Оглавление

[2 ВАРИАНТЫ ЗАДАНИЙ ПО ЛАБОРАТОРНЫМ РАБОТАМ И ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТНЫХ МАТЕРИАЛОВ 1](#_Toc137916517)

[2.1 ВАРИАНТЫ ЛАБОРАТОРНЫХ РАБОТ 1](#_Toc137916518)

[2.1.1 ПРЕДМЕТНЫЕ ОБЛАСТИ, ИСПОЛЬЗУЕМЫЕ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ 2](#_Toc137916519)

[2.1.2 ПРОЦЕССЫ, РЕАЛИЗУЕМЫЕ ДЛЯ ПРЕДМЕТНЫХ ОБЛАСТЕЙ 6](#_Toc137916520)

[2.1.3 ¶ПРИЛОЖЕНИЯ БД, РЕАЛИЗУЕМЫЕ ДЛЯ ВЕДЕНИЯ БД 6](#_Toc137916521)

[2.2 КРАТКОЕ ОПИСАНИЕ ЛАБОРАТОРНЫХ РАБОТ 9](#_Toc137916522)

[2.3 АЛГОРИТМЫ АВТОМАТИЗИРУЕМЫХ ПРОЦЕССОВ, ИСПОЛЬЗУЕМЫХ В ВАРИАНТАХ ЛАБОРАТОРНЫХ РАБОТ 11](#_Toc137916523)

[2.4 ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТНЫХ МАТЕРИАЛОВ 12](#_Toc137916524)

[3 СОДЕРЖАНИЕ РАБОТ ПО ВЫПОЛНЕНИЮ ЗАДАНИЙ 14](#_Toc137916525)

[3.1 ЛАБОРАТОРНАЯ РАБОТА №1. АНАЛИЗ ПРО И РАЗРАБОТКА КОНЦЕПТУАЛЬНОЙ СХЕМЫ БД 14](#_Toc137916526)

[3.1.1 ЭТАПЫ РАЗРАБОТКИ КОНЦЕПТУАЛЬНОЙ СХЕМЫ ПРО 14](#_Toc137916527)

[3.1.2 ВЫДЕЛЕНИЕ СУЩНОСТЕЙ ПРО И ИХ АТРИБУТОВ 15](#_Toc137916528)

[3.1.3 ВЫДЕЛЕНИЕ СВЯЗЕЙ МЕЖДУ СУЩНОСТЯМИ И ОПРЕДЕЛЕНИЕ ИХ ТИПОВ 16](#_Toc137916529)

[3.1.4 КОНЦЕПТУАЛЬНАЯ СХЕМА БД 16](#_Toc137916530)

[3.1.5 СОДЕРЖАНИЕ ОТЧЕТА 17](#_Toc137916531)

[3.2 ЛАБОРАТОРНАЯ РАБОТА №2. УСТАНОВКА, НАСТРОЙКА И ИЗУЧЕНИЕ СУБД POSTGRESQL 17](#_Toc137916532)

[3.2.1 УСТАНОВКА СУБД POSTGRESQL 18](#_Toc137916533)

[3.2.2 СОЗДАНИЕ ПУСТОЙ БД 20](#_Toc137916534)

[3.2.3 СОЗДАНИЕ СХЕМЫ БД 23](#_Toc137916535)

[3.2.4 СОЗДАНИЕ ДОМЕНОВ 24](#_Toc137916536)

[3.2.5 ¶СОЗДАНИЕ ТАБЛИЦ 25](#_Toc137916537)

[3.2.6 Создание таблицы с помощью языка SQL 33](#_Toc137916538)

[3.2.7 ¶СОДЕРЖАНИЕ ОТЧЕТА 37](#_Toc137916539)

[3.3 ЛАБОРАТОРНАЯ РАБОТА №3. МОДИФИКАЦИЯ БД И ВЫПОЛНЕНИЕ МАНИПУЛЯЦИЙ С ДАННЫМИ БАЗЫ 38](#_Toc137916540)

[3.3.1 СОЗДАНИЕ ИНДЕКСОВ 38](#_Toc137916541)

[3.3.2 СОЗДАНИЕ ПОСЛЕДОВАТЕЛЬНОСТЕЙ 40](#_Toc137916542)

[3.3.3 ЗАПОЛНЕНИЕ ТАБЛИЦ БД С ПОМОЩЬЮ ОПЕРАТОРА INSERT 40](#_Toc137916543)

[3.4 ЛАБОРАТОРНАЯ РАБОТА №4. ВЫПОЛНЕНИЕ ОДНОТАБЛИЧНЫХ ЗАПРОСОВ SELECT 49](#_Toc137916544)

[3.5 ЛАБОРАТОРНАЯ РАБОТА №5. ВЫПОЛНЕНИЕ МНОГОТАБЛИЧНЫХ ЗАПРОСОВ SELECT 55](#_Toc137916545)

[3.6 ЛАБОРАТОРНАЯ РАБОТА №6. РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР 61](#_Toc137916546)

[3.7 ЛАБОРАТОРНАЯ РАБОТА №7. РАЗРАБОТКА ТРИГГЕРНЫХ ФУНКЦИЙ И ТРИГГЕРОВ 64](#_Toc137916547)

[4 БИБЛИОГРАФИЧЕСКИЙ СПИСОК 68](#_Toc137916548)

Для выполнения лабораторных работ предварительно требуется установить следующие свободно распространяемые программные средства:

* СУБД *PostgreSQL* – реляционную СУБД с открытым кодом и свободным распространением. *PostgreSQL* работает в системах *Unix* и *Windows*;
* язык *PL/pgSQL*, обеспечивающий написание кодов хранимых процедур, триггерных функций и триггеров, содержится в составе ус- тановленной СУБД *PostgreSQL*;
* среду программирования *NetBeens IDE* с открытым исходным кодом. Среда предоставляет все средства, необходимые для создания профессиональных приложений, корпоративных, мобильных и *Web*- приложений на платформе *Java* (а также *C/C++*, *PHP*, *Groovy*, *JavaS- cript* и *Ruby*).

¶

# ВАРИАНТЫ ЗАДАНИЙ ПО ЛАБОРАТОРНЫМ РАБОТАМ И ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТНЫХ МАТЕРИАЛОВ

## ВАРИАНТЫ ЛАБОРАТОРНЫХ РАБОТ

Для выполнения лабораторных работ для каждого студента определяется задание, составленное из вариантов:

* предметной области (ПрО), фрагмента внешнего мира, реализуемого в БД;
* автоматизируемого процесса, бизнес-процесса, осуществляю- щего изменение состояния БД;
* реализации приложения для ведения БД, обеспечивающего ввод в БД значений атрибутов бизнес-процесса.

### ПРЕДМЕТНЫЕ ОБЛАСТИ, ИСПОЛЬЗУЕМЫЕ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

В качестве вариантов (ПрО), бизнес-процессы которых требу ется автоматизировать с помощью информационной системы, рас сматриваются следующие.

1. ***ПрО «Сеть оптовых складов автомобилей»*** описывает дея тельность оптовых складов компании по приему поставляемых ав томобилей и их продажи и должна содержать следующие сущности:

* «Оптовые склады автомобилей» с указанием идентификатора, города и региона, в котором расположен каждый оптовый склад;
* «Автомобили на складах» с указанием идентификатора, наименования, цены и количества автомобилей различных марок на опто вых складах;
* «Сотрудники склада автомобилей» с указанием идентификато ра, фамилии, имени, должности, возраста каждого сотрудника, а

¶также идентификатора оптового склада, в котором сотрудник рабо тает;

* «Планы оптовых складов автомобилей» с указанием иденти фикатора оптового склада, даты составления планов, планов постав ки и продажи каждого оптового склада, данных о фактических по ставках и продажах оптового склада;
* «Планы сотрудников» с указанием идентификатора, даты со ставления планов, планов поставки и продажи каждого сотрудника, данных о фактических поставках и продажах сотрудников;
* «Поставки в оптовые склады автомобилей» с указанием иден- тификатора и даты поставки, идентификатора и количества товаров, идентификатора каждого сотрудника, принявшего товар;
* «Продажи оптовых складов автомобилей» с указанием иден- тификатора и даты продажи, идентификатора и количества товаров, идентификатора каждого сотрудника, продавшего товар.

1. ***ПрО «Сеть оптовых складов лекарств»*** описывает деятель- ность оптовых складов компании по приему поставляемых товаров и их продажи и должна содержать следующие сущности:

* «Оптовые склады лекарств» с указанием идентификатора, го- рода и региона, в котором расположен каждый оптовый склад;
* «Лекарства на складах» с указанием идентификатора, наиме- нования, цены и количества каждого вида товара на оптовом складе;
* «Сотрудники складов лекарств» с указанием идентификатора, фамилии, имени, должности, возраста каждого сотрудника, а также идентификатора оптового склада, в котором сотрудник работает;
* «Планы оптовых складов» с указанием идентификатора опто- вого склада, даты составления планов, планов поставки и продажи для каждого оптового склада, данных о фактических поставках и продажах оптового склада;
* «Планы сотрудников складов» с указанием идентификатора, даты составления планов, планов поставки и продажи сотрудника, данных о фактических поставках и продажах сотрудника;
* ¶«Поставки в оптовые склады лекарств» с указанием идентифи- катора и даты поставки, идентификатора и количества товаров, идентификатора каждого сотрудника, принявшего товар;
* «Продажи оптового склада лекарств» с указанием идентифика- тора и даты продажи, идентификатора и количества товаров, иден- тификатора каждого сотрудника, продавшего товар.

