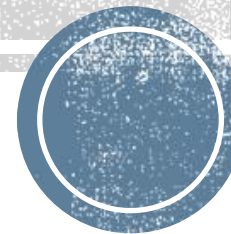


Практические занятия №№5-6

Знакомство с СУБД PostgreSQL: логическая и физическая архитектура.



Индексы

Индексы - специальные объекты базы данных, предназначенные для ускорения доступа к данным.

В PostgreSQL существует 6 типа индексов: B-tree, Hash, GiST (Generalized Search Tree), SP-GiST (space partitioning GiST - деревья квадрантов, k-мерные деревья, префиксные деревья), GIN (Generalized Inverted Index) и BRIN (Block Range Index - для очень больших таблиц).

Каждый тип индекса имеет свой алгоритм реализации, что позволяет существенно увеличить быстродействие, если для определенного вида данных выбирать определенный тип индекса.

PostgreSQL позволяет также создавать индексы с использованием выражений и частичные (partial) индексы (с использованием служебного слова WHERE).



Роли и привилегии

PostgreSQL управляет привилегиями в БД, используя концепцию ролей.

Ролью может быть отдельный пользователь БД, группа пользователей.

Роли могут быть владельцами объектов в БД (например таблиц), а также могут назначать привилегии доступа к этим объектам для других ролей.

Возможно предоставить одной роли членство в другой роли и соответственно передать этой роли права той роли, членом которой она будет.

Концепция ролей заменила старую концепцию пользователей и групп, предоставив ту же функциональность.

Начиная с версии 9.0 в PostgreSQL поддерживаются права на схемы и права по умолчанию.



Правила

Система правил (более правильно говорить: система правил изменения запросов) позволяет изменять запрос согласно заданным правилам и потом передает измененный запрос планировщику запросов для планирования и выполнения.

Система правил является мощным инструментом и может быть использована во многих случаях, таких как хранимые процедуры и представления.



Хранимые процедуры и триггеры

Хранимые процедуры могут быть использованы в триггерах и могут возвращать любой из поддерживаемых типов данных, а также массивы и списки. Начиная с версии 9.0, вызывать хранимые процедуры можно с указанием именуемых параметров.

Триггеры определяются как функции, которые инициируются операциями манипулирования.

Триггеры могут быть назначены до или после операций INSERT, UPDATE или DELETE.

Если произошло событие, на которое был назначен триггер, то вызывается закрепленная за этим триггером процедура.

При написании функций для триггеров могут использоваться разные языки программирования. Триггеры ассоциируются с таблицами и выполняются в алфавитном порядке.

В версии 9.0 введены триггеры на столбцы и, кроме того, при объявлении триггера можно использовать ключевое слово WHEN, которое добавляет дополнительное условие для срабатывания триггера.



Функции

Функции являются блоками кода, выполняемыми на сервере, а не на стороне клиента БД. Функции могут писаться на одном из таких языков:

встроенный процедурный язык PL/pgSQL, во многом аналогичный языку PL/SQL, что используется в СУБД Oracle;

скриптовые языки ▫ PL/Lua, PL/LOLCODE, PL/Perl, PL/PHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl и PL/Scheme ;

классические языки C, C + +, Java (через модуль PL/Java);

статистический язык R (через модуль PL/R).



Схемы

Схемы являются как бы дополнительными областями видимости внутри БД.

Также схему можно сравнить и с дополнительным путем (название схемы должно указываться перед названием таблицы) и с каталогом, внутри которого можно разместить таблицы.

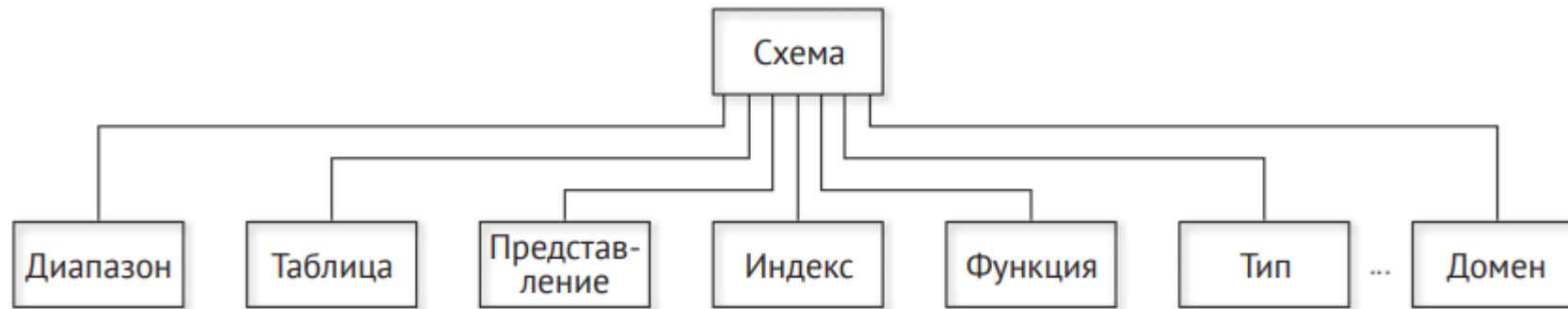
В любой БД по умолчанию существует схема public, в которой создаются все таблицы и которую не нужно указывать специально.

Администратор БД может создавать другие схемы (и разграничивать доступ к ним), что обеспечивает еще один уровень распределения прав доступа для пользователей, позволяет выделить каждому пользователю как бы персональный раздел внутри БД с теми же названиями таблиц, что и у других пользователей.



Схемы

Схема содержит все именованные объекты базы данных: таблицы, представления, функции, агрегаты, индексы, последовательности, триггеры, типы данных, домены и диапазоны.



Применение схем

- управление авторизацией;
- организация объектов базы данных;
- хранение стороннего SQL-кода.



Некоторые типы. Числовые типы

Имя	Размер	Описание	Диапазон
<code>smallint</code>	2 байта	целое в небольшом диапазоне	-32768 .. +32767
<code>integer</code>	4 байта	типичный выбор для целых чисел	-2147483648 .. +2147483647
<code>bigint</code>	8 байт	целое в большом диапазоне	-9223372036854775808 .. 9223372036854775807
<code>decimal</code>	переменный	вещественное число с указанной точностью	до 131072 цифр до десятичной точки и до 16383 — после
<code>numeric</code>	переменный	вещественное число с указанной точностью	до 131072 цифр до десятичной точки и до 16383 — после
<code>real</code>	4 байта	вещественное число с переменной точностью	точность в пределах 6 десятичных цифр
<code>double precision</code>	8 байт	вещественное число с переменной точностью	точность в пределах 15 десятичных цифр
<code>smallserial</code>	2 байта	небольшое целое с автоувеличением	1 .. 32767
<code>serial</code>	4 байта	целое с автоувеличением	1 .. 2147483647
<code>bigserial</code>	8 байт	большое целое с автоувеличением	1 .. 9223372036854775807

<https://postgrespro.ru/docs/postgresql/9.6/datatype>



Символьные типы

Имя типа	Примечания	Концевые пробелы	Максимальная длина
char	Эквивалентно char(1)	Семантически не значимы	1
name	Эквивалентно varchar(64). Используется PostgreSQL для имен объектов	Семантически значимы	64
char(n)	Псевдоним: character(n). Последовательность символов фиксированной длины. Внутреннее название – bpchar (blank padded character)	Семантически не значимы	10485760
varchar(n)	Псевдоним: character varying(n). Последовательность символов переменной длины, но не более n	Семантически значимы	10485760
text	Последовательность символов переменной длины	Семантически значимы	Не ограничена



Дата и время

Имя типа	Размер в байтах	Описание	Нижняя граница	Верхняя граница
timestamp without timezone	8	Дата и время без часового пояса, эквивалентно timestamp	4713 до н. э.	294276 н. э.
timestamp with timezone	8	Дата и время с часовым поясом, эквивалентно timestamptz	4713 до н. э.	294276 н. э.
date	4	Только дата	4713 до н. э.	294276 н. э.
time without timezone	8	Время суток	00:00:00	24:00:00
time with timezone	12	Время суток с часовым поясом	00:00:00+1459	24:00:00-1459
interval	16	Временной интервал	-178 000 000 лет	+178 000 000 лет



Денежный тип

Имя	Размер	Описание	Диапазон
money	8 байт	денежная сумма	-92233720368547758.08 .. +92233720368547758.07



Двоичный и логический типы

Имя	Размер	Описание
<code>bytea</code>	1 или 4 байта плюс сама двоичная строка	двоичная строка переменной длины

Имя	Размер	Описание
<code>boolean</code>	1 байт	состояние: истина или ложь



Геометрические типы

Имя	Размер	Описание	Представление
point	16 байт	Точка на плоскости	(x,y)
line	32 байта	Бесконечная прямая	$\{A,B,C\}$
lseg	32 байта	Отрезок	$((x_1,y_1),(x_2,y_2))$
box	32 байта	Прямоугольник	$((x_1,y_1),(x_2,y_2))$
path	16+16n байт	Закрытый путь (подобный многоугольнику)	$((x_1,y_1),\dots)$
path	16+16n байт	Открытый путь	$[(x_1,y_1),\dots]$
polygon	40+16n байт	Многоугольник (подобный закрытому пути)	$((x_1,y_1),\dots)$
circle	24 байта	Окружность	$\langle(x,y),r\rangle$ (центр окружности и радиус)



Типы, описывающие сетевые адреса

Имя	Размер	Описание
<code>cidr</code>	7 или 19 байт	Сети IPv4 и IPv6
<code>inet</code>	7 или 19 байт	Узлы и сети IPv4 и IPv6
<code>macaddr</code>	6 байт	MAC-адреса



Введение в язык описания данных (DDL)

Подмножество языка SQL, предназначенное для описания данных (SQL DDL), включает операторы CREATE, ALTER и DROP, а также GRANT и REVOKE, используемые для управления разграничением доступа.

За ключевым словом, определяющим оператор, следует другое ключевое слово, определяющее тип объекта, к которому применяется оператор, — например

CREATE TABLE является оператором создания таблиц.

Далее обычно следует имя объекта базы данных и затем предложения, уточняющие, что именно делается при выполнении оператора.



Создание таблицы

Выполняется с помощью оператора CREATE TABLE. Оператор создает пустую таблицу.

Значения в таблицу вводятся с помощью команды INSERT INTO. Оператор CREATE TABLE определяет имя таблицы и множество поименованных столбцов в указанном порядке. Для каждого столбца определен тип. Синтаксис команды следующий:

```
CREATE TABLE <таблица>  
(<столбец1> <тип данных1>[,...n])
```

```
CREATE TABLE customers  
(  
    Id SERIAL PRIMARY KEY,  
    FirstName CHARACTER VARYING(30),  
    LastName CHARACTER VARYING(30),  
    Email CHARACTER VARYING(30),  
    Age INTEGER  
);
```



Типы таблиц

- постоянная - возникает в момент создания и пропадает после удаления;
- временная - существует в течение сеанса. Часто такие таблицы используют в процедурных языках для моделирования бизнес-логики;
- нежурналируемая - операции с нежурналируемыми таблицами выполняются гораздо быстрее, чем с постоянными. Но они неустойчивы к сбоям.
- дочерняя - такая таблица наследует одну или несколько таблиц.



Ограничения

При создании или изменении структуры таблицы могут быть определены ограничения на вводимые значения. В этом случае будет отвергаться любое значение, которое не соответствует заданным ограничениям.

Можно задать ограничение, действующее на один столбец или на группу столбцов.

Ограничение на один столбец добавляется после определения типа данных столбца и действует только на значения этого столбца.

Ограничение на группу столбцов называется ограничением на таблицу.

Оно размещается после определения последнего столбца и действует на столбцы, которые указываются в скобках после этого ограничения.



Типы ограничений

1. Ограничение NOT NULL запрещает использование в указанном столбце NULL-значений. Такое ограничение может быть указано только как ограничение на столбец.

```
CREATE TABLE Customers
```

```
(
```

```
  Id SERIAL PRIMARY KEY,
```

```
  FirstName CHARACTER VARYING(20) NOT NULL,
```

```
  LastName CHARACTER VARYING(20) NOT NULL,
```

```
  Age INTEGER
```

```
);
```



Типы ограничений

2. Ограничение UNIQUE. Ограничение разрешает использовать только уникальные значения для указанных столбцов.

```
CREATE TABLE Customers
```

```
(
```

```
  Id SERIAL PRIMARY KEY,
```

```
  FirstName CHARACTER VARYING(20),
```

```
  LastName CHARACTER VARYING(20),
```

```
  Email CHARACTER VARYING(30) UNIQUE,
```

```
  Phone CHARACTER VARYING(30) UNIQUE,
```

```
  Age INTEGER
```

```
);
```

В данном случае столбцы, которые представляют электронный адрес и телефон, будут иметь уникальные значения. И мы не сможем добавить в таблицу две строки, у которых значения для этих столбцов будут совпадать.



Типы ограничений

Также мы можем определить этот атрибут на уровне таблицы:

```
CREATE TABLE Customers
```

```
(
```

```
  Id SERIAL PRIMARY KEY,
```

```
  FirstName CHARACTER VARYING(20),
```

```
  LastName CHARACTER VARYING(20),
```

```
  Email CHARACTER VARYING(30),
```

```
  Phone CHARACTER VARYING(30),
```

```
  Age INTEGER,
```

```
  UNIQUE(Email),
```

```
  UNIQUE(Phone)
```

```
);
```



Типы ограничений

3. Ограничение первичных ключей PRIMARY KEY.

Простой первичный ключ задается как ограничение на конкретный столбец.

Пример.

```
CREATE TABLE Customers
```

```
(  
    Id SERIAL PRIMARY KEY,  
    FirstName CHARACTER VARYING(30),  
    LastName CHARACTER VARYING(30),  
    Email CHARACTER VARYING(30),  
    Age INTEGER  
)
```

Первичный ключ уникально идентифицирует строку в таблице. В качестве первичного ключа необязательно должны выступать столбцы с типом SERIAL, они могут представлять любой другой тип.



Типы ограничений

Установка первичного ключа на уровне таблицы:

```
CREATE TABLE Customers
```

```
(
```

```
  Id SERIAL,
```

```
  FirstName CHARACTER VARYING(30),
```

```
  LastName CHARACTER VARYING(30),
```

```
  Email CHARACTER VARYING(30),
```

```
  Age INTEGER,
```

```
  PRIMARY KEY(Id)
```

```
);
```



Типы ограничений

Составные первичные ключи описываются как ограничения на таблицу, то есть после определения всех столбцов.

```
CREATE TABLE OrderLines  
(  
    OrderId INTEGER,  
    ProductId INTEGER,  
    Quantity INTEGER,  
    Price MONEY,  
    PRIMARY KEY(OrderId, ProductId)  
);
```



Типы ограничений

4. Ограничение на проверку CHECK. Задает множество возможных значений атрибута. Оно записывается как ограничение на столбец или таблицу. Для одного столбца может быть задано несколько ограничений.

Пример.

```
CREATE TABLE Customers
```

```
(
```

```
  Id SERIAL PRIMARY KEY,
```

```
  FirstName CHARACTER VARYING(20),
```

```
  LastName CHARACTER VARYING(20),
```

```
  Age INTEGER DEFAULT 18 CHECK(Age >0 AND Age < 100),
```

```
  Email CHARACTER VARYING(30) UNIQUE CHECK(Email !="),
```

```
  Phone CHARACTER VARYING(20) UNIQUE CHECK(Phone !=")
```

```
);
```



Типы ограничений

5. Ограничение DEFAULT. Позволяет задать значение по умолчанию. Ограничение задается только как ограничение на столбец.

