с количеством обновляемых значений
  - **Модификация элемента**
    - отдельный элемент в массиве может быть заменен новой константой базового типа массива
    - необходимо использовать нижний индекс, чтобы указать, какое значение массива следует заменить

Полное изменение массива

	employee_id	books
1	102	{The Hitchhiker's Guide to the Galaxy}

```
UPDATE favorite_books
       SET books= E'{"The Hitchhiker\'s Guide to the Galaxy",
                    "The Restaurant at the End of the Universe"}'
       WHERE employee_id = 102;
```

	employee_id	books
1	102	{The Hitchhiker's Guide to the Galaxy,The Restaurant at the End of the Universe}

Модификация среза

	employee_id	books
1	103	{The Hobbit,Kitten,Squared}

```
UPDATE favorite_books
       SET books[2:3]= E'{"Dogs",
                         "The Restaurants"}'
       WHERE employee_id = 103;
```

	employee_id	books
1	103	{The Hobbit,Dogs,The Restaurants}

Модификация элемента

	employee_id	books[1]
1	102	The Hitchhiker's Guide to the Galaxy
2	103	The Hobbit

```
UPDATE favorite_books
   SET books[1] = E'There and Back Again: A Hobbit\'s Holiday'
 WHERE books[1] = 'The Hobbit';
```

	employee_id	books[1]
1	102	The Hitchhiker's Guide to the Galaxy
2	103	There and Back Again: A Hobbit's Holiday

Ошибки обновления

- При нарушении ограничений:
  - UNIQUE
  - CHECK
  - NOT NULL
  - FOREIGN KEY
- При изменении значения PK
  - Если **FK** поддерживает **каскадное обновление** при изменении значения **PK** , в связанных записях значения соответствующих **FK** будут обновлены **АВТОМАТИЧЕСКИ**

Возврат данных из изменённых строк

- В процессе обработки записей бывает полезно получать данные из модифицируемых строк с помощью предложения **RETURNING**
  - В предложении RETURNING допускается то же содержимое, что и в выходном списке команды SELECT
- ***При добавлении данных (INSERT)***
  - RETURNING возвращает строки в том виде, в каком они были вставлены.
  - Это может быть полезно при использовании вычисляемых значений по умолчанию
- ***При изменении данных (UPDATE)***
  - RETURNING возвращает новое содержимое изменённых строк
- ***При удалении данных (DELETE)***
  - RETURNING возвращает содержимое удалённых строк
- Если для целевой таблицы заданы триггеры, **RETURNING** возвращает данные из строк, изменённых триггерами

Пример

```
INSERT INTO public.client
       ("Lname" ,"Fname" ,"City" ,"RegDate")
VALUES
('Petrov' , 'Ivan' , 'Moscow' , current_date),
('Semenov' , 'Ivan' , 'SPb' , DEFAULT)
RETURNING *;
```

	ClientID	Lname	Fname	City	RegDate
1	3	Petrov	Ivan	Moscow	2022-12-14
2	4	Semenov	Ivan	SPb	2022-12-14

```
UPDATE public.client
SET "RegDate" = "RegDate" - interval '5 mons'
where "ClientID" = 3
RETURNING "ClientID", "RegDate";
```

	ClientID	RegDate
1	3	2022-07-14

```
delete from  public.client
where "ClientID" = 4
RETURNING "ClientID", "RegDate", "ClientID" ;
```

	ClientID	RegDate	ClientID
1	4	2022-12-14	4

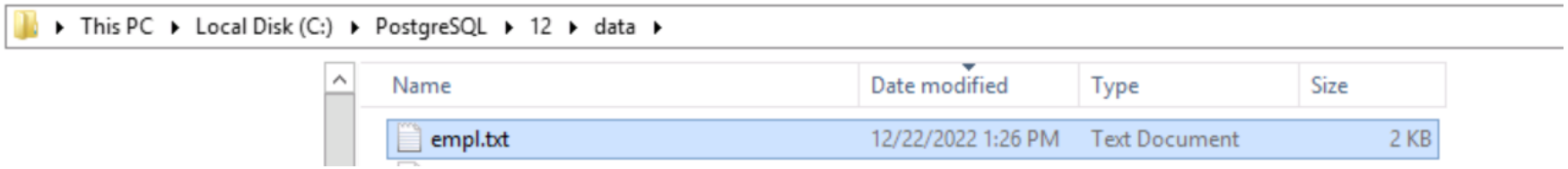
Вставка данных из внешних файлов

- PostgreSQL предоставляет возможность вставки данных в таблицы БД из внешних файлов с помощью команды **COPY FROM**
- Файлы, используемые для ввода с помощью COPY, должны быть:
  - либо в стандартном текстовом формате ASCII, поля которого разделены универсальным символом
  - либо в формате двоичной таблицы PostgreSQL
- Команда **COPY FROM** работает намного быстрее, чем обычная команда **INSERT**:
  - данные считываются как одна транзакция непосредственно в целевую таблицу
  - вся процедура COPY завершится ошибкой, если хотя бы одна строка не будет соответствовать требованиям (типы данных, ограничения)

COPY FROM

```
COPY имя_таблицы [ ( имя_столбца [, ...] ) ]
    FROM { 'имя_файла' | PROGRAM 'команда' | STDIN }
    [ [ WITH ] ( параметр [, ...] ) ]
```

```
COPY "HR"."Employees" TO 'C:\\PostgreSQL\\12\\Data\\empl.txt';
```




```
CREATE TABLE employees_new (LIKE  "HR"."Employees");
COPY public.employees_new FROM 'C:\\PostgreSQL\\12\\Data\\empl.txt';
SELECT * FROM public.employees_new;
```

	empid	lastname	firstname	title	titleofcourtesy	birthdate	hiredate
1	2	Funk	Don	Vice President; Sales	Dr.	1962-02-19	2002-08-14
2	3	Lew	Judy	Sales Manager	Ms.	1973-08-30	2002-04-01

MERGE (начиная с 15 версии)

- Оператор MERGE позволяет добавить, изменить или удалить строки целевой таблицы на основе сравнения с источником записей
  - Действия оператора MERGE имеют тот же эффект, что и обычные одноимённые команды UPDATE, INSERT или DELETE
  - Синтаксис этих команд в MERGE отличается отсутствием предложения WHERE и имени таблицы.
  - Действия этих команд выполняются с целевой_таблицей
  - При указании DO NOTHING исходная строка пропускается

```
MERGE INTO имя_целевой_таблицы [ [ AS ] целевой_псевдоним ]
USING источник_данных ON условие_соединения
{ WHEN MATCHED [ AND условие ]
    THEN { изменение_при_объединении | удаление_при_объединении | DO NOTHING }
  WHEN NOT MATCHED [ AND условие ]
    THEN { добавление_при_объединении | DO NOTHING } }
```

https://postgrespro.ru/docs/postgresql/15/sql-merge

- https://www.tutorialsteacher.com/postgresql/