1. ***ПрО «Сеть оптовых складов продовольствия»*** описывает деятельность оптовых складов компании по приему поставляемых товаров и их продажи и должна содержать следующие сущности:

* «Оптовые склады продовольствия» с указанием идентифика- тора, города и региона, в котором расположен каждый оптовый склад компании;
* «Продовольствие на складах» с указанием идентификатора, наименования, цены и количества каждого вида товаров на оптовых складах;
* «Сотрудники складов продовольствия» с указанием идентифи- катора, фамилии, имени, должности, возраста сотрудников, а также идентификаторов оптовых складов, в которых сотрудники работают;
* «Планы оптовых складов продовольствия» с указанием иден- тификатора оптового склада, даты составления планов, планов по- ставок и продаж, а также данных о фактических поставках и прода- жах для каждого оптового склада;
* «Планы сотрудников складов» с указанием идентификатора, даты составления планов, планов поставок и продаж каждого сот- рудника, данных о фактических поставках и продажах сотрудников;
* «Поставки в оптовые склады продовольствия» с указанием идентификатора и даты поставки, идентификатора и количества то- варов, идентификатора сотрудника, принявшего товар;
* «Продажи оптовых складов продовольствия» с указанием идентификатора и даты продажи, идентификатора и количества товаров, идентификатора сотрудника, продавшего товар.

1. ***ПрО «Сеть автомобильных магазинов»*** описывает деятель- ность магазинов компании по приему поставляемых автомобилей и их продажи и должна содержать следующие сущности:

* «Автомобильные магазины» с указанием идентификатора, го- рода и региона, в котором расположен каждый магазин;
* «Автомобили на складах» с указанием идентификатора, на- именования, цены и количества товаров на складах;
* «Сотрудники автомобильных магазинов» с указанием иденти- фикатора, фамилии, имени, должности, возраста каждого сотрудни- ка, а также идентификатора магазина, в котором сотрудник работает;
* «Планы автомобильных магазинов» с указанием идентифика- тора магазина, даты составления планов, планов поставки и продажи магазина, данных о фактических поставках и продажах магазинов;
* «Планы сотрудников магазинов» с указанием идентификатора сотрудника, даты составления планов, планов поставки и продажи каждого сотрудника, данных о фактических поставках и продажах сотрудника;
* «Поставки в автомобильные магазины» с указанием иденти- фикатора и даты поставки, идентификатора и количества товаров, идентификатора сотрудника, принявшего товар;
* «Продажи автомобильных магазинов» с указанием идентифи- катора и даты продажи, идентификатора и количества товаров, идентификатора сотрудника, продавшего товар.

1. ***ПрО «Сеть аптек»*** описывает деятельность аптек компании по приему поставляемых лекарств и их продажи и должна содержать следующие сущности:

* «Аптеки» с указанием идентификатора, города и региона, в ко- тором расположена каждая аптека;
* «Лекарства на складах» с указанием идентификатора, наиме- нования, цены и количества лекарств на складах;
* «Сотрудники аптек» с указанием идентификатора, фамилии, имени, должности, возраста каждого сотрудника, а также иденти- фикатора аптеки, в котором сотрудник работает;
* ¶«Планы аптек» с указанием идентификатора аптеки, даты со- ставления планов, планов поставки и продажи аптеки, данных о фактических поставках и продажах аптеки;
* «Планы сотрудников аптек» с указанием идентификатора, да- ты составления планов, планов поставок и продаж для каждого со- трудника, данных о фактических поставках и продажах сотрудника;
* «Поставки в аптеки» с указанием идентификатора и даты по- ставок, идентификатора и количества лекарств, идентификатора ка- ждого сотрудника, принявшего лекарства;
* «Продажи аптек» с указанием идентификатора и даты прода- жи, идентификатора и количества лекарств, идентификатора каждо- го сотрудника, продавшего лекарства.

1. ***ПрО «Сеть продовольственных магазинов»*** описывает дея- тельность продовольственных магазинов компании по приему пос- тавляемых продуктов и их продажи и должна содержать следующие сущности:

* «Продовольственные магазины» с указанием идентификатора, города и региона, в котором расположен каждый магазин;
* «Товар на складах» с указанием идентификатора, наименова- ния, цены и количества товаров на складах;
* «Сотрудники магазинов» с указанием идентификатора, фами- лии, имени, должности, возраста каждого сотрудника, а также иден- тификатора продовольственного магазина, в котором он работает;
* «Планы магазинов» с указанием идентификатора, даты состав- ления планов, планов поставок и продаж магазинов, данных о фак- тических поставках и продажах магазинов;
* «Планы сотрудников» с указанием идентификатора, даты со- ставления планов, планов поставок и продаж каждого сотрудника, данных о фактических поставках и продажах сотрудника;
* «Поставки в магазины» с указанием идентификатора и даты поставок, идентификатора и количества товара, идентификатора со- трудника, принявшего товар;
* «Продажи магазинов» с указанием идентификатора и даты продаж, идентификатора и количества товаров, идентификатора со- трудника, продавшего товар.

***Примечание***. В качестве аналога для разработки концептуаль- ной схемы БД используйте учебную БД, описанную в курсе лекций [1]. Копирование текста описания, а также использование наимено- ваний сущностей и атрибутов из работы [1] в отчетах по выполне- нию лабораторных работ не допускается.

Для приведенных описаний ПрО необходимо создать БД. При этом каждый студент должен разрабатывать концептуальную схему БД ***самостоятельно.*** Это означает, ***что не допускается наличие совпадающих наименований сущностей и атрибутов*** в отчетах разных студентов.

### ПРОЦЕССЫ, РЕАЛИЗУЕМЫЕ ДЛЯ ПРЕДМЕТНЫХ ОБЛАСТЕЙ

Во всех описанных в предыдущем подразделе ПрО должны быть реализованы процессы учета приема и продажи товаров в оптовых складах, магазинах и аптеках.

Моделирование *процесса «Учет приема товаров»* должно быть реализовано путем:

* записи параметров поставки в соответствующие таблицы БД;
* разработки кодов хранимых процедур, реализующих измене- ния в БД после записи значений параметров поставки;
* разработки триггерной функции, организующей упорядочен- ное выполнение хранимых процедур, реализующих изменения в БД;
* разработки триггера, срабатывающего после записи значений параметров поставки.

Моделирование *процесса «Учет продаж товаров»* должно быть реализовано путем:

* записи параметров продажи в соответствующие таблицы БД;
* разработки кодов хранимых процедур, реализующих измене- ния в БД после записи значений параметров продажи;
* разработки триггерной функции, организующей упорядочен- ное выполнение хранимых процедур, реализующих изменения в БД;
* разработки триггера, срабатывающего после записи значений параметров продажи.

### ¶ПРИЛОЖЕНИЯ БД, РЕАЛИЗУЕМЫЕ ДЛЯ ВЕДЕНИЯ БД

**Сущности-процессы БД.** Процессы, описанные в предыдущем разделе, запускаются путем выполнения записи в таблицу, для кото- рой реализован триггер, срабатывающий при выполнении операции *insert*. Таковыми, в зависимости от вашего варианта являются сущ- ности-процессы:

* «Учет приема товаров автомобильного склада»;
* «Учет продаж товаров автомобильного склада»;
* «Учет приема товаров склада лекарств»;
* «Учет продаж товаров склада лекарств»;
* «Учет приема товаров склада продовольствия»;
* «Учет продаж товаров склада продовольствия»;
* «Учет приема товаров автомобильного магазина»;
* «Учет продаж товаров автомобильного магазина»;
* «Учет приема лекарств аптекой»;
* «Учет продаж лекарств аптекой»;
* «Учет приема товаров продовольственным магазином»;
* «Учет продаж товаров продовольственным магазином». Приведенные длинные названия сущностей-процессов не следу-

ет повторять в своих логических схемах БД ваших вариантов. Эти названия приведены для максимального раскрытия существа обо- значаемых объектов, а руководствоваться при именованиях этих сущностей рекомендациями, изложенными в разделе 1.4.

**Приложение БД.** Для работы пользователей с созданными БД необходимо разработать приложение БД и реализовать его в среде *NetBeans*. Процедура вставки данных процесса, осуществляемая приложением, должна привести к срабатыванию триггера, опреде- ляющего последовательность вызовов хранимых процедур БД, осу- ществляющих согласованное изменение данных базы с целью со- хранения ее целостности после выполнения соответствующего про- цесса (описание процесса разработки приложения БД см. в разделе

9.3 пособия [1] и в работах [8, 9]).

¶В зависимости от вашего варианта для записи значений атрибу- тов процессов в БД ваше приложение БД должно использовать либо интерфейс *Statement*, либо интерфейс *PreparedStatement*.