Пример.

```
CREATE TABLE Customers  
(  
    Id SERIAL PRIMARY KEY,  
    FirstName CHARACTER VARYING(20),  
    LastName CHARACTER VARYING(20),  
    Age INTEGER DEFAULT 18  
);
```



Типы ограничений

6. Ограничения-исключения EXCLUDE

Ограничения-исключения гарантируют, что при сравнении любых двух строк по указанным столбцам или выражениям с помощью заданных операторов, минимум одно из этих сравнений возвратит false или NULL. Записывается это так:

```
CREATE TABLE circles (  
    c circle,  
    EXCLUDE USING gist (c WITH &&)  
);
```



Типы ограничений

7. Ограничение внешнего ключа FOREIGN KEY.

Для связи между таблицами применяются внешние ключи. Внешний ключ устанавливается для столбца из зависимой, подчиненной таблицы (referencing table), и указывает на один из столбцов из главной таблицы (referenced table). Как правило, внешний ключ указывает на первичный ключ из связанной главной таблицы.

Общий синтаксис установки внешнего ключа на уровне столбца:

REFERENCES главная_таблица (столбец_главной_таблицы)

[ON DELETE {CASCADE|RESTRICT}]

[ON UPDATE {CASCADE|RESTRICT}]



Типы ограничений

Чтобы установить связь между таблицами, после ключевого слова REFERENCES указывается имя связанной таблицы и далее в скобках имя столбца из этой таблицы, на который будет указывать внешний ключ.

После выражения REFERENCES может идти выражение ON DELETE и ON UPDATE, которые уточняют поведение при удалении или обновлении данных.

Общий синтаксис установки внешнего ключа на уровне таблицы:

FOREIGN KEY (столбец1, столбец2, ... столбецN)

REFERENCES главная_таблица (столбец_главной_таблицы1, столбец_главной_таблицы2, ... столбец_главной_таблицыN)

[ON DELETE {CASCADE | RESTRICT}]

[ON UPDATE {CASCADE | RESTRICT}]



Типы ограничений

Например, определим две таблицы и свяжем их посредством внешнего ключа:

```
CREATE TABLE Customers
```

```
(
```

```
  Id SERIAL PRIMARY KEY,
```

```
  Age INTEGER,
```

```
  FirstName VARCHAR(20) NOT NULL
```

```
);
```

```
CREATE TABLE Orders
```

```
(
```

```
  Id SERIAL PRIMARY KEY,
```

```
  CustomerId INTEGER,
```

```
  Quantity INTEGER,
```

```
  FOREIGN KEY (CustomerId) REFERENCES Customers (Id)
```

```
);
```



Типы ограничений

При задании внешнего ключа можно определить стратегию поддержки ссылочной целостности отдельно для удаления и изменения записей в подчиненной таблице.

Стратегия для удаления указывается после слов ON DELETE, а стратегия для изменения - после слов ON UPDATE.

- **CASCADE:** автоматически удаляет или изменяет строки из зависимой таблицы при удалении или изменении связанных строк в главной таблице.
- **RESTRICT:** предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице. То есть фактически какие-либо действия отсутствуют.
- **NO ACTION:** действие по умолчанию, предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице. И генерирует ошибку. В отличие от RESTRICT выполняет отложенную проверку на связанность между таблицами.
- **SET NULL:** при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение NULL.
- **SET DEFAULT:** при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение по умолчанию, которое задается с помощью атрибуты DEFAULT. Если для столбца не задано значение по умолчанию, то в качестве него применяется значение NULL.



Типы ограничений

Оператор CONSTRAINT. Установка имени ограничений.

С помощью ключевого слова CONSTRAINT можно задать имя для ограничений. В качестве ограничений могут использоваться PRIMARY KEY, UNIQUE, CHECK.

Имена ограничений можно задать на уровне столбцов. Они указываются после CONSTRAINT перед атрибутами:

```
CREATE TABLE Customers
```

```
(
```

```
  Id SERIAL CONSTRAINT customer_Id PRIMARY KEY,
```

```
  Age INTEGER CONSTRAINT customers_age_check CHECK(Age >0 AND Age < 100),
```

```
  FirstName CHARACTER VARYING(20) NOT NULL,
```

```
  LastName CHARACTER VARYING(20) NOT NULL,
```

```
  Email CHARACTER VARYING(30) CONSTRAINT customers_email_key UNIQUE,
```

```
  Phone CHARACTER VARYING(20) CONSTRAINT customers_phone_key UNIQUE
```

```
);
```



При создании таблиц сверяйтесь со следующим минимальным контрольным списком.

- Что будет первичным ключом?
- Каковы значения по умолчанию у каждого столбца?
- Каков тип каждого столбца?
- Каковы ограничения на столбцы или группы столбцов?
- Правильно ли заданы разрешения для таблиц, последовательностей и схем?
- Заданы ли внешние ключи с соответствующими действиями?
- Каков жизненный цикл данных?
- Какие операции над данными разрешены?



Изменение структуры таблицы

Для модификации структуры и ограничений существующей таблицы используется оператор ALTER TABLE.

Одна инструкция ALTER TABLE может быть применена только к одному столбцу или ограничению.

Синтаксис инструкции ALTER TABLE имеет формы:

1. Добавление столбца к таблице.

```
ALTER TABLE <имя таблицы>
```

```
ADD <имя столбца> <тип>;
```

2. Добавление ограничения к таблице.

```
ALTER TABLE <имя таблицы>
```

```
ADD CONSTRAINT <имя ограничения> <ограничение>;
```



Изменение структуры таблицы

3. Изменение свойств столбца.

```
ALTER TABLE <имя таблицы>
```

```
ALTER COLUMN <имя столбца> <тип> [<ограничение>];
```

4. Удаление столбца из таблицы.

```
ALTER TABLE <имя таблицы>
```

```
DROP COLUMN <имя столбца>;
```

5. Удаление ограничения таблицы.

```
ALTER TABLE <имя таблицы>
```

```
DROP CONSTRAINT <имя ограничения>;
```



Последовательности

Типы данных `smallserial`, `serial` и `bigserial` не являются настоящими типами, а представляют собой просто удобное средство для создания столбцов с уникальными идентификаторами (подобное свойству `AUTO_INCREMENT` в некоторых СУБД). В текущей реализации запись:

```
CREATE TABLE имя_таблицы (  
    имя_столбца SERIAL  
);
```

равнозначна следующим командам:

```
CREATE SEQUENCE имя_таблицы_имя_столбца_seq;  
CREATE TABLE имя_таблицы (  
    имя_столбца integer NOT NULL DEFAULT nextval('имя_таблицы_имя_столбца_seq')  
);  
ALTER SEQUENCE имя_таблицы_имя_столбца_seq OWNED BY имя_таблицы.имя_столбца;
```



Удаление таблицы

Удаляет таблицу из БД. Удалять можно только пустые таблицы.

```
DROP TABLE <имя таблицы>
```

Например,

```
DROP TABLE PD
```

Команду DROP TABLE нельзя использовать для удаления таблицы, на которую ссылается ограничение FOREIGN KEY. Сначала следует удалить ссылающееся ограничение FOREIGN KEY или ссылающуюся таблицу.

Например, чтобы удалить таблицу D, надо выполнить команды:

```
DROP TABLE PD
```

```
DROP TABLE D
```



Графический клиент: pgAdmin

pgAdmin — популярное графическое средство для администрирования PostgreSQL. Программа упрощает основные задачи администрирования, отображает объекты баз данных, позволяет выполнять запросы SQL

Чтобы запустить pgAdmin 4 на Windows, воспользуйтесь установщиком на странице pgadmin.org/download.

Процесс установки прост и очевиден, все предлагаемые значения можно оставить без изменений.

Для Debian и Ubuntu подключите репозиторий PostgreSQL и выполните команду:

```
$ sudo apt-get install pgadmin4
```



Подключение к серверу

Нажмите на значок **Добавить новый сервер** (Add New Server) и в появившемся окне на вкладке **Общие** (General) введите произвольное имя (Name) для соединения.

На вкладке **Соединение** (Connection) введите имя сервера (Host name/address), порт (Port), имя пользователя (Username) и пароль (Password).

Если не хотите вводить пароль каждый раз вручную, отметьте флажок **Сохранить пароль** (Save password).

Например, для postgres это можно сделать следующей командой (в psql):

```
postgres=# ALTER ROLE postgres PASSWORD  
'p@ssw0rd';
```



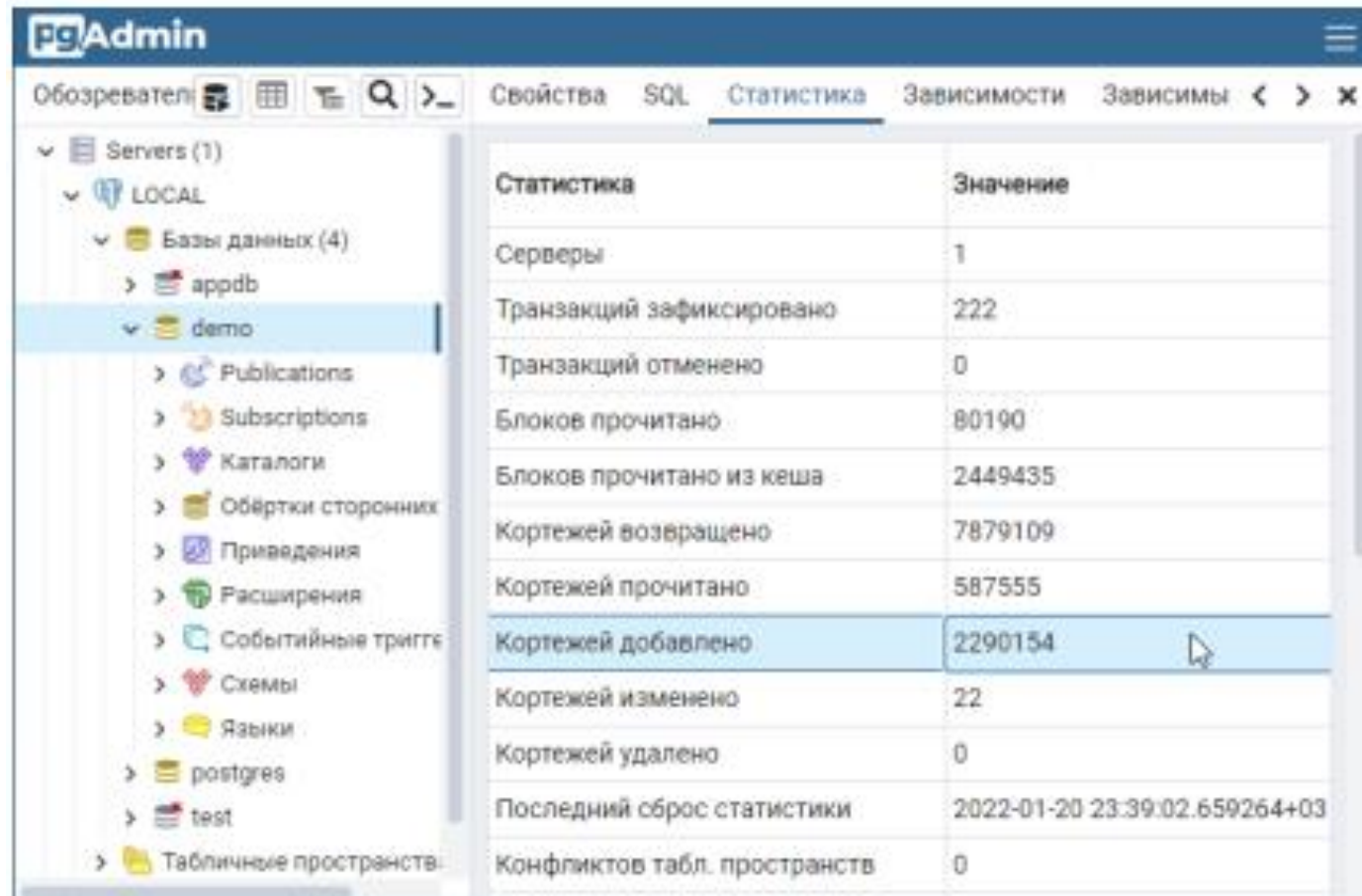
The screenshot shows a window titled "Создание Сервер" (Create Server) with a close button in the top right corner. The window has five tabs: "General", "Соединение" (Connection), "SSL", "SSH Tunnel", and "Дополнительно" (Advanced). The "Соединение" tab is currently selected. The form contains the following fields and controls:

- Имя/адрес сервера** (Host name/address): Text input field containing "localhost".
- Порт** (Port): Text input field containing "5432".
- Служебная база данных** (Service): Text input field containing "postgres".
- Имя пользователя** (Username): Text input field containing "postgres".
- Kerberos authentication?**: A toggle switch that is currently turned off.
- Пароль** (Password): Password input field with masked characters "*****".
- Сохранить пароль?** (Save password?): A toggle switch that is currently turned on.
- Роль** (Role): Text input field.
- Service**: Text input field.

At the bottom of the window, there are three buttons: an information icon (i), a question mark icon (?), and a "Закреть" (Close) button. To the right of these are two more buttons: "Сбросить" (Reset) and "Сохранить" (Save).



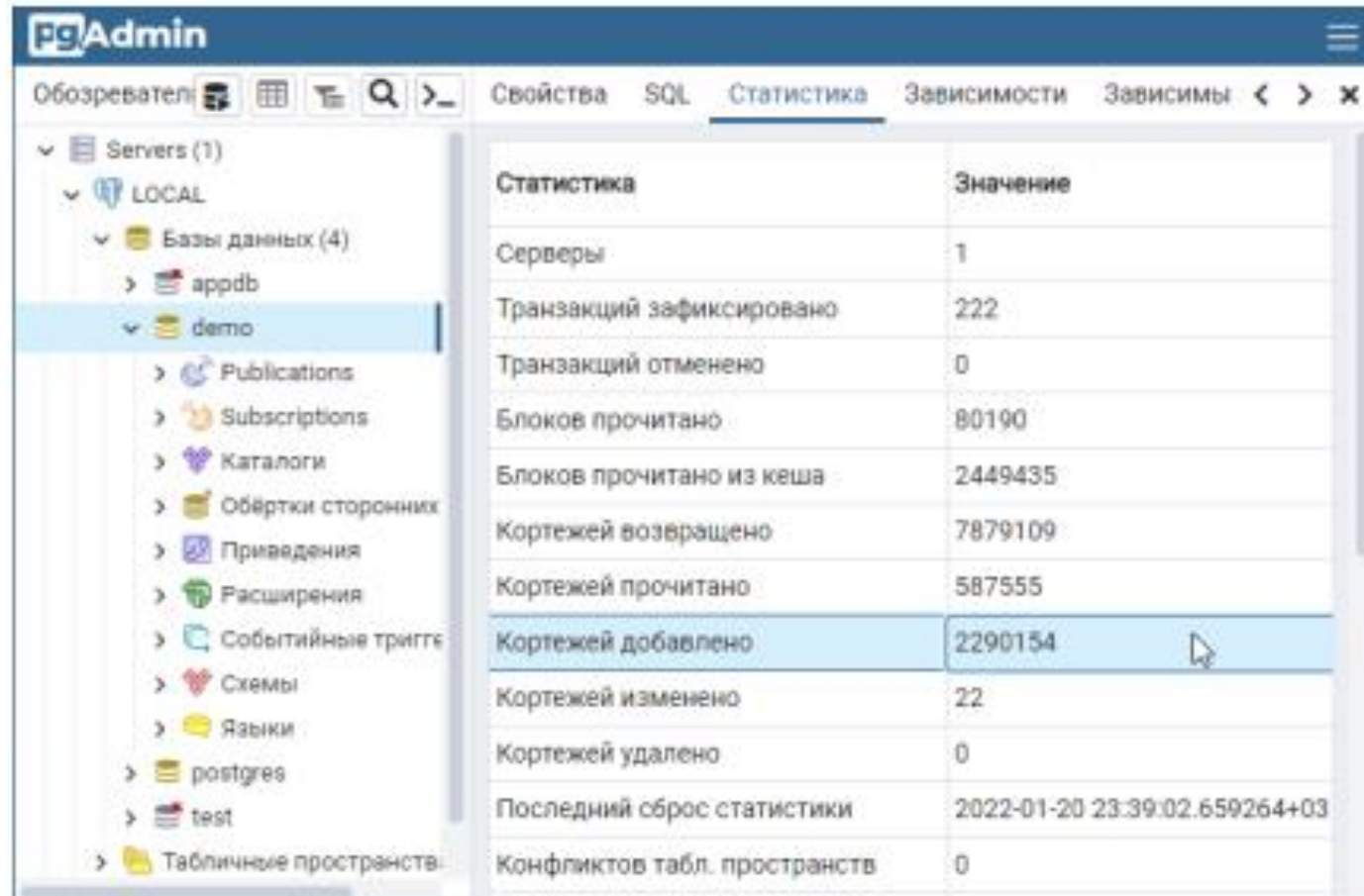
Навигатор



В левой части окна находится навигатор объектов. Разворачивая пункты списка, можно спуститься до сервера, в данном случае он называется LOCAL.