**Использование интерфейса *Statement*.** Для записи значений параметров процессов поставки или продажи, архивирования дан- ных за прошедший период и планирования поставки или продажи на следующий период путем их записи в БД необходимо реализовать приложение, использующее для взаимодействия с БД, интерфейса *Statement* (описание применения данного интерфейса см. в разделе

* 1. пособия [1] и в работах [8, 9]).

**Использование интерфейса *PreparedStatement*.** Для записи значений параметров процессов поставки или продажи, архивирова- ния данных за прошедший период и планирования поставки или продажи на следующий период путем их записи в БД необходимо реализовать приложение, использующее для взаимодействия с БД, интерфейса *PreparedStatement* (описание применения данного ин- терфейса см. в разделе 9.3 пособия [1] и в работах [8, 9]).

* + 1. ВАРИАНТЫ ЛАБОРАТОРНЫХ РАБОТ

Варианты выполнения лабораторных работ приведены в сле- дующей таблице.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ вар** | **ПрО** | **Автоматизируемый процесс** | **Интерфейс JDBC** |
| 1 | Автомоб. склад | Учет приема автомобилей | Statement |
| 2 | Автомоб. склад | Учет приема автомобилей | Prepared- Statement |
| 3 | Автомоб. склад | Учет продаж автомобилей | Statement |
| 4 | Автомоб. склад | Учет продаж автомобилей | Prepared- Statement |
| 5 | Склад лекарств | Учет приема лекарств | Statement |
| 6 | Склад лекарств | Учет приема лекарств | Prepared- Statement |
| 7 | Склад лекарств | Учет продаж лекарств | Statement |
| 8 | Склад лекарств | Учет продаж лекарств | Prepared- Statement |
| 9 | Склад продовольствия | Учет приема продовольствия | Statement |
| 10 | Склад продовольствия | Учет приема продовольствия | Prepared- Statement |
| 11 | Склад продовольствия | Учет продаж продовольствия | Statement |
| 12 | Склад продовольствия | Учет продаж продовольствия | Prepared- Statement |
| 13 | Автомагазин | Учет приема автомобилей | Statement |
| 14 | Автомагазин | Учет приема автомобилей | Prepared- Statement |
| 15 | Автомагазин | Учет продаж автомобилей | Statement |
| 16 | Автомагазин | Учет продаж автомобилей | Prepared- Statement |
| 17 | Аптека | Учет приема лекарств | Statement |
| 18 | Аптека | Учет приема лекарств | Prepared- Statement |
| 19 | Аптека | Учет продаж лекарств | Statement |
| 20 | Аптека | Учет продаж лекарств | Prepared- Statement |
| 21 | Прод. магазин | Учет приема продовольствия | Statement |
| 22 | Прод. магазин | Учет приема продовольствия | Prepared- Statement |
| 23 | Прод. магазин | Учет продаж продовольствия | Statement |
| 24 | Прод. магазин | Учет продаж продовольствия | Prepared- Statement |

## КРАТКОЕ ОПИСАНИЕ ЛАБОРАТОРНЫХ РАБОТ

Описываемые лабораторные работы составлены таким образом, чтобы охватить все этапы разработки информационной системы. С целью освоения навыков разработки информационных систем на основе БД в ходе выполнения работ вы должны:

* определить домены, таблицы, индексы, представления и по- следовательности, а также связи между таблицами. На данном этапе создается БД;
* продемонстрировать выполнение операций манипуляции и вы- борки, включая вложенные запросы, запросы с агрегатными функ- циями, запросы с группировкой, многотабличные запросы. На этом этапе осуществляется демонстрация работы с созданной БД;
* определить хранимые процедуры, триггеры, триггерные функ- ции и приложения БД в среде NetBeans. На последнем этапе осваи- ваются технологии разработки программного кода для ведения БД в ходе эксплуатации информационной системы.

Для реализации этих этапов выполняются следующие работы.

В лабораторной работе №1:

* анализ ПрО, заданной в вашем варианте. В ходе анализа ПрО выявляются сущности (информационные объекты), соответствующие требованиям нормализации данных. Среди информационных объектов выделите сущности-объекты и сущности-процессы;
* определение связей между выделенными сущностями и кон- цептуальной модели ПрО в виде диаграммы.

В лабораторной работе №2:

* установка СУБД PostgreSQL;
* создание пустой БД;
* создание схем БД;
* создание доменов;
* создание таблиц вашей БД;
* установка связей между таблицами и определение правил со- хранения ссылочной целостности для каждой связи.

¶В лабораторной работе №3:

* создание индексов;
* создание последовательностей;
* заполнение таблиц БД с помощью оператора INSERT. Должно быть заполнено не менее 5 строк в каждой таблице;
* обновление строк таблиц с помощью оператора UPDATE;
* удаление строк таблиц с помощью оператора DELETE;
* создание представлений.

В лабораторной работе №4:

* выборка данных из таблиц с помощью оператора SELECT;
* использование агрегатных функций;
* использование запросов с группировкой.

В лабораторной работе №5 демонстрируется использование запросов SELECT:

* во вложенных запросах;
* многотабличных запросов с внутренним объединением таблиц;
* многотабличных запросов с внешним объединением таблиц.

В лабораторной работе №6 демонстрируется использование в хранимых процедурах следующих компонентов языка :

* команд SELECT INTO;
* атрибутов %TYPE;
* атрибутов %ROWTYPE;
* условных команд;
* циклов;
* функций, обеспечивающих сохранение целостности функционально зависимых данных вашей БД.

В лабораторной работе №7:

* создание триггерных функций и триггеров;
* разработка триггеров, обеспечивающих организацию последо- вательного выполнения хранимых процедур вашей БД.

В лабораторной работе №8:

* установка платформы NetBeans;
* разработка приложения для вставки записей в таблицу-процесс вашего варианта в среде NetBeans.

Более детальное содержание работ по выполнению перечислен ных заданий приведено в Главе 2 настоящих указаний.

## АЛГОРИТМЫ АВТОМАТИЗИРУЕМЫХ ПРОЦЕССОВ, ИСПОЛЬЗУЕМЫХ В ВАРИАНТАХ ЛАБОРАТОРНЫХ РАБОТ

В приведенных в подразделе [1.1.1](#_bookmark3) вариантах используются следующие автоматизируемые процессы:

* поступление товара;
* продажа товара.

Алгоритмы выполнения этих процессов описываются следующим образом.

Поступление товара. Поступление товара моделируется введением данных о поставке в соответствующую таблицу БД и реализацией изменения в других ее таблицах, вызванных событием поставки товара. При этом должно быть реализовано:

* выполнение оператора INSERT с данными о поставке в соот- ветствующей таблице вашей БД;
* разработка хранимой процедуры, реализующей увеличение количества поставленного товара на складе;
* разработка хранимой процедуры, реализующей увеличение суммы фактических поставок сотрудника, принявшего поставлен- ный товар;
* разработка хранимой процедуры, реализующей увеличение суммы фактических поставок магазина, в котором работает сотрудник, принявший поставленный товар;
* разработка триггерной функции, обеспечивающей последовательное выполнение хранимых процедур, реализующих изменения в БД по факту поставки товара;
* разработка триггера, обеспечивающего вызов триггерной функции после выполнения оператора INSERT с данными о поставке;
* разработка приложения для вставки записей в таблицу-процесс вашего варианта в среде NetBeans.

Продажа товара. Продажа товара моделируется введением дан- ных о продаже в соответствующую таблицу БД и реализацией изме- нения в других ее таблицах, вызванных событием продажи товара. При этом должно быть реализовано:

* выполнение оператора INSERT с данными о продаже в соот- ветствующей таблице вашей БД;
* разработка хранимой процедуры, реализующей уменьшение количества проданного товара на складе;
* разработка хранимой процедуры, реализующей увеличение суммы фактических продаж сотрудника, продавшего товар, в соответствующей таблице БД;
* разработка хранимой процедуры, реализующей увеличение суммы фактических продаж магазина, в котором работает сотрудник, продавший товар, в соответствующей таблице БД;
* разработка триггерной функции, обеспечивающей последовательное выполнение хранимых процедур, реализующих изменения в БД по факту продажи товара;
* разработка триггера, обеспечивающего вызов триггерной функции после выполнения оператора INSERT с данными о продаже;
* разработка приложения для вставки записей в таблицу-про- цесс вашего варианта в среде NetBeans.

Примечание. В качестве справочных данных при реализации ваших автоматизируемых процедур используйте материалы Приложения учебного пособия [1] в части описания хранимых процедур, триггерных функций и триггеров (см. пункты П.1.2.6, П.1.2.7, П.1.2.8) и разработки приложений для ведения БД (см. подразделы П.2.1, П.2.2).

## ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТНЫХ МАТЕРИАЛОВ

По каждой лабораторной работе должен быть оформлен отчет, содержащий следующие разделы:

1. Титульный лист.
2. Цель выполнения работы.
3. Описание содержания выполненных работ.
4. Выводы.

На титульном листе укажите номер варианта, название автоматизируемого процесса, а также название интерфейса JDBC, назначенного для вашего варианта. Образец заполнения титульного листа приведен на Рис. 1.1.

Во втором разделе опишите, какие навыки должны быть приобретены в результате выполнения работы.

В разделе описания содержания работ, структура которого зависит от выполняемой работы, должны быть приведены:

* текст, поясняющий ход выполнения работы, разбивается на разделы и подразделы в зависимости от выполняемой лабораторной работы (см. ниже);
* в тексте описания должны быть приведены листинги разработанного кода (при их наличии);
* в тексте описания должны быть приведены скриншоты, иллюстрирующие практический ход выполнения и результаты работы.
* В заключении (раздел «Выводы») должны быть подытожены результаты выполненной работы.

В части описания процесса разработки БД должны соблюдаться следующие требования:

* при разработке логической схемы БД ПрО для обозначения сущностей и их атрибутов использовать наименования на русском языке на латинице (следует отметить, что СУБД PostgreSQL до- пускает именование объектов БД и на кириллице. Но именование объектов логической схемы на латинице облегчает процедуры им- порта/экспорта объектов создаваемой БД в/из БД других СУБД);
* при разработке концептуальной схемы БД ПрО для обозначе- ния сущностей и их атрибутов использовать наименования на рус- ском языке на кириллице;
* наименования сущностей и их атрибутов должны быть по возможности краткими (не более 15 букв).

# СОДЕРЖАНИЕ РАБОТ ПО ВЫПОЛНЕНИЮ ЗАДАНИЙ

Отчеты должны выполняться самостоятельно: копирование текста с данных учебно-методических указаний, а также с других отчетов не допускается.

Описываемые в учебно-методических указаниях лабораторные работы охватывают следующие этапы разработки БД:

* проектирования БД, в рамках которой разрабатывается концеп- туальная схема предметной области (ПрО);
* разработки логической схемы БД и ее объектов (доменов, таб- лиц, представлений, индексов и т. д.);
* заполнения БД и выполнения операций манипулирования дан- ными базы на языке SQL;
* выполнения однотабличных запросов по выборке данных;
* выполнения многотабличных запросов по выборке данных;
* разработки хранимых процедур;
* разработки триггеров;
* разработки программ ведения БД.

В данных указаниях также описываются возможности, повышающие эффективность БД, созданных на базе СУБД PostgreSQL, в том числе:

* использование ограничений, обеспечивающих снижение объема программного кода обработки данных;
* создание последовательностей, облегчающих ввод данных;
* создание индексов, ускоряющих поиск данных и т. д.

Описание порядка и содержания выполнения лабораторных работ в данных методических указаниях осуществлено на примере учебной БД для ПрО, описанной в работе [1].

## ЛАБОРАТОРНАЯ РАБОТА №1. АНАЛИЗ ПРО И РАЗРАБОТКА КОНЦЕПТУАЛЬНОЙ СХЕМЫ БД

Целью данной работы является овладение навыками анализа ПрО и разработки концептуальной схемы БД для заданного вариан- та лабораторной работы.

Перед выполнением данной лабораторной работы внимательно изучите раздел 2.2 работы [1, 2].

### ЭТАПЫ РАЗРАБОТКИ КОНЦЕПТУАЛЬНОЙ СХЕМЫ ПРО

Определение сущностей, их свойств и связей между ними путем анализа ПрО осуществляется без использования средств автоматизированного проектирования, а созданная в результате выполнения данного этапа модель используется в качестве исходных данных для разработки схемы БД и называется концептуальной моделью ПрО. Элементы и связи концептуальной модели обозначаются средствами естественного языка.

Любой документ, используемый в ходе деятельности организации или предприятия, содержит информацию, описывающую сущности, представленные набором количественных параметров, называемых показателями. Поэтому сведения, необходимые для построения концептуальной модели БД, извлекаются путем дискретизации информации, содержащейся в документах, обеспечивающих деятельность автоматизируемой ПрО.

С целью выявления показателей документа проводится анализ каждого документа, в ходе которого определяется совокупность описываемых сущностей, а также их характеристик или реквизитов. На основе реквизитного состава определяются показатели, отра- жающие содержание документа. При этом выполняются следующие этапы:

1. Анализ системы документов ПрО.
2. Выделение реквизитов (характеристик), показателей, отне- сенных к тем или иным сущностям.
3. Определение связей между сущностями.

### ВЫДЕЛЕНИЕ СУЩНОСТЕЙ ПРО И ИХ АТРИБУТОВ

В результате анализа системы документов составляется описание ПрО, охватывающее ее реальные сущности и процессы и определяющее все необходимые источники информации для удовлетво- рения предполагаемых запросов пользователя и потребности в об- работке данных.

Пример составления описания ПрО приведен в разделе 2.2 учебного пособия, в котором показано, как по этому описанию определяются сущности, их атрибуты, а также связи между сущностя- ми. При описании выполняемой лабораторной работы каждая выделенная сущность должна быть обоснована с точки зрения необходимости ее использования при автоматизации бизнес-процессов ПрО.

Следует отметить, что описание ПрО с выделенными сущностями и их атрибутами уже содержится в описаниях вариантов ПрО, приведенных в подразделе [1.1.1](#_bookmark3) настоящих методических указаний, но каждый студент должен иметь представление о всех этапах раз- работки концептуальной схемы ПрО. Поэтому после изучения теории анализа структуры документа студент выполняет:

1. присвоение имен сущностям и их характеристикам по описанию своего варианта;
2. определение связей между сущностями, т. е. выявление зависимостей между реквизитами (характеристиками) сущностей;
3. определение и обоснование правил сохранения ссылочной целостности для каждой связи.

### ВЫДЕЛЕНИЕ СВЯЗЕЙ МЕЖДУ СУЩНОСТЯМИ И ОПРЕДЕЛЕНИЕ ИХ ТИПОВ

После присвоения имен сущностям ПрО и их характеристикам требуется установить связи между выявленными объектами ПрО, а для этого необходимо установить функциональные зависимости между атрибутами объектов или реквизитами (см. подраздел 2.5.1 пособия [1]).

Связь устанавливается между двумя информационными объектами, если экземпляры этих информационных объектов логически взаимосвязаны. Например, возьмем два объекта: «Магазин» и «Со- трудник» из описания ПрО в учебном пособии.

Связь экземпляров этих объектов определяется тем обстоятельством, что каждый сотрудник работает в каком-либо магазине, что является одним из свойств сотрудника. В описываемом примере этим свойством является «Идентификатор магазина». С другой стороны, этим свойством обладает и объект «Магазин» (совпадение наименований таких свойств необязательно, хотя и облегчает понимание схемы).

Связь экземпляров объектов «Магазин» и «Сотрудник» определяется совпадением значений свойств «Идентификатор магазина» этих объектов. Например, если значение свойства «Идентификатор магазина» экземпляра объекта «Сотрудник» равно «1», то это означает, что данный сотрудник связан с экземпляром объекта

«Магазин», значение свойства «Идентификатор магазина» которого также равно «1». Это означает, что он работает в данном конкретном магазине.

### КОНЦЕПТУАЛЬНАЯ СХЕМА БД

На основе выявленных сущностей ПрО, их свойств и связей между объектами строится концептуальная модель, которая отображает данные ПрО в виде диаграммы, содержащей совокупности информационных объектов и связей между ними. Эта модель является основой для разработки логической модели БД.

В разделе 2.2 учебного пособия приведено описание концептуальной схемы данных, наглядно отображающей логическую структуру БД. На этой схеме прямоугольники отображают таблицы БД, а список их полей приведен в описании схемы. Связи концептуальной схемы показывают, по каким полям осуществляется взаимосвязь таблиц. Имена ключевых полей для наглядности выделены и находятся в верхней части полного списка полей каждой таблицы.

По полученной модели далее строится база данных, описываемая в следующей лабораторной работе.

### СОДЕРЖАНИЕ ОТЧЕТА

После выполнения предписанных в данном разделе действий составьте отчет, содержащий следующие разделы.

1. Цель выполнения работы.
2. Описание содержания выполненных работ.
   1. Разработка концептуальной схемы ПрО.
      1. Описание анализа ПрО и процесса выделения ее сущно- стей в рамках требований вашего варианта, т. е. привести обоснова- ние для каждой сущности вашего варианта с точки зрения ее необ- ходимости для организации бизнес-процессов автоматизации дея- тельности вашей ПрО.
      2. Определение перечня атрибутов для каждой сущности, т. е. обоснование их необходимости для организации бизнес-процессов автоматизации деятельности вашей ПрО. В окончательный пере- чень должны быть включены только атрибуты, необходимые для организации бизнес-процессов автоматизации деятельности вашей ПрО.
      3. Определение атрибутов, для которых необходимо устано- вить ограничения на значения. Для последующего определения до- мена на основе этого ограничения определите характер описывае- мых ограничений.
      4. Определение связей между сущностями путем выявления атрибутов первичного (для родительской таблицы) и вторичного (для дочерней таблицы) ключей связываемых таблиц.
   2. Приведите диаграмму концептуальной схемы (пример диа- граммы концептуальной схемы см. в разделе 2.2 учебного пособия).
3. Выводы.

## ЛАБОРАТОРНАЯ РАБОТА №2. УСТАНОВКА, НАСТРОЙКА И ИЗУЧЕНИЕ СУБД POSTGRESQL

Целью данной работы является:

1. установка и настройка свободно распространяемой СУБД PostgreSQL;
2. физическая реализация концептуальной схемы в виде логической схемы БД с помощью установленной СУБД.