Навигатор



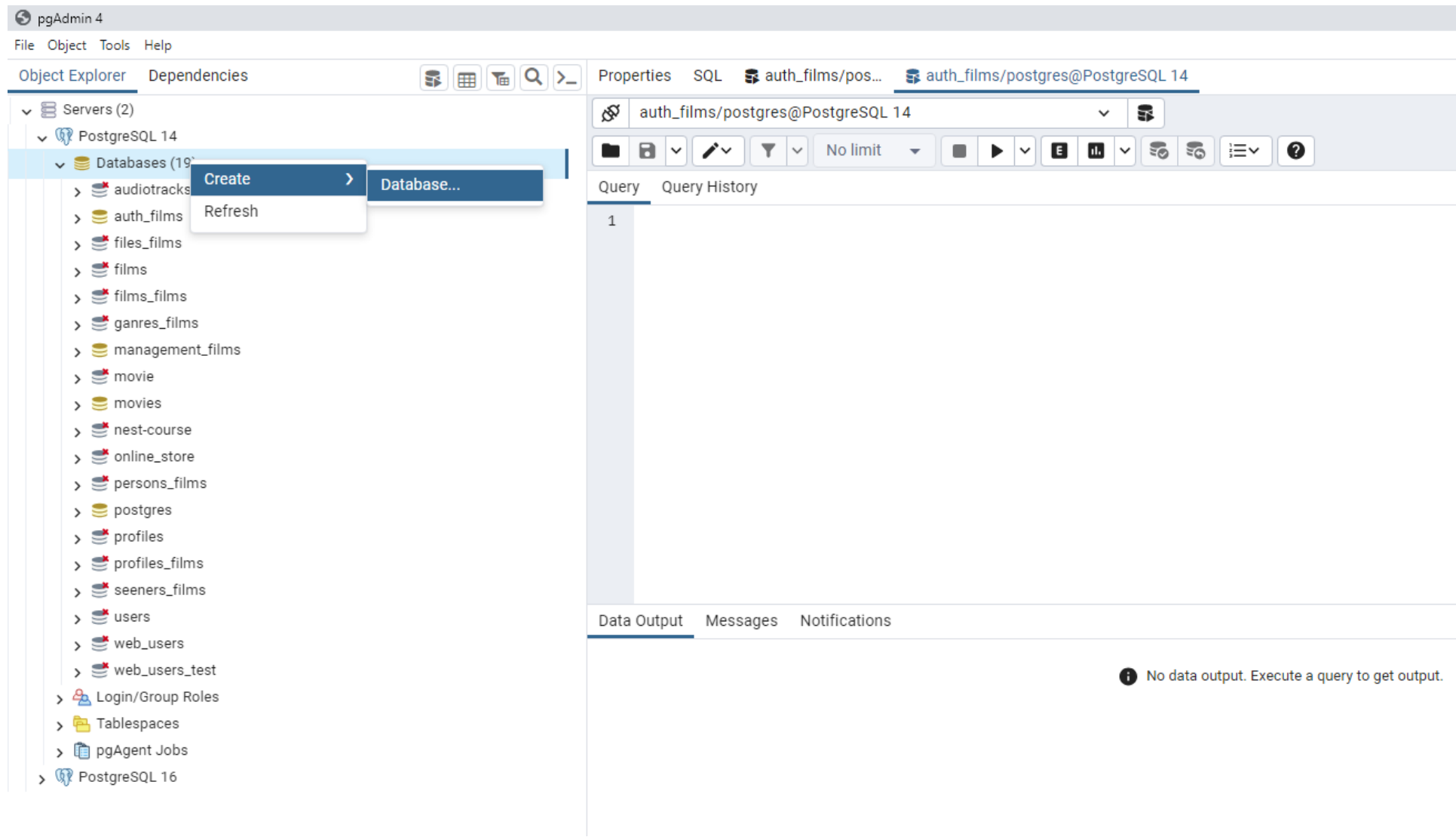
Развернув пункт Схемы (Schemas) для базы данных demo, можно обнаружить все таблицы, посмотреть их столбцы, ограничения целостности и др.

Для каждого типа объекта в контекстном меню (по правой кнопке мыши) приведен список действий, которые с ним можно совершить.

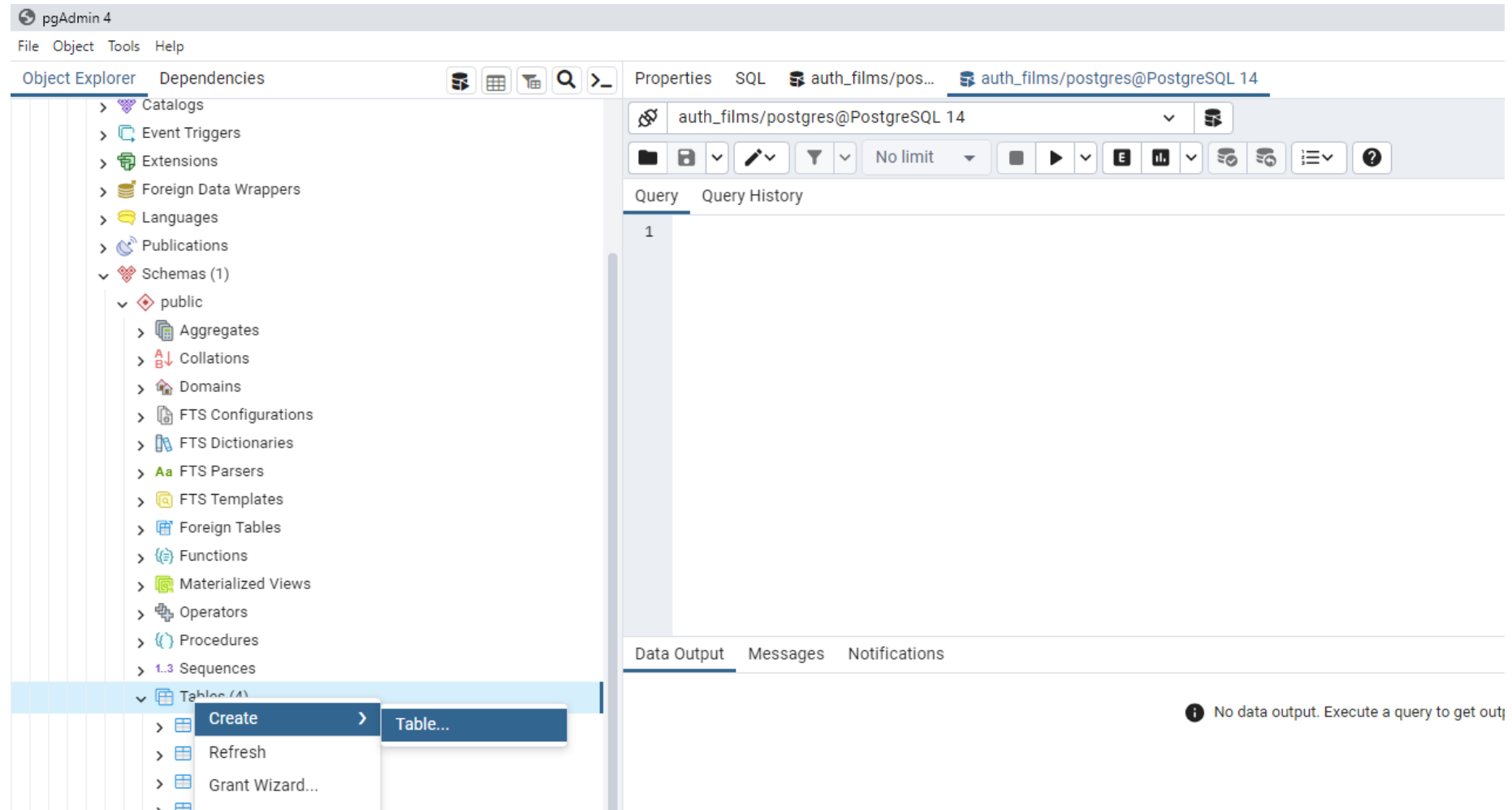
Например, выгрузить в файл или загрузить из файла, выдать привилегии, удалить.



Создание базы данных



Создание таблицы



Создание таблицы

Create - Table

General

Columns

Advanced

Constraints

Partitions

Parameters

Security

SQL

Inherited from table(s)

Select to inherit from...

Columns

+

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div></div>	<div>Select an item...</div>			<div></div>	<div></div>	<div></div>

Add row

!

 'Name' cannot be empty.

i

?

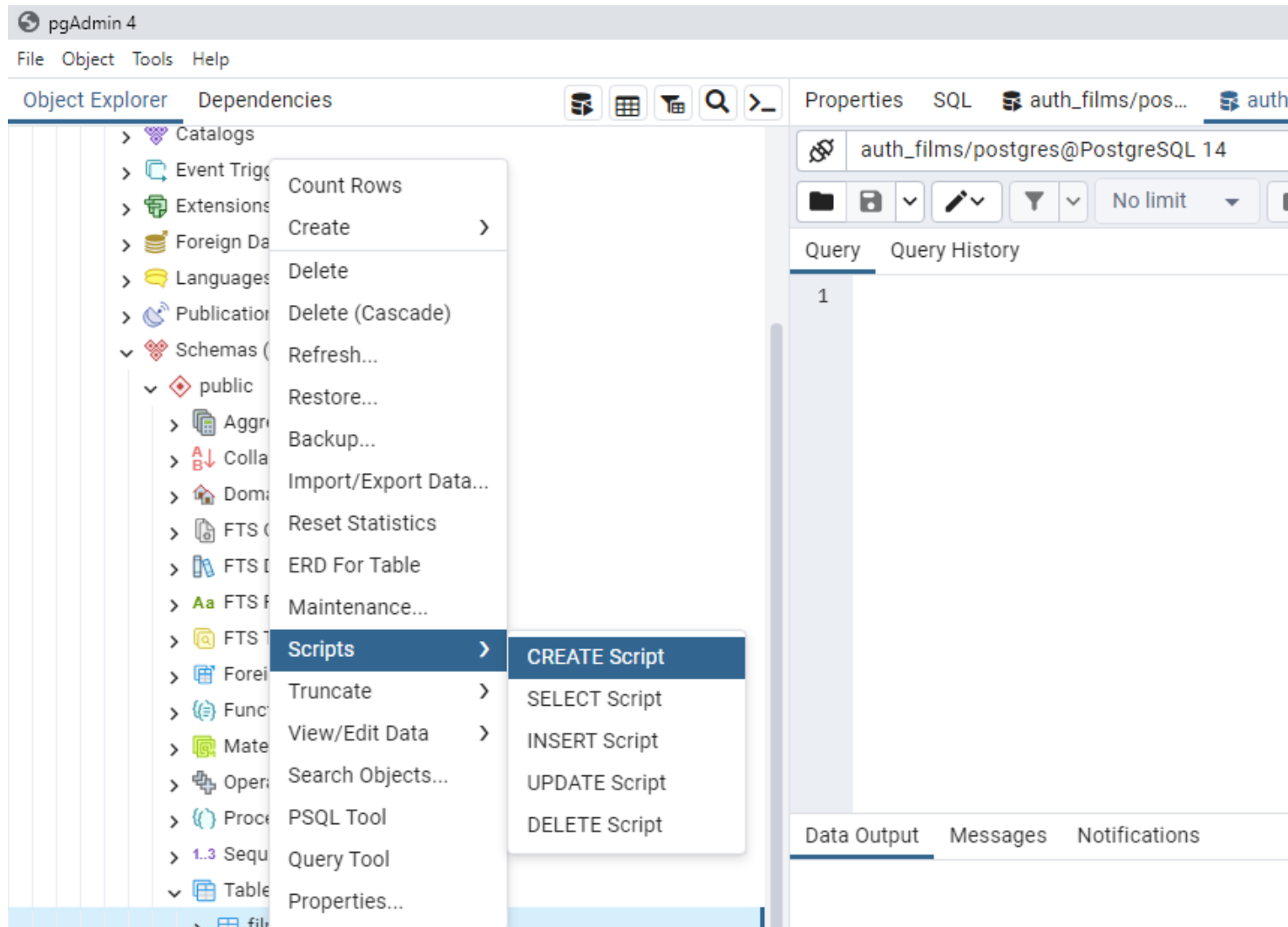
Close

Reset

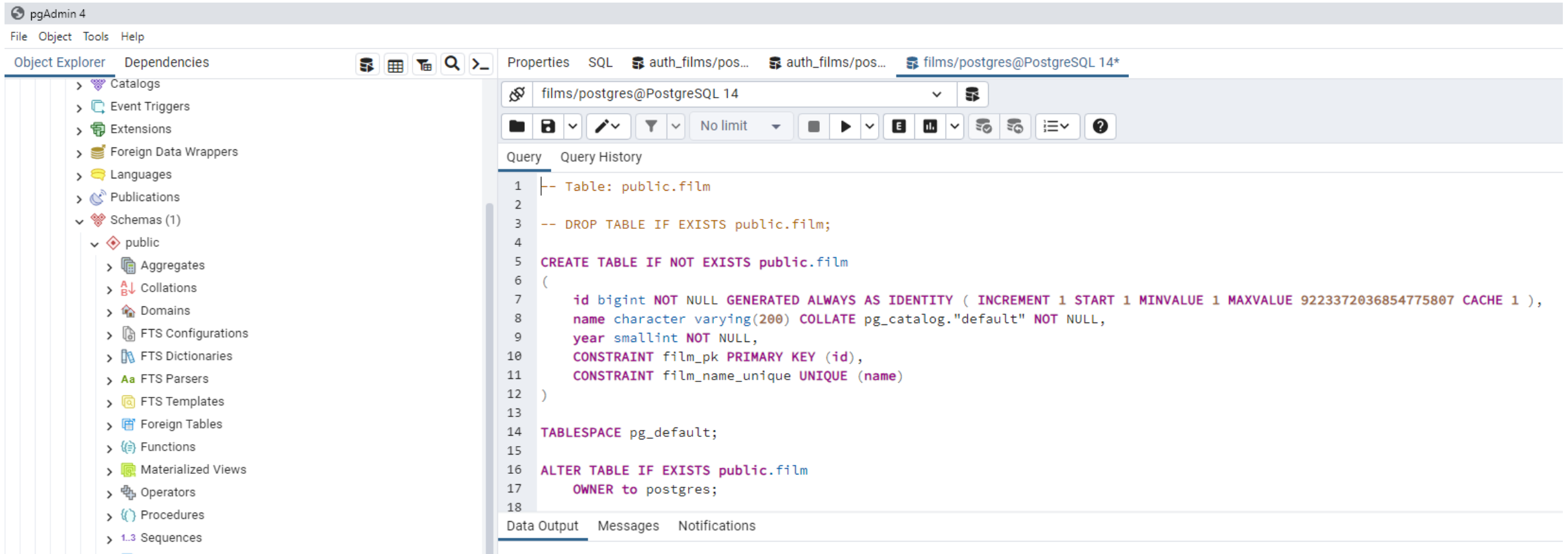
Save



Посмотреть скрипт для создания таблицы



Скрипт для создания таблицы



The screenshot displays the pgAdmin 4 web interface. On the left, the 'Object Explorer' pane shows the database structure, with the 'public' schema selected. The main area is divided into a toolbar and a query editor. The toolbar includes icons for connecting, saving, editing, and running queries. The query editor shows a SQL script for creating a table named 'film' in the 'public' schema. The script includes a comment, a DROP statement, a CREATE TABLE statement with a primary key and a unique constraint, and an ALTER TABLE statement to set the owner to 'postgres'.

```
1  -- Table: public.film
2
3  -- DROP TABLE IF EXISTS public.film;
4
5  CREATE TABLE IF NOT EXISTS public.film
6  (
7      id bigint NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 9223372036854775807 CACHE 1 ),
8      name character varying(200) COLLATE pg_catalog."default" NOT NULL,
9      year smallint NOT NULL,
10     CONSTRAINT film_pk PRIMARY KEY (id),
11     CONSTRAINT film_name_unique UNIQUE (name)
12 )
13
14 TABLESPACE pg_default;
15
16 ALTER TABLE IF EXISTS public.film
17     OWNER to postgres;
18
```