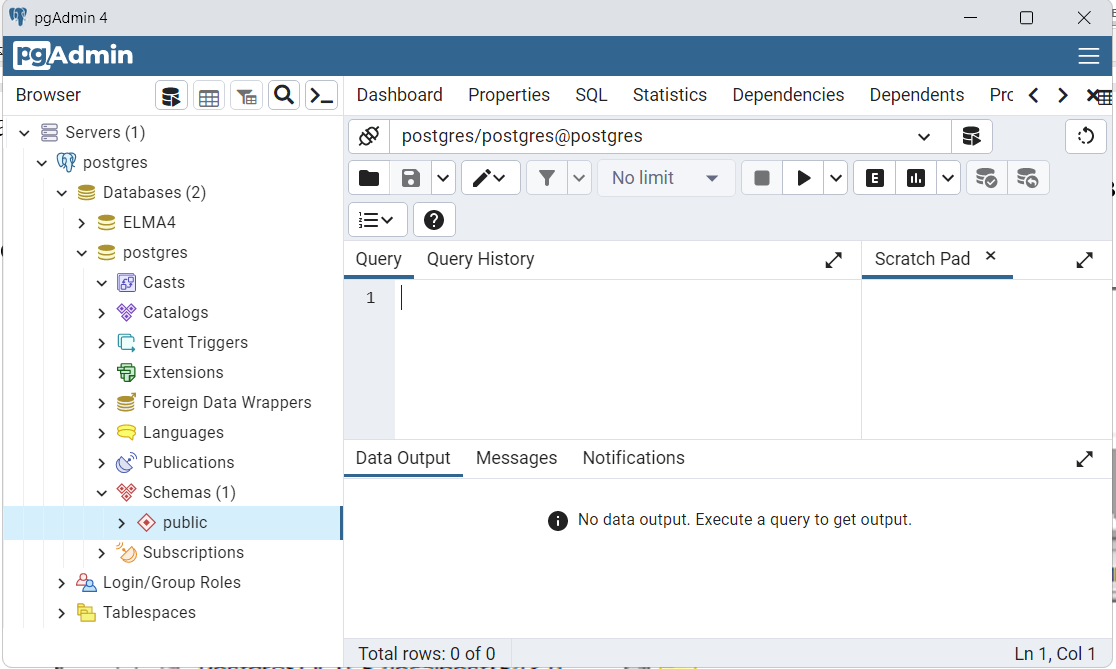
### УСТАНОВКА СУБД POSTGRESQL

Для установки СУБД PostgreSQL на ваш компьютер войдите на сайт <https://www.postgresql.org> и скачайте одну из версий СУБД для Windows. В данных методических указаниях используется версия PostgreSQL 12 (рекомендуется), на которой установлен графический интерфейс pgAdmin 4.

Установка данного программного средства не требует отдельного описания; просто следуйте указаниям программы-установщика. В ходе установки введите пароль (для учебной БД рекомендуется выбрать простой пароль, например, admin), а в выпадающем списке Locale выберите Russia.

Желательно выбрать для установки полную версию дистрибутива, включающую в свой состав необходимые драйверы. Полные версии дистрибутивов можно определить по объему памяти.

Если, по каким-либо причинам, у вас установлена СУБД без драйвера jdbc, скачайте его с сайта https://jdbc.postgresql.org/download.html и установите отдельно.

¶ 

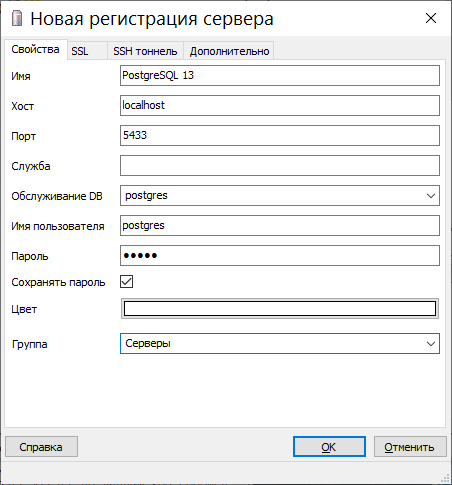
*Рис. 2.1 Главное окно pgAdmin*

После завершения установки запустите pgAdmin, на браузере объектов которого увидите узел установленного сервера БД. Дважды щелкните по этому узлу и введите запрашиваемый пароль. При корректном вводе пароля сервер открывается (визуально в значке узла снимается красный крестик).

Раскройте узел Базы данных (0), и вы увидите базу данных по умолчанию postgres ([Рис. 2.1](#_bookmark19)). На приведенном рисунке на сервере уже установлены десять БД, поэтому узел обозначен надписью Базы данных (2).

Если же на вашем компьютере уже установлена версия PostgreSQL младше 9.5, или вы хотите пользоваться новыми версиями СУБД, то можно поступить следующим образом.

1. После установки такой версии (пусть это будет PostgreSQL версии 13, как показано на скриншоте), скачайте версию pgAdmin 1.27.1b. Например, это можно сделать с сайта https://vvs.ru/pg/.
2. Установите pgAdmin III указанной версии.
3. Запустите pgAdmin III и подключите сервер установленной версии СУБД в следующем порядке:
   1. выберите в строке меню pgAdmin III команду Файл/Добавить сервер …;



*Рис. 2.2 Диалог «Новая регистрация сервера»*

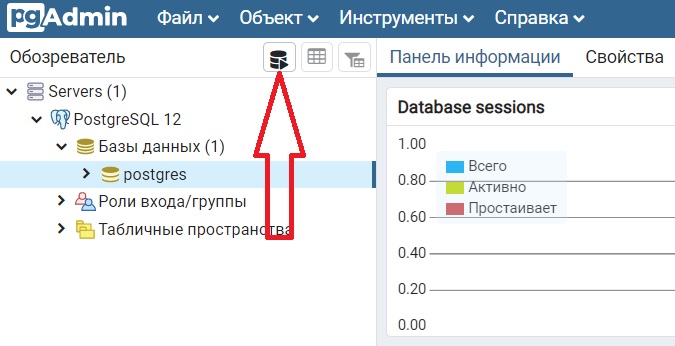
* 1. в диалоге «Новая регистрация сервера» введите данные, как показано на [Рис. 2.2](#_bookmark20), и нажмите кнопку «ОК». На левой панели графического интерфейса pgAdmin III в узле «Серверы ()» добавляется строка с наименованием добавленного сервера СУБД ([Рис. 2.3](#_bookmark21)).

|  |
| --- |
|  |
| Рис. 2.3 Графический интерфейс pgAdmin 4 |

После установки СУБД готова к работе, и первое, что вы долж- ны сделать, – это создать свою БД. Следует иметь в виду, что данная версия графического интерфейса скачана с неофициального сайта. Но в процессе его эксплуатации автором данного учебно- методического пособия ошибок в работе не было выявлено.

### СОЗДАНИЕ ПУСТОЙ БД

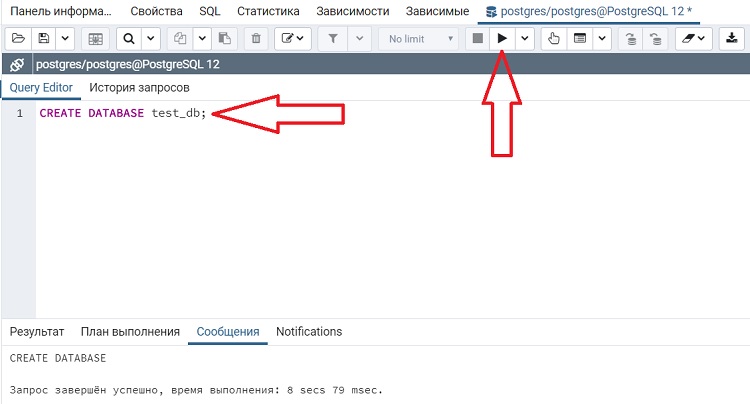
Базу данных можно создать помощью [языка SQL](https://info-comp.ru/sql). Чтобы это сделать, выделите системную базу данных «postgres» и запустите редактор SQL запросов (Запросник), иконка располагается на панели инструментов, или зайдите в меню «Инструменты –> Запросник».



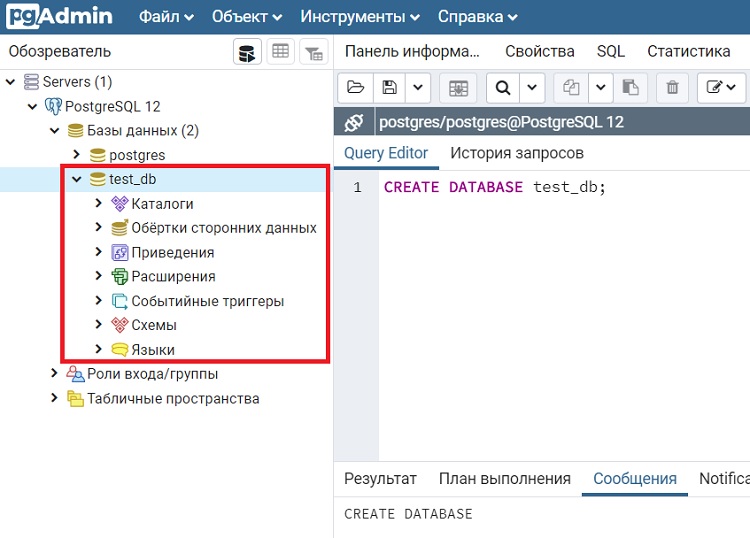
Далее напишите простую SQL инструкцию

CREATE DATABASE test\_db;

И запустите ее на выполнение, для этого нажмите на кнопку «Выполнить» (Execute).