Задание ограничений

pgAdmin 4

File Object Tools Help

Object Explorer Dependencies

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (4)

film

film_genres

films

genres

Properties SQL auth_films/pos... auth_films/pos... films/postgres@PostgreSQL 14*

films/postgres@PostgreSQL 14







No limit

film

General Columns Advanced Constraints Parameters Security SQL

Inherited from table(s) Select to inherit from...

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
 	id	bigint			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
 	name	character varying	200		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 	year	smallint			<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Close Reset Save



Задание ограничений

The screenshot shows the pgAdmin 4 interface. On the left, the 'Object Explorer' pane displays the database structure, with the 'film' table selected under the 'public' schema. The main window shows the 'Constraints' tab for the 'film' table, listing several foreign key constraints. The constraints are as follows:

Name	Columns	Referenced Table
composer_person_fk	(composer_id) -> (id)	public.person
country_countries_fk	(country_id) -> (id)	public.countries
designer_person_fk	(designer_id) -> (id)	public.person
director_person_fk	(director_id) -> (id)	public.person
editor_person_fk	(editor_id) -> (id)	public.person
operator_person_fk	(operator_id) -> (id)	public.person
producer_person_fk	(producer_id) -> (id)	public.person
scenario_person_fk	(scenario_id) -> (id)	public.person

At the bottom of the window, there are buttons for 'Close', 'Reset', and 'Save'.



Пример: таблица_связка «student_discipl», создание внешних ключей

Create - Table

General Columns Advanced Constraints Partitions Parameters Security SQL

Inherited from table(s) Select to inherit from...

Columns +

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
⋮	✎	student_id_ref	serial			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
⋮	✎	discipl_id_ref	serial			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

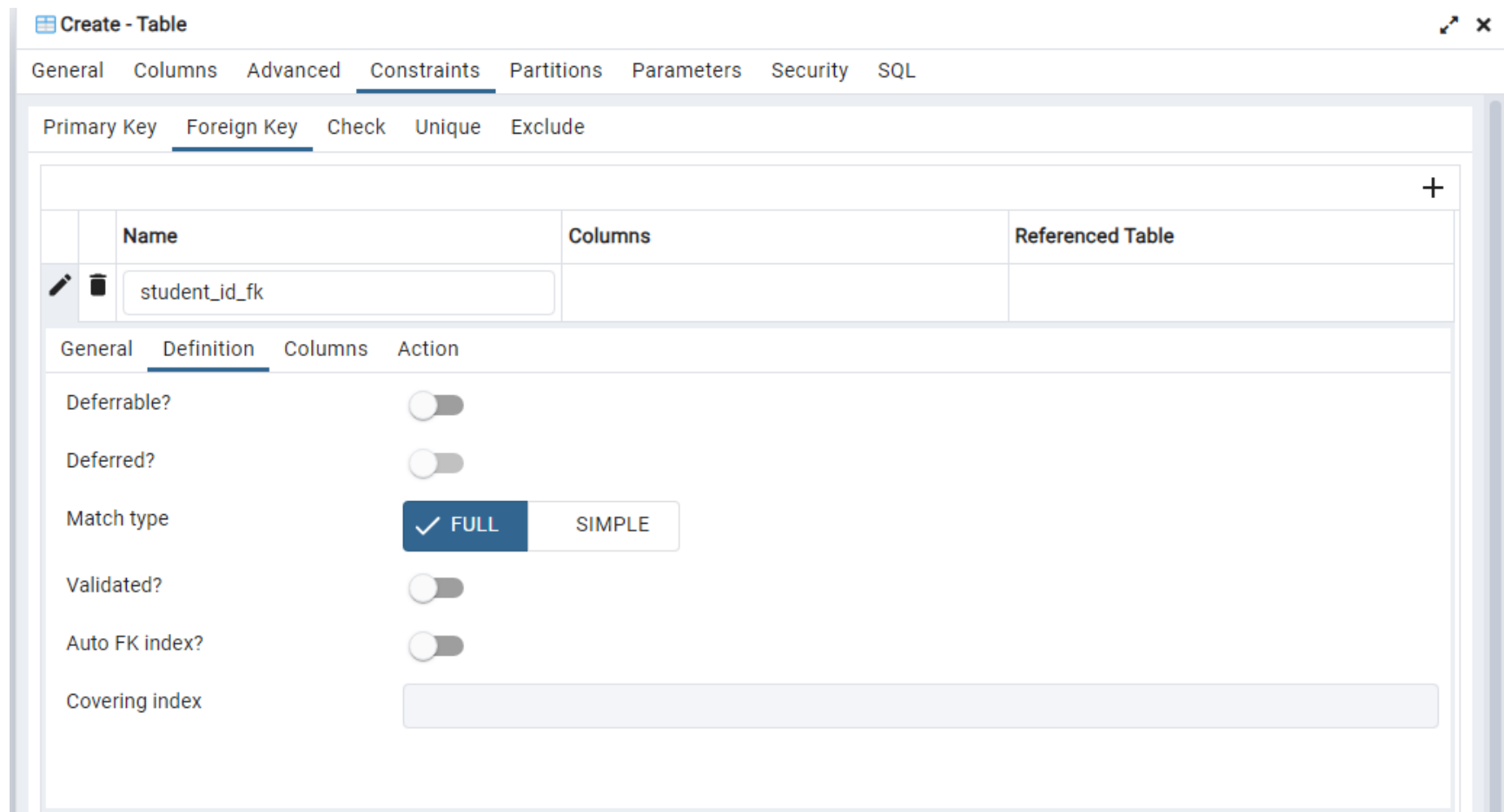
ⓘ ?

Close Reset Save

Добавлены поля student_id_ref и discipl_id_ref, которые будут внешними ключами, вместе они составляют первичный ключ



Создание внешних ключей



Create - Table

General Columns Advanced **Constraints** Partitions Parameters Security SQL

Primary Key **Foreign Key** Check Unique Exclude

Name	Columns	Referenced Table
student_id_fk		

General **Definition** Columns Action

Deferrable? ☐

Deferred? ☐

Match type **✓ FULL** SIMPLE

Validated? ☐

Auto FK index? ☐

Covering index

Добавляется ограничение с именем student_id_fk: нужно выбрать столбец данной таблицы (см. следующий слайд) и сделать его ссылающимся на student_id в таблице student.



Создание внешних ключей

Create - Table

General Columns Advanced **Constraints** Partitions Parameters Security SQL

Primary Key **Foreign Key** Check Unique Exclude

Name	Columns	Referenced Table
student_id_fk		

General Definition **Columns** Action

Columns

Local column	student_id_ref
References	public.student
Referencing	student_id

Add

Local	Referenced	Referenced Table
-------	------------	------------------

Выбраны:

- столбец в текущей таблице - student_id_ref,
- таблица, на которую надо сослаться - public.student,
- столбец в таблице, на который надо сослаться - student.id.

Нажимаем Add.





Создание внешних ключей

Create - Table

General Columns Advanced **Constraints** Partitions Parameters Security SQL

Primary Key **Foreign Key** Check Unique Exclude

	Name	Columns	Referenced Table
 	student_id_fk	(student_id_ref) -> (student_id)	public.student

General Definition **Columns** Action


Columns



Local column




References

Referencing

Add

	Local	Referenced	Referenced Table
	student_id_ref	student_id	public.student

 Close  Reset  Save

Один внешний ключ создан.

То же нужно повторить для другого столбца таблицы-связки: нажимаем + в правом верхнем углу.



Создание внешних ключей

Create - Table ↗ ✕

General Columns Advanced **Constraints** Partitions Parameters Security SQL

✕

student_id_ref

student_id

public.student

✎

✕

discipl_id_ref

General Definition **Columns** Action

Columns

Local column

discipl_id_ref

✕ | ▾

References

public.discipl

✕ | ▾

Referencing

discipl_id

✕ | ▾

Add

Local

Referenced

Referenced Table

Создано новое
ограничение
discipl_id_ref



Создание внешних ключей

Create - Table

General Columns Advanced **Constraints** Partitions Parameters Security SQL

Referencing Select an item...

Add

Local	Referenced	Referenced Table
student_id_ref	student_id	public.student

discipl_id_ref (discipl_id_ref) -> (discipl_id) public.discipl

General Definition **Columns** Action

Columns

Local column Select an item...

References public.discipl

Referencing Select an item...

Add

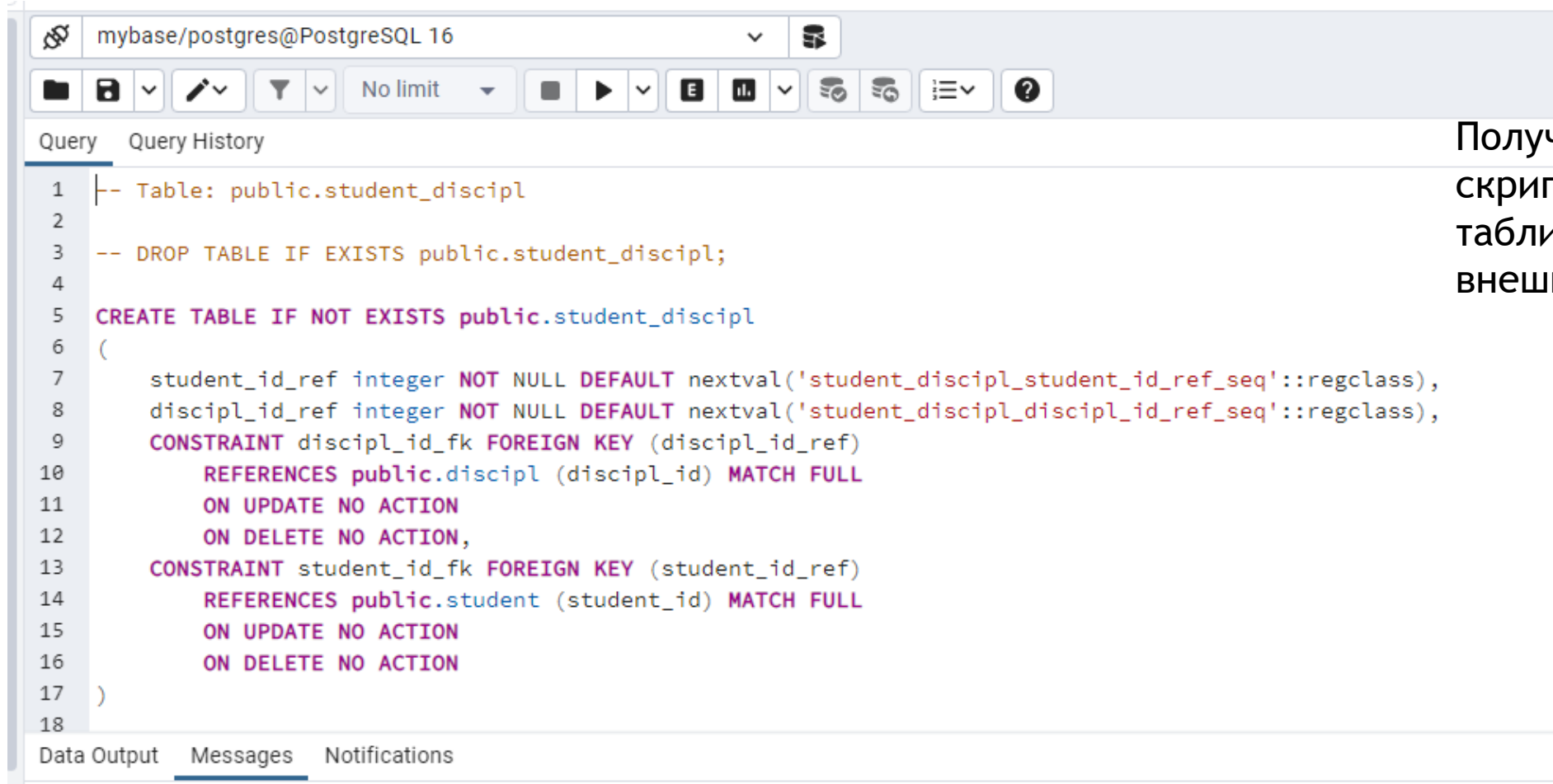
Local	Referenced	Referenced Table
discipl_id_ref	discipl_id	public.discipl

Close Reset Save

Далее нужно нажать на Save (если создаете таблицу, закрытие отменит ее создание!)



Создание внешних ключей



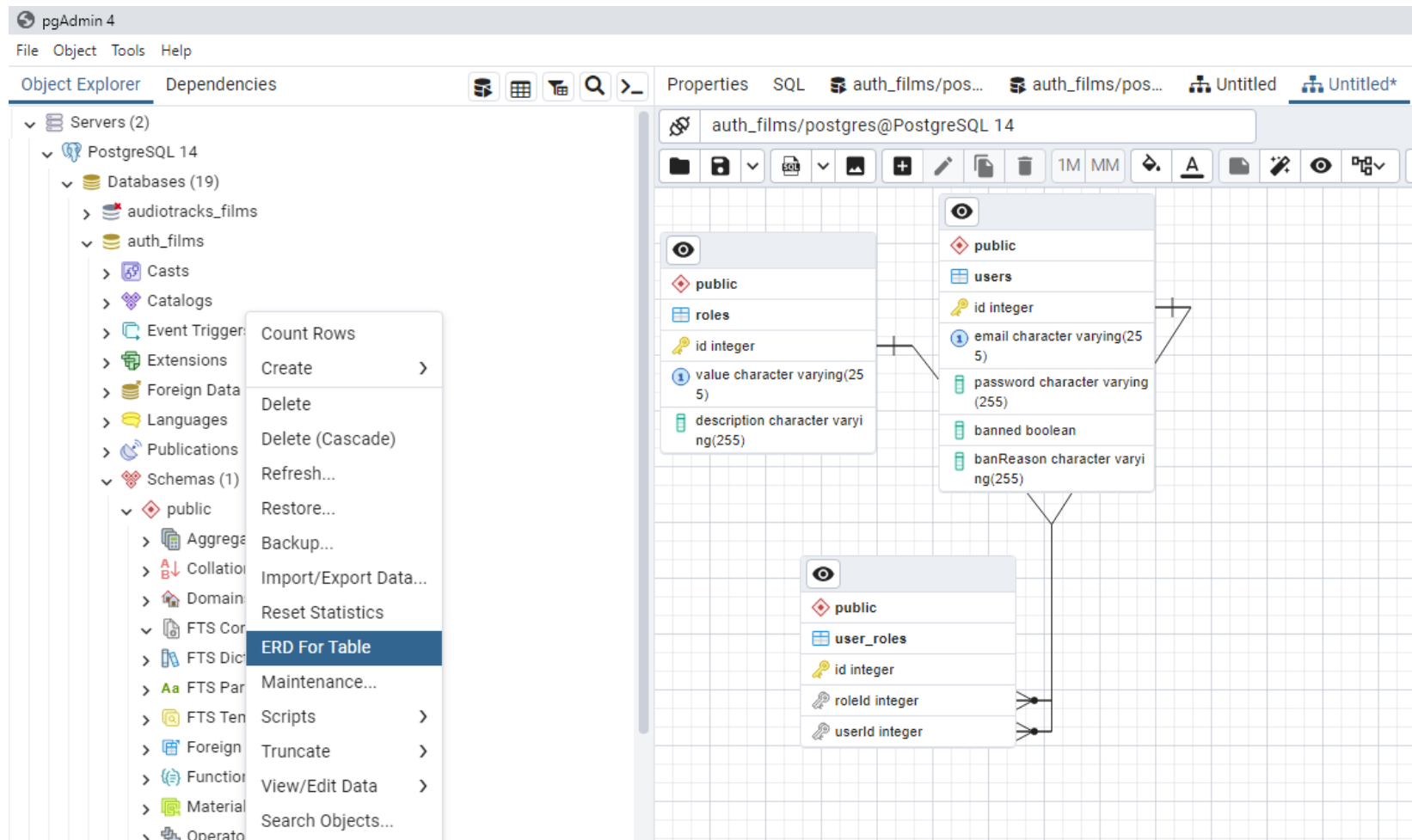
The screenshot shows a PostgreSQL client interface with a toolbar at the top containing icons for file operations, query execution, and settings. The main area displays a SQL script for creating a table named `public.student_discipl`. The script includes comments, a `DROP` statement, and a `CREATE TABLE` statement with two foreign key constraints. The bottom of the interface has tabs for 'Data Output', 'Messages', and 'Notifications'.

```
1  |-- Table: public.student_discipl
2
3  -- DROP TABLE IF EXISTS public.student_discipl;
4
5  CREATE TABLE IF NOT EXISTS public.student_discipl
6  (
7      student_id_ref integer NOT NULL DEFAULT nextval('student_discipl_student_id_ref_seq'::regclass),
8      discipl_id_ref integer NOT NULL DEFAULT nextval('student_discipl_discipl_id_ref_seq'::regclass),
9      CONSTRAINT discipl_id_fk FOREIGN KEY (discipl_id_ref)
10         REFERENCES public.discipl (discipl_id) MATCH FULL
11         ON UPDATE NO ACTION
12         ON DELETE NO ACTION,
13      CONSTRAINT student_id_fk FOREIGN KEY (student_id_ref)
14         REFERENCES public.student (student_id) MATCH FULL
15         ON UPDATE NO ACTION
16         ON DELETE NO ACTION
17  )
18
```

Получившийся скрипт для создания таблицы-связки с внешними ключами



Возможность автоматического построения ER-диаграммы



Упражнение 1.

1. Создать с помощью pgAdmin базу данных для заданной предметной области (выбрать из списка по варианту), содержащую
 - не менее 5 таблиц (задать названия и описания), каждая из которых содержит не менее 5 атрибутов;
 - все возможные виды ограничений (не менее 15), в том числе не менее 5 внешних ключей;
 - задать имена для некоторых ограничений;
 - настроить ограничения ссылочной целостности (использовать все возможные варианты ограничений).
2. Автоматически создать скрипты для создания структуры базы данных.



Упражнение 2.

Разработать запросы для изменения структуры БД, предварительно сохранив копию исходной БД:

- а. добавление/удаление/модификация столбцов и ограничений в существующие таблицы (модифицировать не менее 3 таблиц).
- б. создание дополнительной таблицы;
- в. добавление столбца в дополнительную таблицу;
- г. создание ограничения внешнего ключа для связи новой таблицы с существующими;
- д. модификация столбца в таблице;
- е. удаление столбца из таблицы;
- ж. удаления таблицы.

В результатах должны быть скрипты для создания исходных таблиц БД и всех упомянутых модификаций.



Варианты предметных областей

1. Библиотека.
2. Интернет-магазин для продажи тканей и фурнитуры.
3. Сеть склад-магазинов продовольственных товаров.
4. Авиарейсы.
5. Типография.
6. Пассажирские перевозки (автобусы).
7. Организация работы такси.
8. Продажа недвижимости.
9. Учет материальных предметов учреждения.
10. Сеть магазинов по продаже компьютерной техники и комплектующих.
11. Гостиничный комплекс.
12. Учет поступающих в вуз абитуриентов.
13. Систематизация журналов и газет.
14. Сеть магазинов по заказу и продаже фастфуда.
15. Школьный журнал.
16. Электронный журнал вуза.
17. Электронные обучающие курсы.
18. Вклады в банках города.
19. Туристические фирмы города.
20. Частная поликлиника.
21. Сеть магазинов бытовой техники.
22. Кадастровый учет недвижимости в городе.
23. Сайт для систематизации и просмотра фильмов.
24. Сайт для систематизации и прослушивания музыки.
25. Сеть магазинов по заказу и продаже кондитерских изделий.



Архитектура PostgreSQL

Существуют 3 основные редакции PostgreSQL:

1. Классический PostgreSQL
2. Российский Postgres Professional
3. 2nd Quadrant Postgres-XL.



Классический вариант



Базовая сборка от PGDG, PostgreSQL Global Development Group.

Свободно распространяемая СУБД PostgreSQL создана на основе некоммерческой СУБД Postgres, разработанной как open-source проект в Калифорнийском университете в Беркли.

Название расшифровывалось как «Post Ingres», и при создании Postgres были применены ранние наработки БД Ingres.



Postgres Professional



Сборка Postgres Pro -
русская коммерческая
СУБД, разработанная
компанией Postgres
Professional с
использованием СУБД
PostgreSQL.

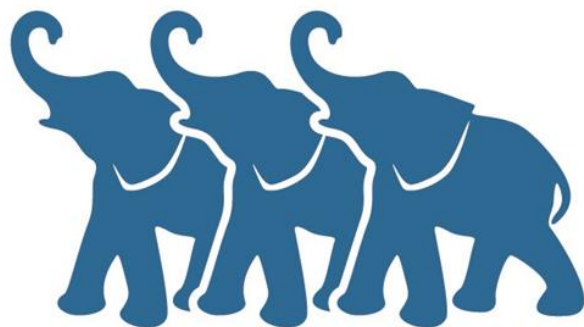
Существуют различные версии Postgres Pro, в том числе:

- Postgres Pro Standard,
 - Postgres Pro Enterprise (промышленная СУБД для высоконагруженных систем),
- защищенные варианты:
- Postgres Pro Certified,
 - Postgres Pro Enterprise сертифицированная версия.

От компании Postgres Professional имеется PostgreSQL (свободная open source лицензия) для Windows.



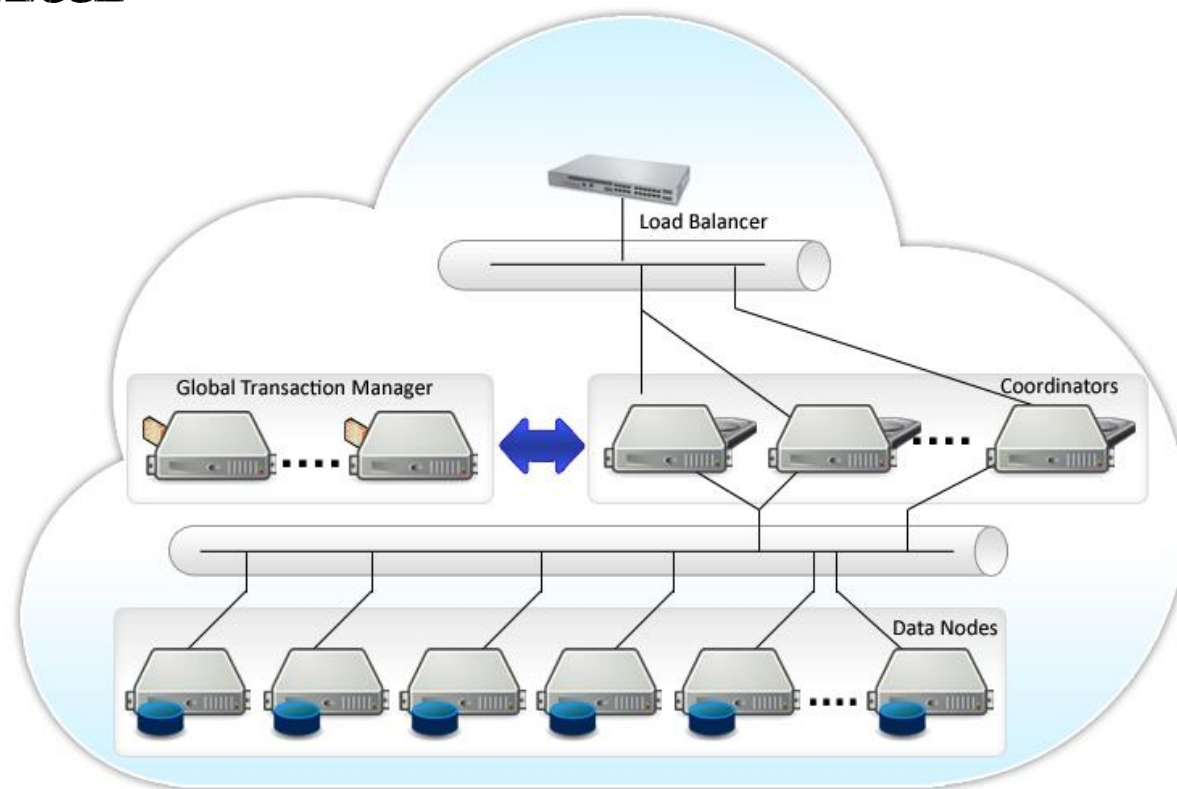
Postgres Professional



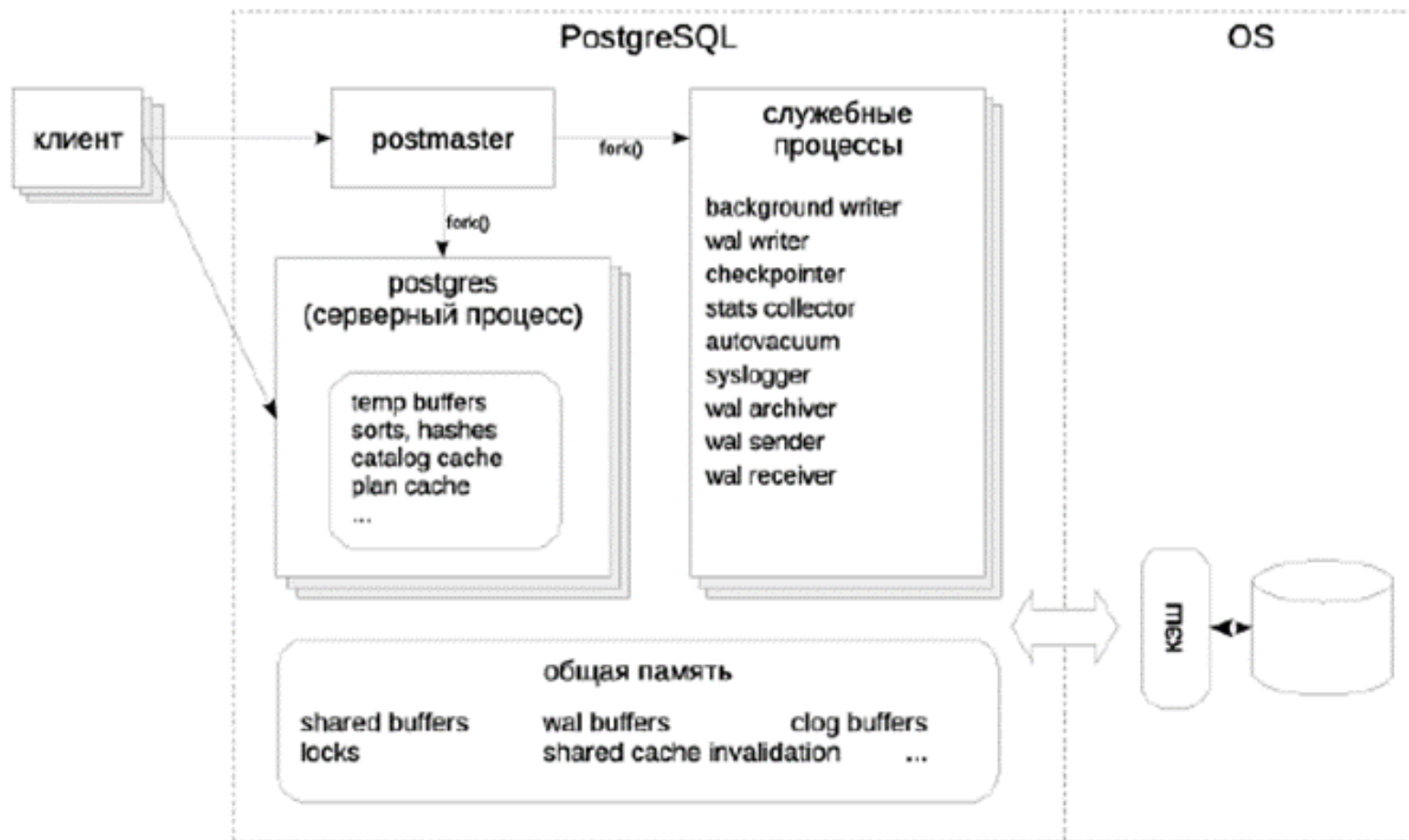
Postgres-XL

Название "Postgres-XL" означает "Расширяемая решетка".

Горизонтально масштабируемый кластер баз данных SQL с открытым исходным кодом, достаточно гибкий для обработки различных рабочих нагрузок базы данных.