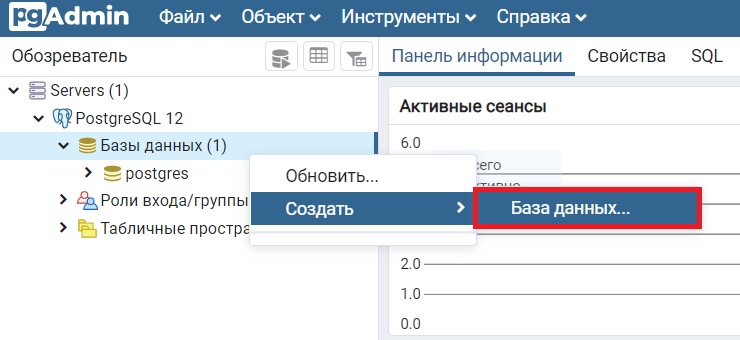


В итоге будет создана база данных с настройками по умолчанию.



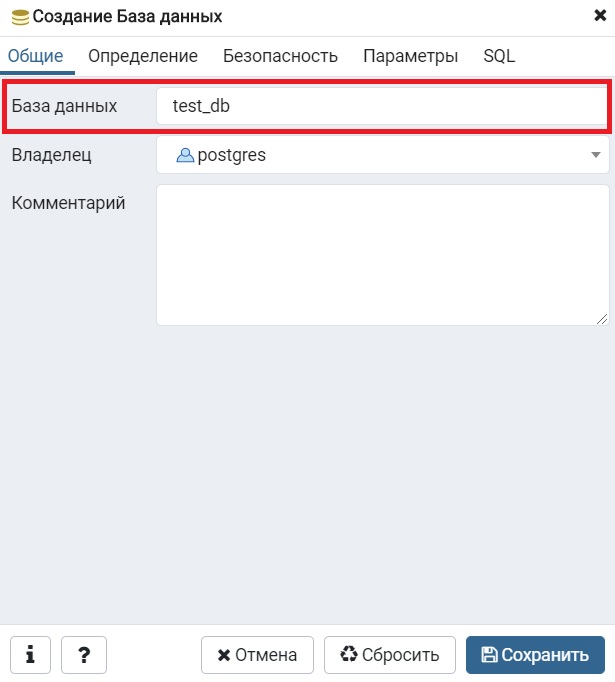
Для создания БД можно воспользоваться также средствами **графического интерфейса pgAdmin**. С этой целью выполните следующие шаги.

Открываем pgAdmin 4, затем в обозревателе щелкаем правой кнопкой мыши по контейнеру «Базы данных». Далее выбираем «Создать -> База данных».

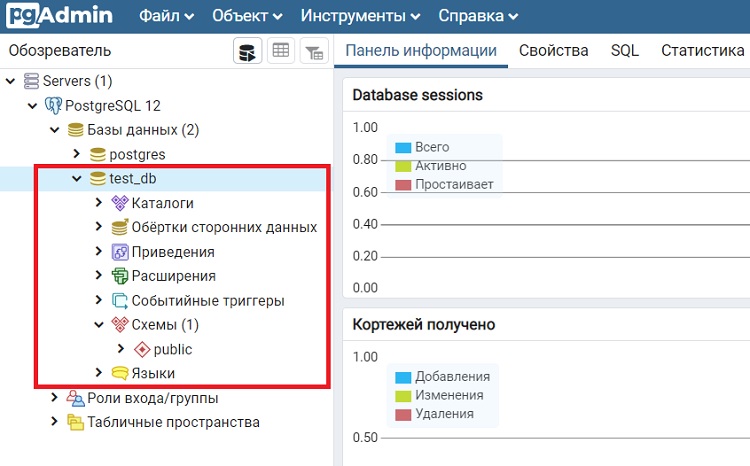


После этого откроется окно «Создание базы данных», где необходимо указать название создаваемой базы данных. И этого достаточно, чтобы создать базу данных с настройками по умолчанию, таким образом, если Вы не хотите внести какие-нибудь дополнительные параметры, можете сразу нажимать «Сохранить».

В данном случае создается база данных test\_db.



В результате в обозревателе отобразится новая база данных.



Обратите внимание на то, что по сравнению с простейшим син- таксисом, приведенным выше, в отображаемом скрипте вы увидите много дополнительных деталей, определяемых СУБД автоматически по настройкам, заданным в ходе ее установки.

На этом создание пустой БД завершается.

### СОЗДАНИЕ СХЕМЫ БД

База данных PostgreSQL может содержать одну или несколько именованных схем, в которых размещаются объекты БД: домены, таблицы, представления, функции и т. д.

Схема БД является удобным средством моделирования данных и позволяет организовывать пространства имен различных разделов ПрО, в которых имена объектов могут совпадать с именами других объектов, существующих в других схемах. Описание схемы БД, а также запроса для ее создания приведено в разделе 2.2 учебного по- собия.

Для создания схемы БД выполните следующие действия.

1. Откройте pgAdmin 4, раскройте узел uchDB/Схемы на левой панели «Браузер объектов», в КМ выберите пункт «Новая схема …» и введите имя схемы ([Рис. 2.5](#_bookmark25)).
2. После этого на левой панели «Браузер объектов» графического интерфейса pgAdmin под узлом «Схемы» появляется дочерний узел с именем Agat.

¶

|  |
| --- |
|  |
| *Рис. 2.5 Окно «Новая схема …»* |

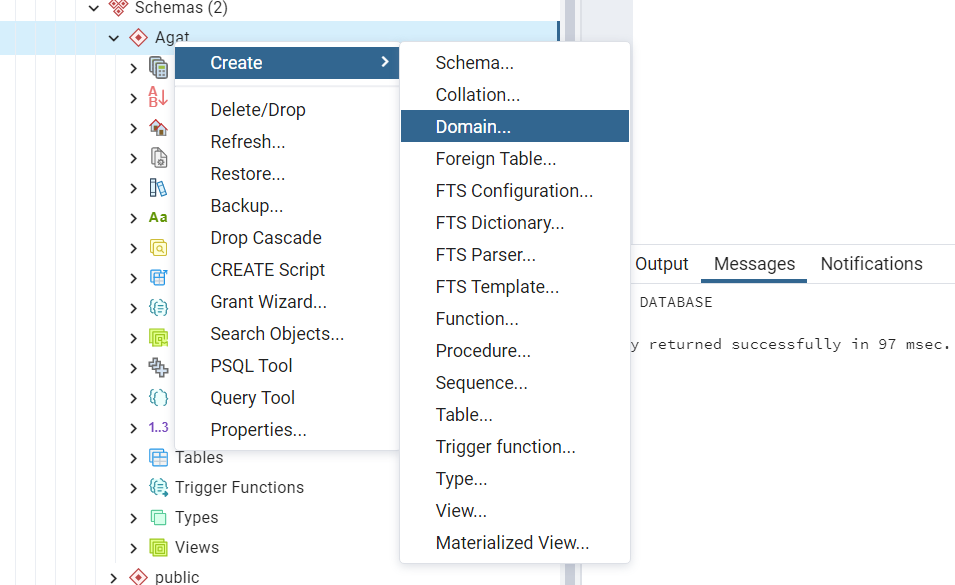
Обратите внимание на то обстоятельство, что в учебной БД создана схема Agat, а также используется схема public, создаваемая по умолчанию.

### СОЗДАНИЕ ДОМЕНОВ

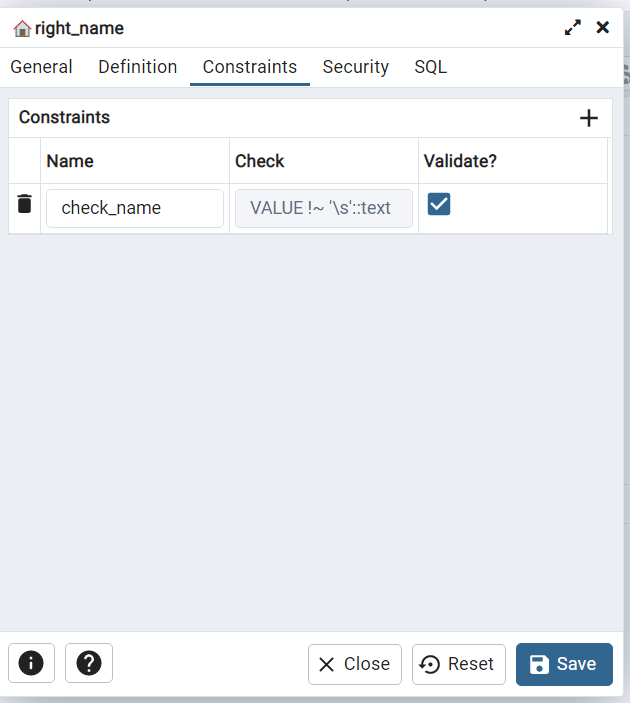
Если в таблице или в нескольких таблицах БД присутствуют столбцы, обладающие одними и теми же характеристиками, можно предварительно описать тип такого столбца с помощью домена, а затем поставить в соответствие каждому из таких столбцов имя домена.

Домены определяются путем введения ограничений на значения базовых встроенных типов, для снижения вероятности ввода ошибочных значений пользователем.

Создание домена выполняется либо в оболочке PGAdmin с последующей настройкой типа данных и ограничений:



Настройка домена:



Либо с помощью команды SQL

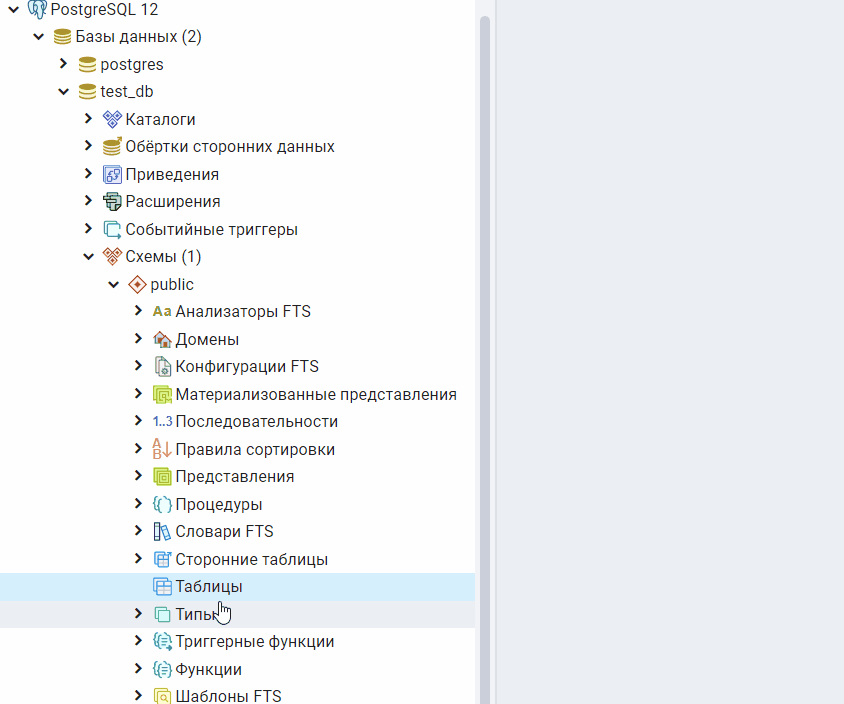
CREATE DOMAIN right\_name AS

TEXT NOT NULL CHECK (value !~ '\s');

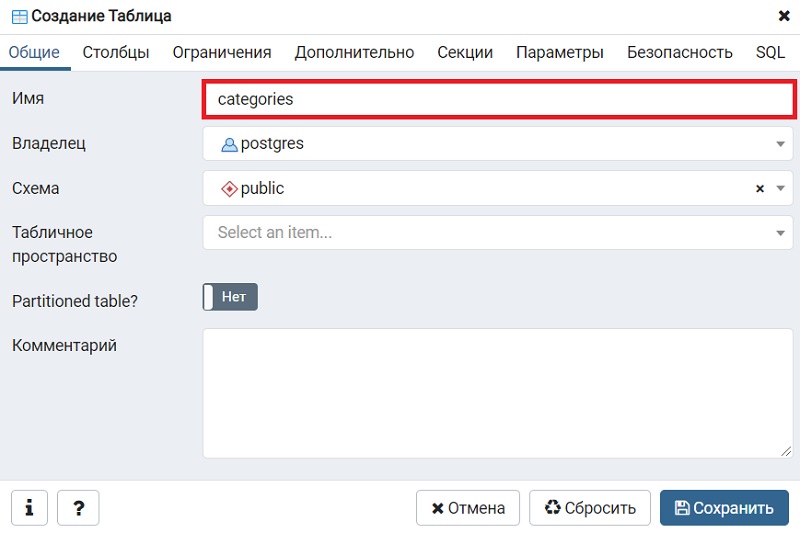
### ¶СОЗДАНИЕ ТАБЛИЦ

При определении таблицы вводятся ее наименование и наиме- нования атрибутов. В качестве идентификатора, т. е. ключевого ат- рибута таблицы, могут быть выбраны один, два или более атрибута. Для создания таблицы базы данных выполните следующие действия.

Чтобы создать таблицу в PostgreSQL с помощью графического конструктора pgAdmin 4, необходимо в обозревателе щёлкнуть правой кнопкой мыши по контейнеру «Таблицы» и выбрать пункт «Создать -> Таблицу».



Затем запустится графический конструктор таблиц, где первым делом нам необходимо ввести название таблицы, сначала давайте создадим таблицу с категориями, чтобы потом в процессе создания таблицы с товарами у нас была возможность сразу определить ограничение внешнего ключа. Поэтому вводим categories.



В случае необходимости Вы можете изменить владельца таблицы и схему, в которой будет располагаться таблица.

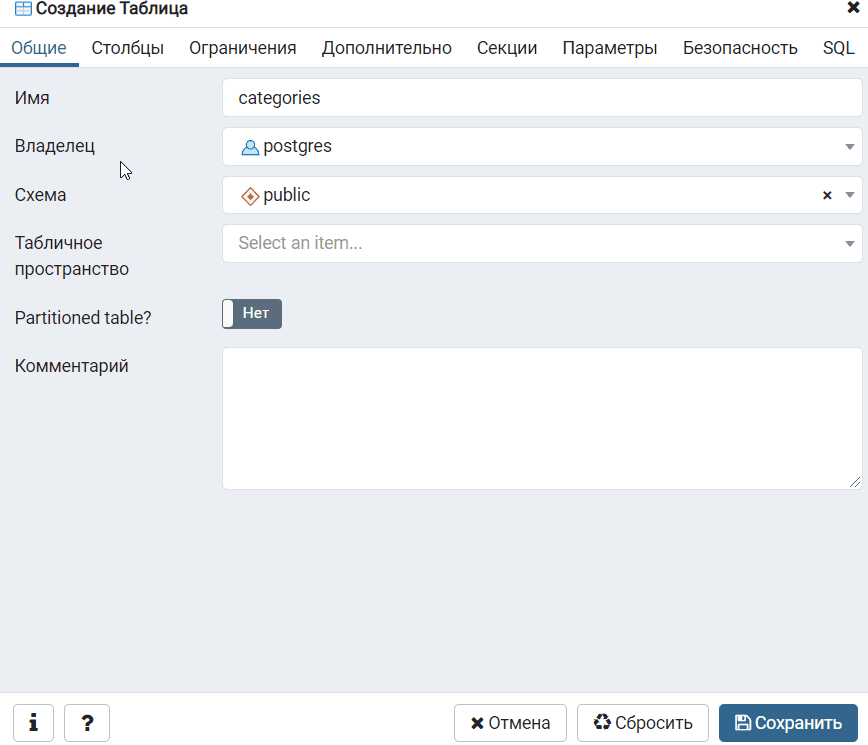
**Схема** – это пространство имен в базе данных, своего рода контейнер объектов внутри базы данных, который позволяет логически разделять данные на схемы.

После того как название таблицы задано, мы можем переходить к определению столбцов. Для этого необходимо перейти на вкладку «Столбцы» и, используя кнопку плюс «+», добавить нужные столбцы.

Сначала создаем идентификатор категории, по условиям нашей задачи он должен отвечать определенным требованиям, поэтому мы должны задать следующие свойства у столбца:

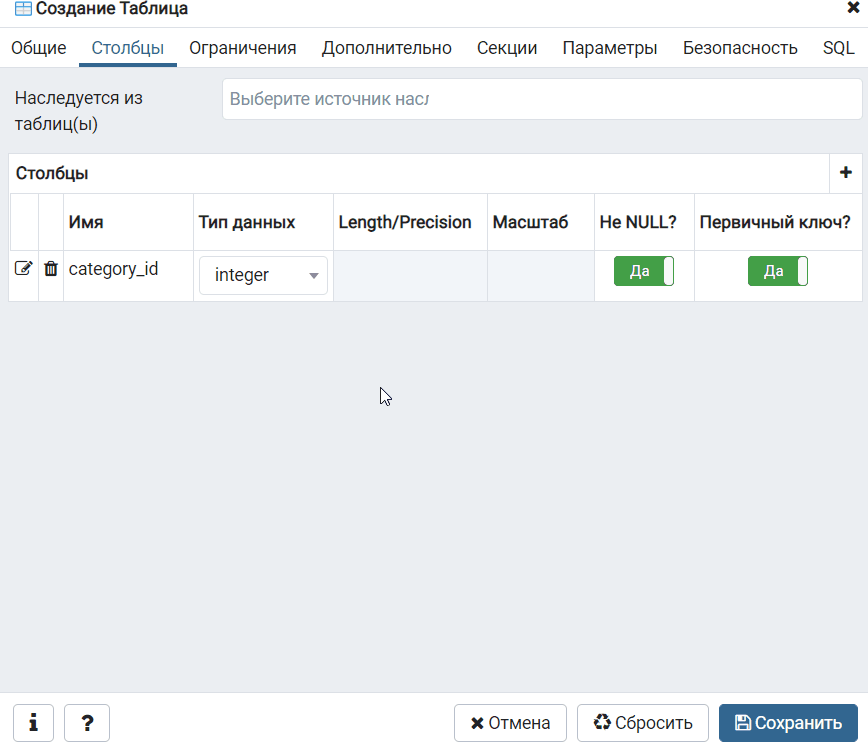
* Включить параметр «Не NULL», чтобы параметр не мог хранить значения NULL;
* Включить параметр «Первичный ключ», чтобы столбец выполнял роль первичного ключа;
* Включить идентификацию, чтобы в столбце автоматически генерировались значения.