Архитектура PostgreSQL



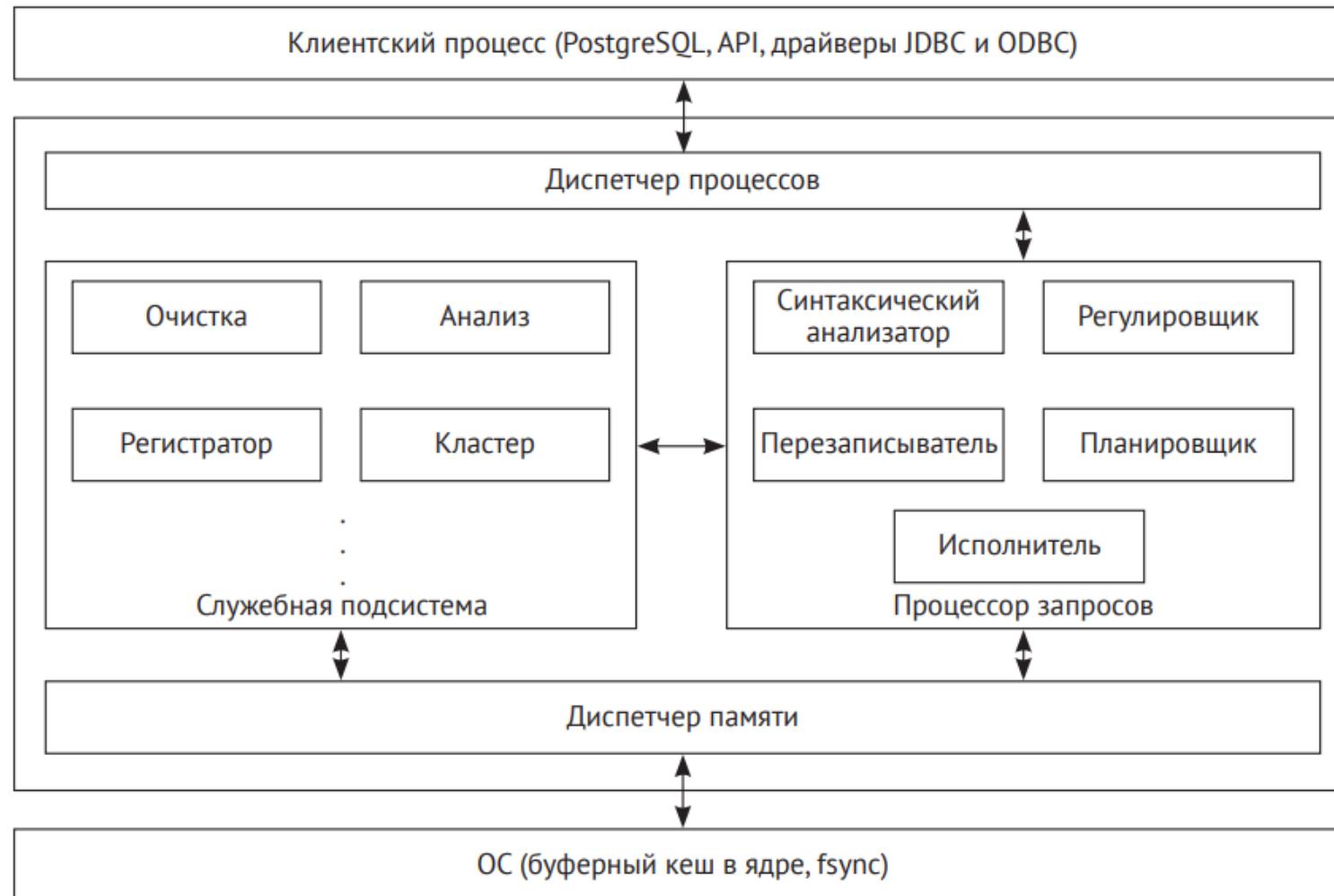
В основе работы СУБД PostgreSQL лежит серверный процесс базы данных, выполняемый на одном сервере.

Доступ из приложений к данным БД PostgreSQL производится с помощью специального процесса базы данных.

Клиентские программы не могут получить самостоятельный доступ к данным даже если они функционируют на том же ПК, на котором осуществляется серверный процесс.



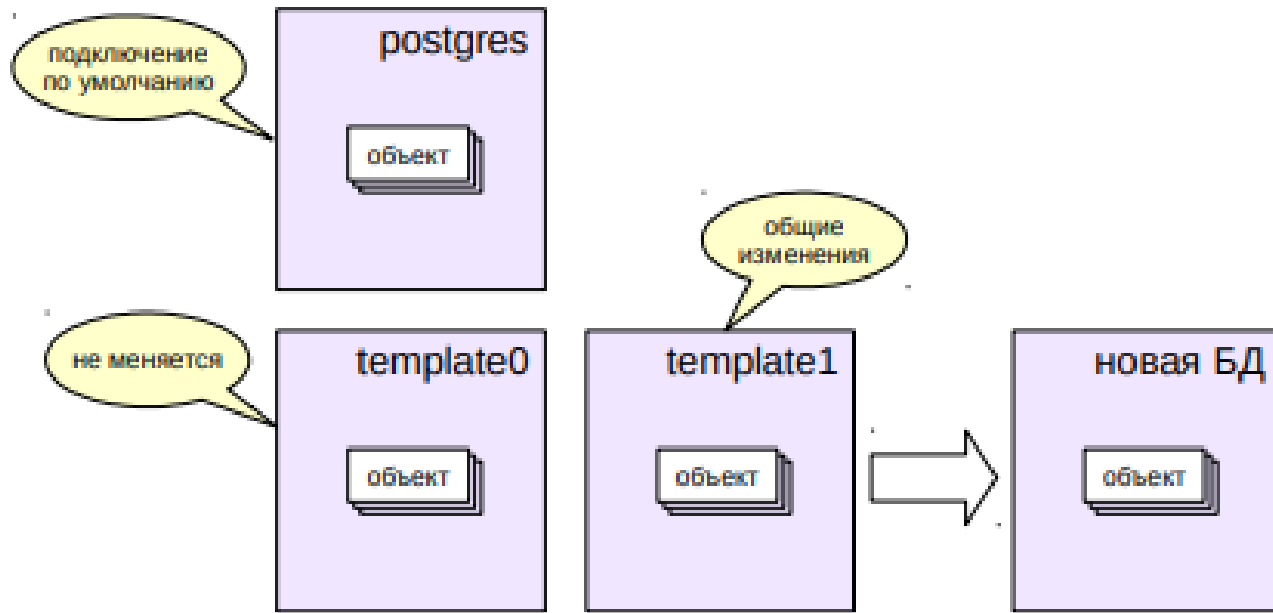
Архитектура PostgreSQL



Кластер баз данных

Инициализация кластера создает три базы данных

Новая база всегда клонируется из существующей



Экземпляр СУБД (экземпляр серверного процесса) управляет несколькими базами данных. Они называются **кластером**.

Шаблон **template1** используется по умолчанию для создания новых БД.

Шаблон **template0** не должен изменяться. Он нужен:

- 1) для восстановления БД из резервной копии, выполненной `pg_dump`,
- 2) при создании новой БД с кодировкой, отличной от указанной при инициализации кластера.

БД **postgres** (не рекомендуется удалять) используется при подключении по умолчанию пользователем **postgres**.



Схемы

Пространство имен для объектов

разделение объектов на логические группы

предотвращение конфликта имен между приложениями

Схема и пользователь — разные сущности

Специальные схемы

public — по умолчанию в ней создаются все объекты

pg_catalog — системный каталог

information_schema — вариант системного каталога

pg_temp — для временных таблиц

...



Табличные пространства

Табличное пространство (ТП) позволяет организовать логику размещения файлов объектов БД в файловой системе, это отдельный каталог с точки зрения файловой системы.

Одно ТП может использоваться несколькими БД. Каждой БД можно назначить как ТП по умолчанию, так и в каждом случае указывать, в каком ТП будет конкретный объект внутри БД.

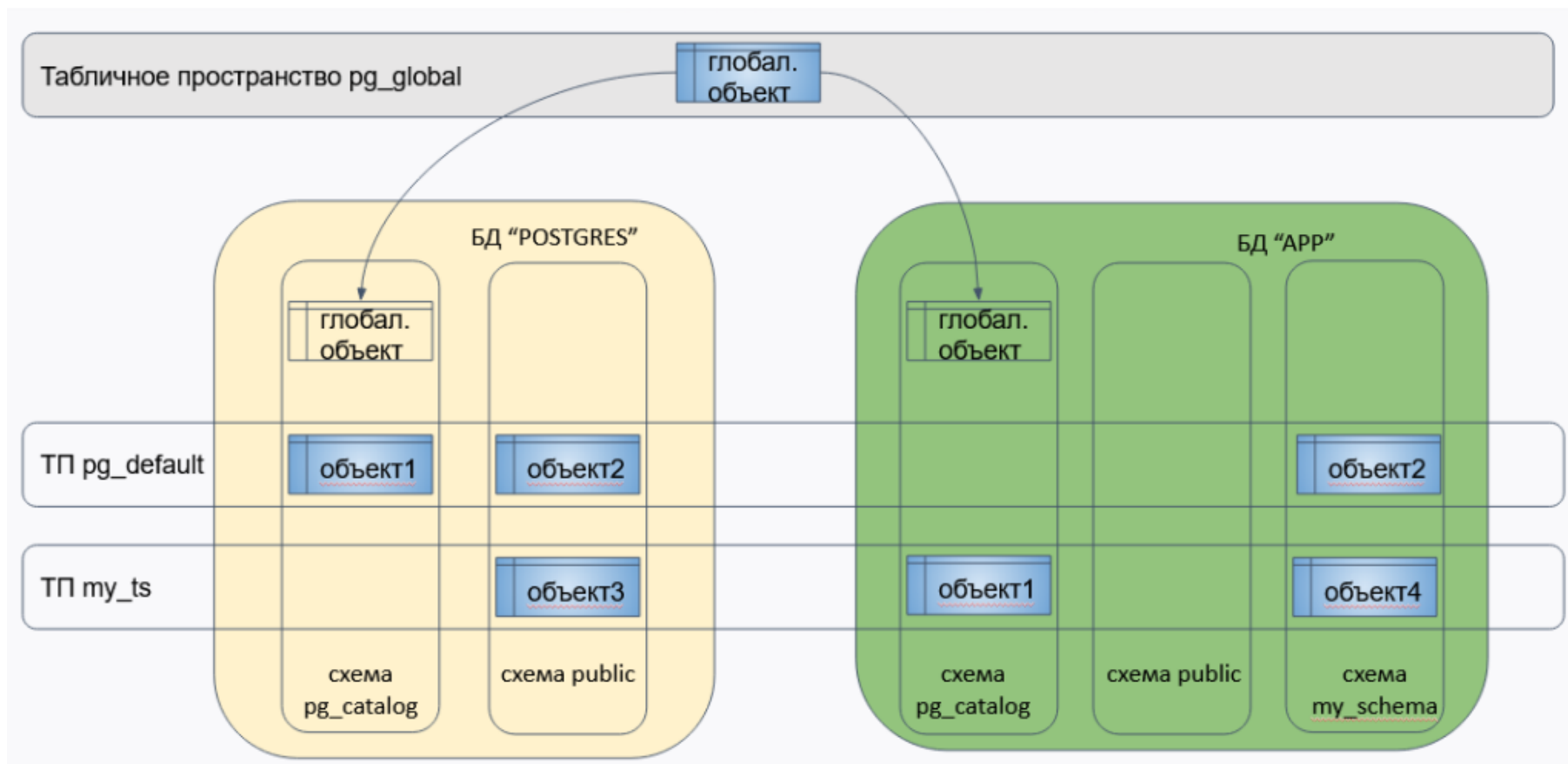
По умолчанию при установке СУБД создаются два табличных пространства:

`pg_default` - используется по умолчанию для баз данных `template1` и `template0`

`pg_global` - используется для общих системных каталогов.

Новые ТП по умолчанию создаются в каталоге `$PGDATA/pg_tblspc`. Можно создать свой каталог и своё ТП.



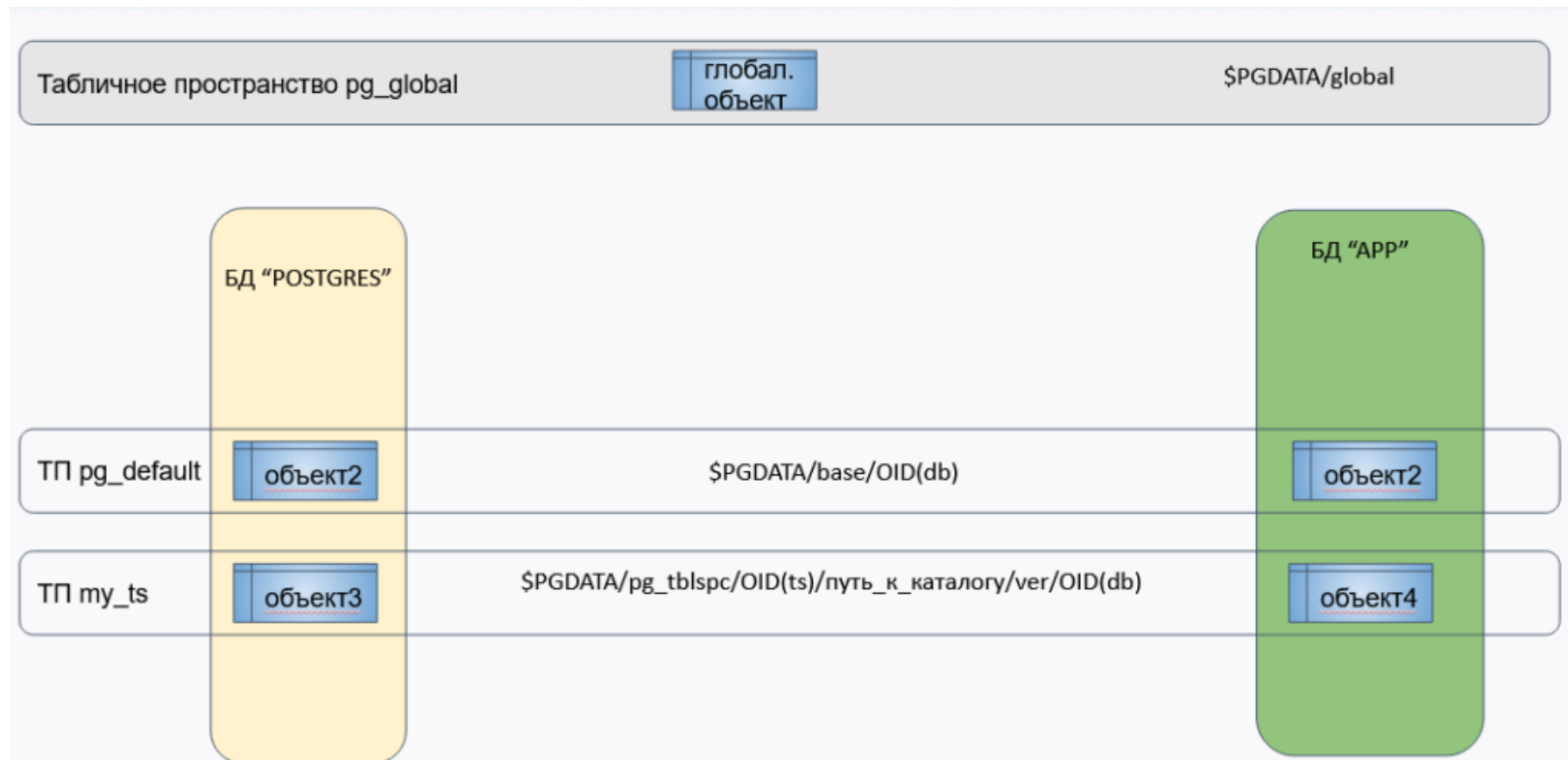


Объект может принадлежать только к одному табличному пространству (`pg_default` или др.), одной базе данных (`app` или `postgres`) и одной схеме. Только глобальные/системные объекты хранятся отдельно в табличном пространстве `pg_global` и присутствуют логически в каждой БД. Одному объекту в БД может соответствовать от одного и более файлов в зависимости от типа объекта и его размера.

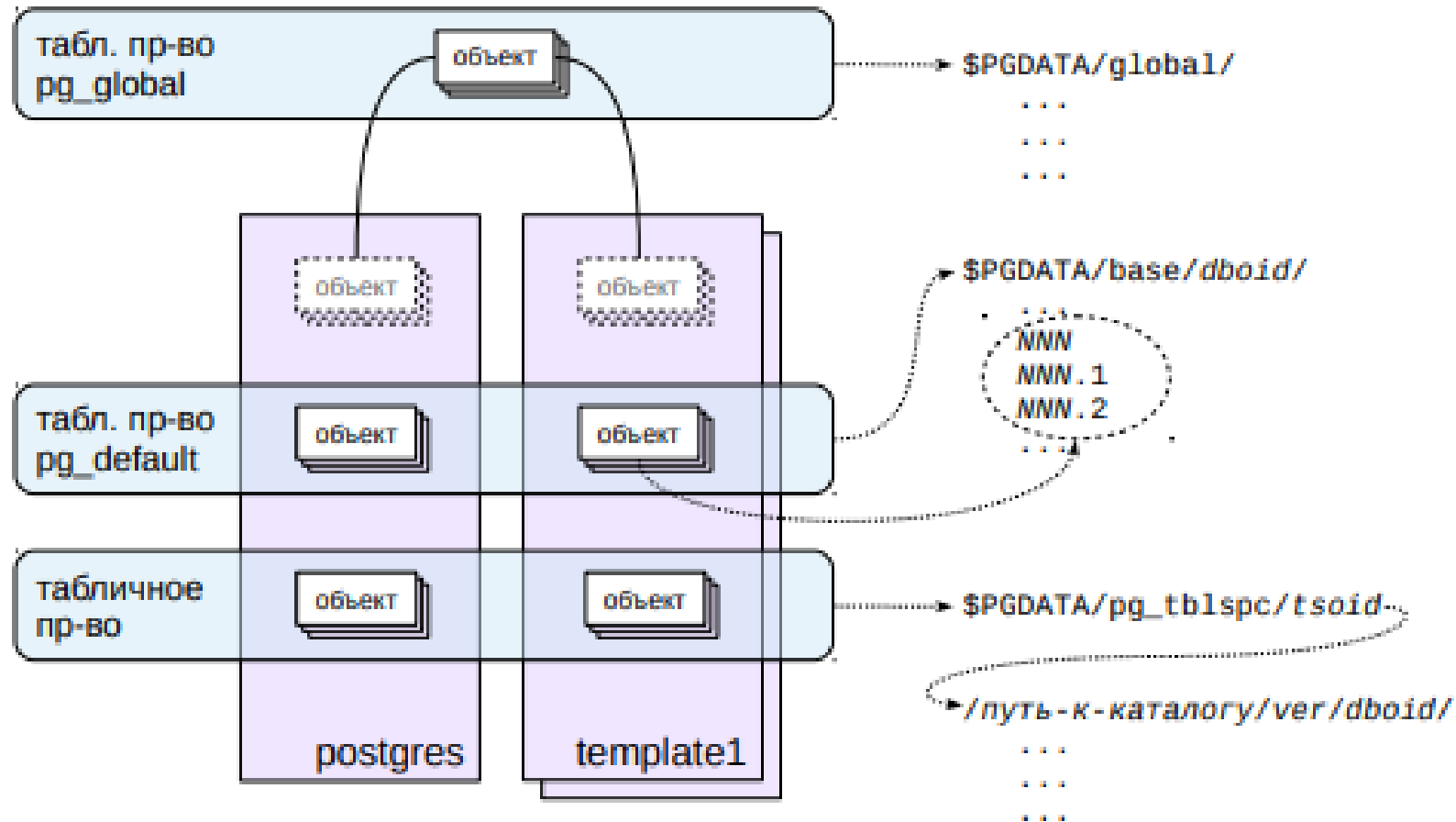


Файлы конфигурации и файлы данных, используемые кластером БД, обычно хранятся в каталоге PGDATA. В каталоге pg_database хранится информация о доступных БД, pg_database разделяется всеми БД кластера.

Внутри каталога с табличным пространством расположены каталоги с БД, названные по OID (идентификатор) БД из pg_database. Полный путь к объекту:



Каталоги, файлы, страницы



Слои

Основной

собственно данные

Карта видимости (vm)

отмечает страницы, на которых все версии строк видны во всех снимках
используется для оптимизации работы процесса очистки
и ускорения индексного доступа

существует только для таблиц

Карта свободного пространства (fsm)

отмечает свободное пространство в страницах после очистки
используется при вставке новых версий строк

Для каждой таблицы может быть создано до 3-х файлов:

- файл с данными - получает имя по номеру файлового узла таблицы (FFF, FFF — номер файлового узла);
- файл со свободными блоками - FFF_fsm
- файл с таблицей видимости - FFF_vm.



Toast

Версия строки должна помещаться на одну страницу

можно сжать часть атрибутов,
или вынести в отдельную TOAST-таблицу,
или сжать и вынести одновременно

TOAST-таблица

схема `pg_toast`
поддержана собственным индексом
«длинные» атрибуты разделены на части размером меньше страницы
читается только при обращении к «длинному» атрибуту
собственная версия
работает прозрачно для приложения



Физически

данные распределены по табличным пространствам (каталогам)

объект представлен несколькими слоями

каждый слой состоит из одного или нескольких файлов-сегментов

Табличными пространствами управляет администратор

Слои, файлы, TOAST — внутренняя кухня PostgreSQL



Путь поиска

Определение схемы объекта

квалифицированное имя (*схема.имя*) явно определяет схему
имя без квалификатора проверяется в схемах, указанных в пути поиска

Путь поиска

определяется параметром *search_path*, реальное значение
current_schemas()

не включаются несуществующие схемы и схемы, к которым нет доступа
схемы *pg_temp* и *pg_catalog* неявно включены первыми,
если не указаны в *search_path*

первая явно указанная в пути схема используется для создания объектов



Порядок поиска объектов

Чтобы временно установить порядок поиска объектов (таблиц, представлений, функций, триггеров), для которых не указана схема явно (unqualified name), можно использовать следующую команду:

```
SET search_path TO userschema, topology, public;
```

Первая указанная схема будет считаться схемой по умолчанию. Именно в ней будут создаваться новые объекты, если схема не будет указана явно.

Чтобы установить порядок поиска навсегда для пользователя, нужно использовать следующую команду:

```
ALTER USER user_name SET search_path TO userschema,topology,public;
```

Чтобы установить порядок поиска навсегда для базы данных нужно использовать следующую команду:

```
ALTER DATABASE "my_database" SET search_path TO userschema, topology, public;
```

Чтобы узнать текущий тип поиска, выполните следующую команду: **SHOW search_path**



Системный каталог

Описание всех объектов кластера

набор таблиц в каждой базе данных (схема pg_catalog)
и несколько глобальных объектов кластера
набор представлений для удобства

Доступ

запросы SQL, специальные команды psql

Правила организации

названия таблиц начинаются с pg_
имена столбцов содержат трехбуквенный префикс
в качестве ключей используются скрытые поля oid типа OID
названия объектов хранятся в нижнем регистре

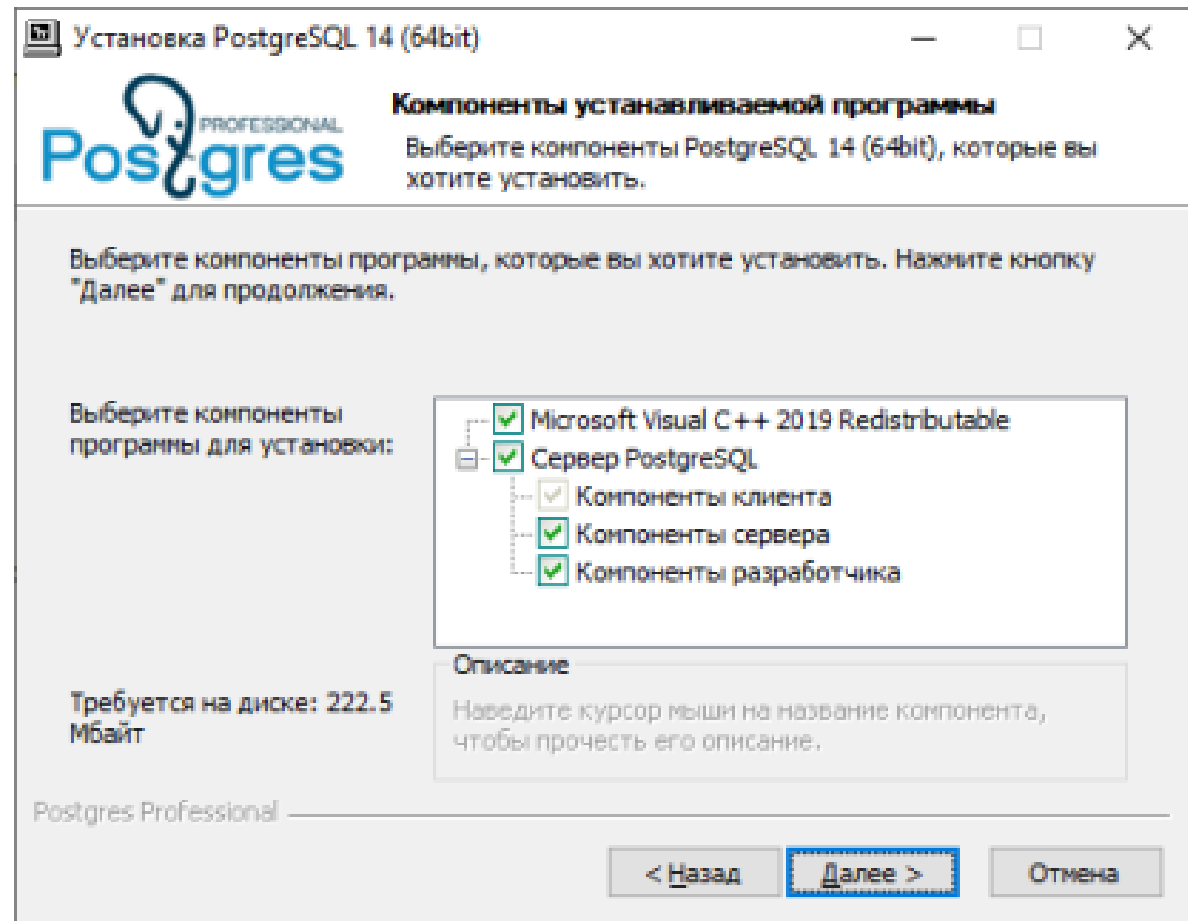


Установка в ОС Windows

Компоненты устанавливаемой программы

Скачайте установщик с и выберите язык установки (русский):
postgrespro.ru/windows.

Установщик построен в традиционном стиле «мастера»: вы можете просто нажимать на кнопку «Далее», если вас устраивают предложенные варианты.

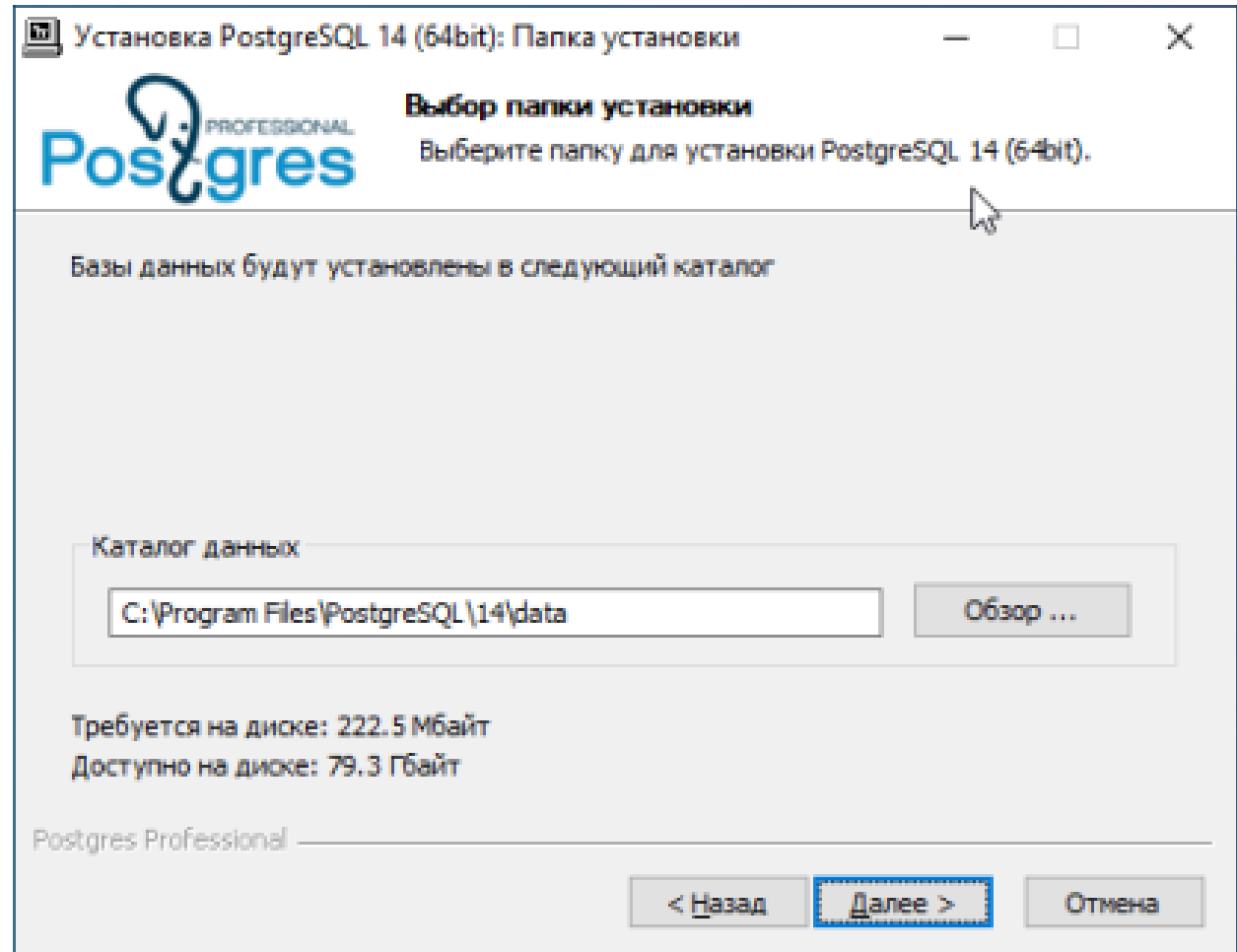


Установка в ОС Windows

Оставьте все флажки, если не уверены, какие выбрать.

Далее следует выбрать каталог для установки PostgreSQL.

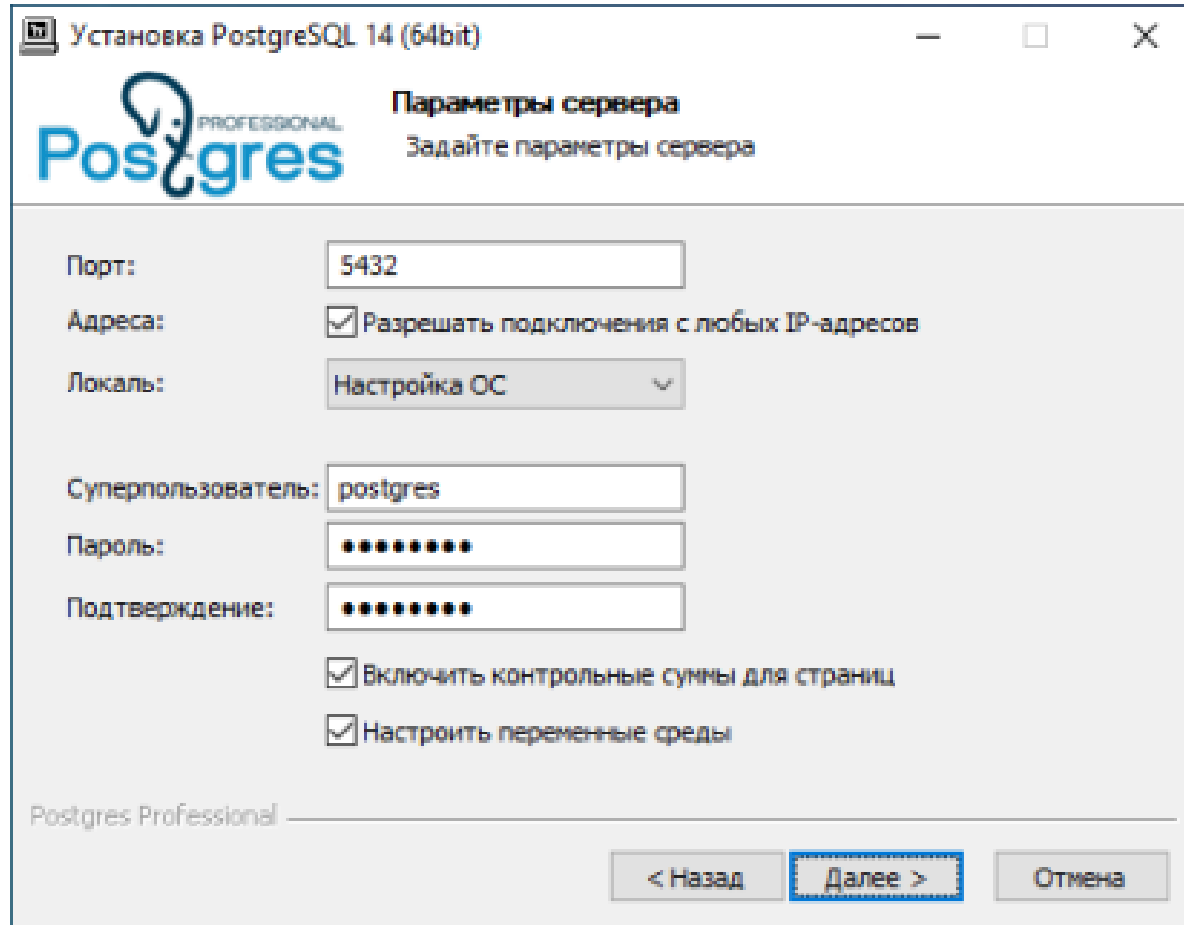
Отдельно можно выбрать расположение каталога для баз данных.



Установка в ОС Windows

Параметры сервера

Выберите локаль «Russian, Russia».
Введите (и подтвердите повторным вводом) пароль пользователя СУБД postgres. Отметьте флажок «Настроить переменные среды», чтобы подключаться к серверу PostgreSQL под текущим пользователем ОС. Остальные поля можно оставить со значениями по умолчанию.



The screenshot shows the 'Установка PostgreSQL 14 (64bit)' window with the 'Параметры сервера' (Server Parameters) tab selected. The window title is 'Установка PostgreSQL 14 (64bit)'. The PostgreSQL logo is visible on the left. The main title is 'Параметры сервера' and the subtitle is 'Задайте параметры сервера'. The form contains the following fields and options:

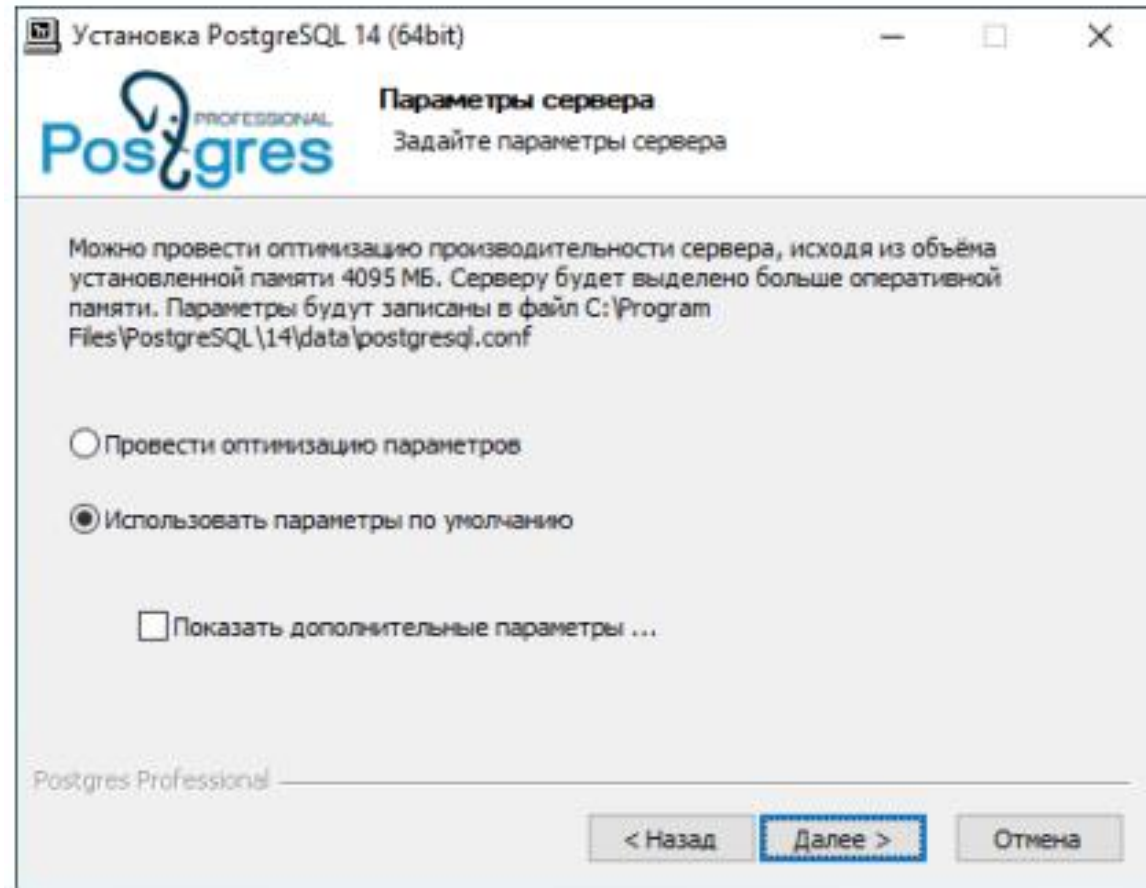
- Порт: 5432
- Адреса: ☒ Разрешать подключения с любых IP-адресов
- Локаль: Настройка ОС (dropdown menu)
- Суперпользователь: postgres
- Пароль: [masked with dots]
- Подтверждение: [masked with dots]
- ☒ Включить контрольные суммы для страниц
- ☒ Настроить переменные среды

At the bottom, there is a 'Postgres Professional' logo and three buttons: '< Назад', 'Далее >' (highlighted with a blue border), and 'Отмена'.

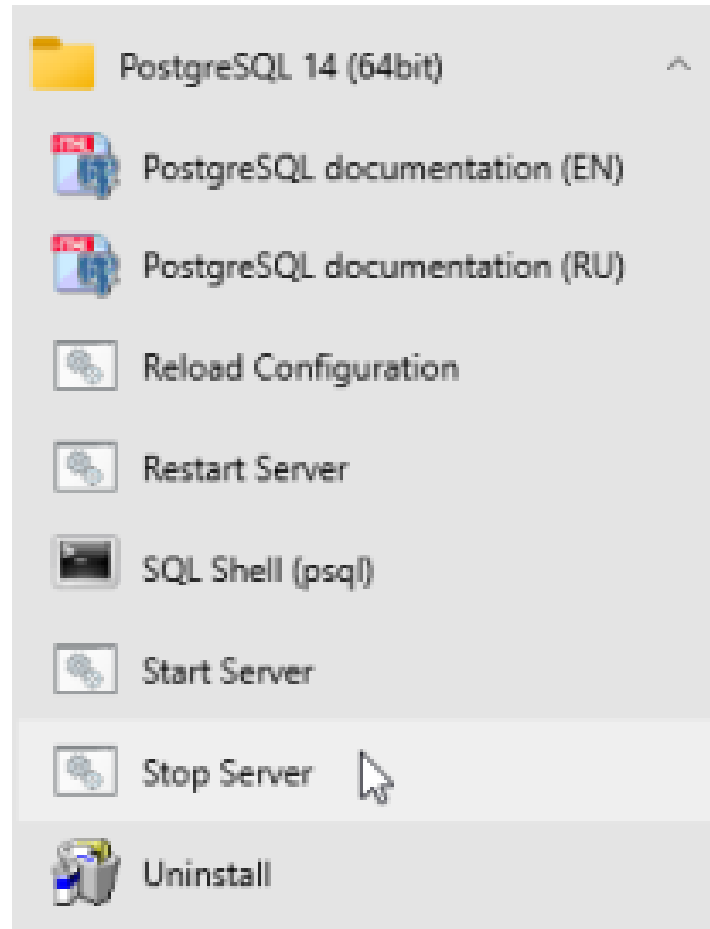


Установка в ОС Windows

Если планируете установить PostgreSQL только для ознакомительных целей, можно отметить вариант «Использовать параметры по умолчанию», чтобы СУБД не занимала много оперативной памяти.



Управление службой



При установке PostgreSQL в вашей системе регистрируется служба, например, «postgresql-16». Она запускается автоматически при старте компьютера под учетной записью Network Service (Сетевая служба). При необходимости вы можете изменить параметры службы с помощью стандартных средств Windows.

Чтобы временно остановить службу сервера баз данных, выполните программу «Stop Server» из папки в меню «Пуск», которую вы указали при установке.

Для запуска службы там же находится программа «Start Server».



Подключение к серверу БД

Для выполнения действий над данными в БД приложение должно установить соединение с сервером базы данных. Для этого требуется указывать:

- сетевой адрес сервера базы данных (имя или Ip-адрес, например, localhost);
- номер порта, через который производится соединение (обычно 5432);
- имя базы данных (имя, которое задает пользователь при создании БД);
- имя пользователя базы данных (например, postgres).

Эти параметры обычно записаны в конфигурационных файлах клиента.



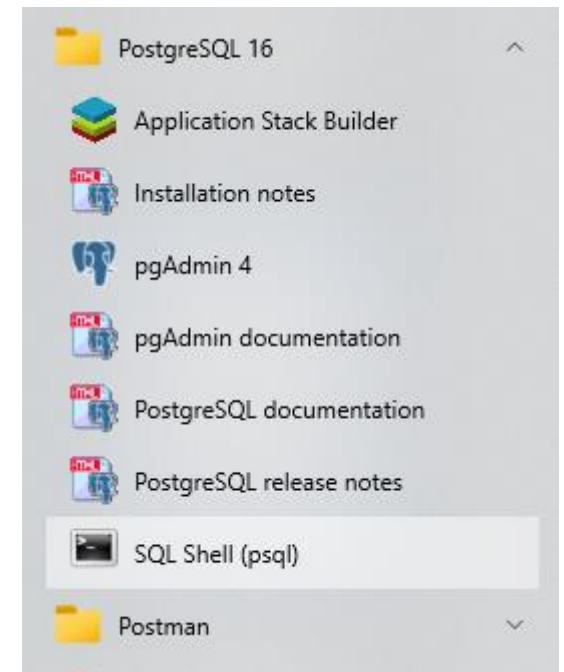
Основные файлы

- postgresql.conf — основной конфигурационный файл, содержащий значения параметров сервера;
- pg_hba.conf — файл, определяющий настройки доступа. В целях безопасности по умолчанию доступ должен быть подтвержден паролем и допускается только с локального компьютера.



Простой клиент: psql

- Для того чтобы работать с базой данных, необходима программа, выполняющая функции клиента.
- psql - текстовый клиент, входит в состав любого комплекта PostgreSQL в любой операционной системе. Для запуска программы psql нужно из командной строки операционной системы ввести команду
- `psql -d имя_бд -U имя_пользователя -h сервер -p порт`
- В ОС Windows запустите программу «SQL Shell (psql)» из меню «Пуск».
- В ответ на запрос введите пароль пользователя postgres, который вы указали при установке PostgreSQL.



Полезные команды psql

- \? Справка по командам psql.
- \h Справка по SQL: список доступных команд или
- синтаксис конкретной команды.
- \x Переключает традиционный табличный вывод (столбцы и строки) на расширенный (каждый столбец на отдельной строке) и обратно. Удобно для просмотра нескольких «широких»
- строк.
- \с имя - переключиться на БД с именем «имя».
- \l Список баз данных.
- \du Список пользователей.
- \dt, \dt *.* , \dt public.* Список таблиц, всех таблиц в схемах, таблиц в схеме public.
- \ds Список системных таблиц.
- \db Список табличных пространств.

- \di Список индексов.
- \dv Список представлений.
- \df Список функций.
- \dn Список схем.
- \dx Список установленных расширений.
- \dp Список привилегий.
- \d имя Подробная информация по конкретному объекту базы данных.
- \d+ имя Еще более подробная информация по конкретному объекту.
- \timing on Показывать время выполнения операторов.
- psql \! chsr 1251 Поменять кодировку на 1251.
- \encoding UTF8 Поменять кодировку на UTF8
- psql \i Путь_к_скрипту - Выполнить скрипт с именем Путь_к_скрипту
- \q Выход из программы

При добавлении + к команде можно получить расширенную информацию.



Упражнение 3.

2. Напишите общий скрипт для создания первоначальной структуры базы данных, разработанной на предыдущем практическом занятии.
3. Запустите скрипт с помощью psql (psql \i <путь к файлу.sql>), предварительно переименовав созданную ранее БД, убедитесь в его работоспособности.
4. Создайте описание базы данных, включающее перечисление ее таблиц, для каждой таблицы - характеристики столбцов, используя команды psql: \dt, \d.
5. Выполните все скрипты для модификации БД, написанные на предыдущем практическом занятии, в psql.
6. Посмотрите список табличных пространств с помощью psql кратко и подробно.
7. Добавьте с помощью pgAdmin 2 новых табличных пространства, создайте дополнительно несколько таблиц в новых табличных пространствах.
8. Добавьте с помощью pgAdmin новую схему в БД.

В результатах должны быть все использованные скрипты и команды psql.



Упражнение 4.

1. Создайте новую базу данных и подключитесь к ней.
2. Создайте схему, названную так же, как и пользователь.
3. Создайте схему `my_app`.
4. Создайте несколько таблиц в обеих схемах, соответствующих БД, созданной в предыдущей практической работе.
5. Получите в `psql` описание созданных схем и список всех таблиц в них.
6. Установите путь поиска так, чтобы при подключении к БД таблицы из обеих схем были доступны по неквалифицированному имени; приоритет должна иметь «пользовательская» схема.
7. Проверьте правильность настройки.

В результатах должны быть все получившиеся скрипты и команды `psql`.