В качестве типа данных выберем целочисленный тип integer.

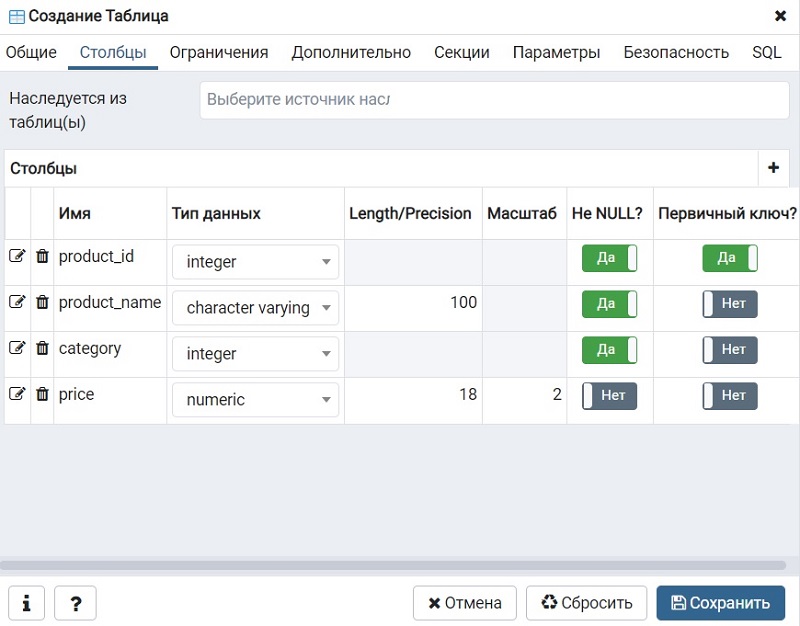


Далее точно так же добавляем столбец для хранения наименования категории. При этом тип данных у нас уже должен быть текстовый, например, character varying (VARCHAR) с длинной 100. Он уже не должен быть первичным ключом и значения генерировать здесь не нужно.

После ввода всех данных мы можем сохранить все изменения, нажав на кнопку *«Сохранить»*, тем самым создав таблицу.

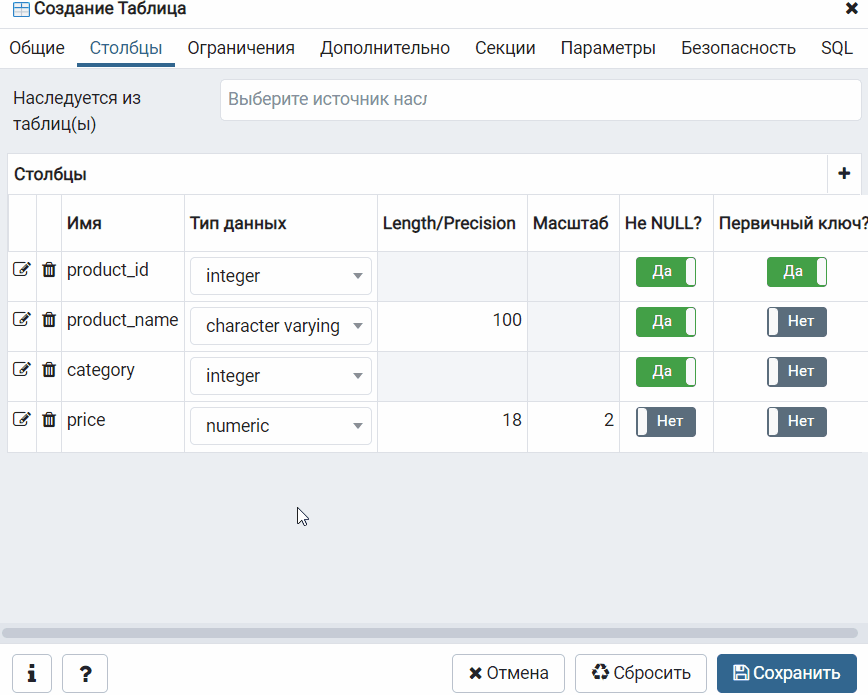


Затем точно также создаем таблицу с товарами и определяем соответствующие для нее столбцы. Столбец идентификатора товара определяем, как первичный ключ, и включаем у него идентификацию с автоматическим генерированием значений.



После того как столбцы определены, нам необходимо добавить значение по умолчанию для столбца category, а также определить ограничение внешнего ключа.

Чтобы у столбца задать значение по умолчанию, необходимо открыть детализированные свойства столбца, перейти там на вкладку «Ограничения» и в поле «По умолчанию» указать значение, которое будет присваиваться по умолчанию, например, 1.

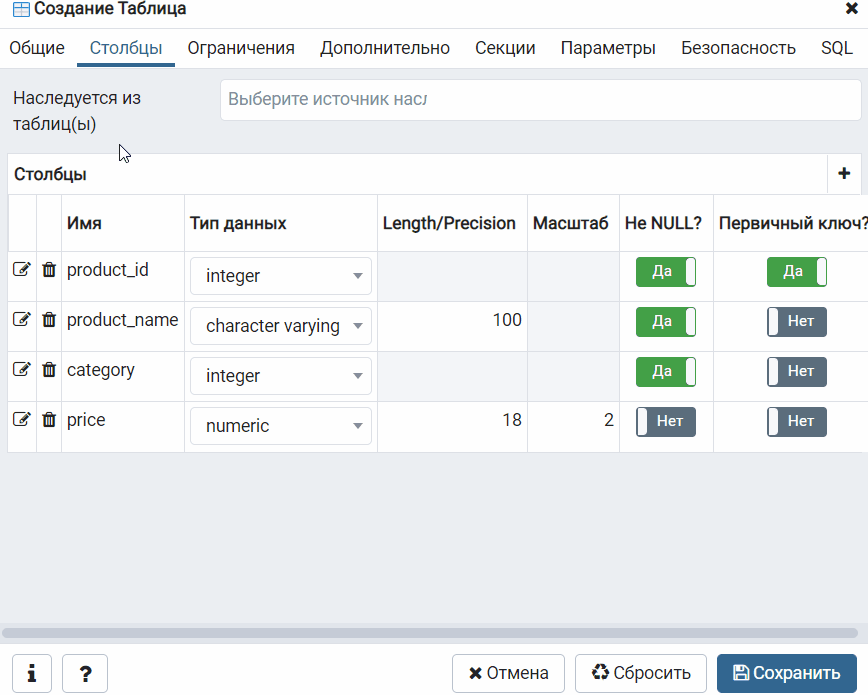


Осталось определить ограничение внешнего ключа, иными словами, чтобы столбец category таблицы goods ссылался на столбец category\_id таблицы categories, таким образом, мы определим связь между этими таблицами.

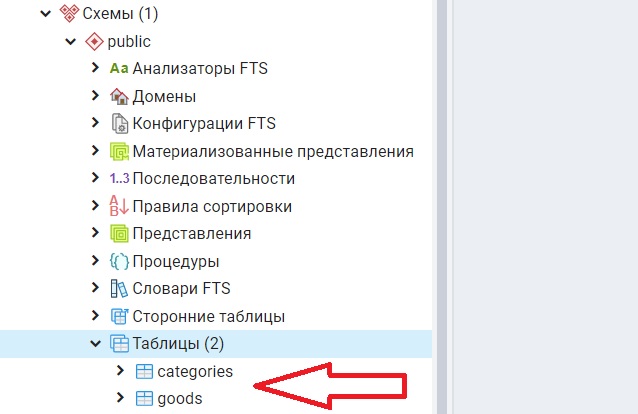
Чтобы это сделать, переходим на вкладку «Ограничения» (основной формы создания таблиц), затем переходим на вкладку «Внешний ключ» и с помощью кнопки плюс «+» добавляем новый внешний ключ.

Вводим название ограничения, и в детализированных свойствах на вкладке «Столбцы» задаем связь между таблицами, т.е. указываем столбцы и нажимаем на плюс «+» для добавления связи.

После этого все требования, указанные в нашей задаче, будут выполнены и мы можем нажать кнопку «Сохранить» для создания таблицы.



Созданные таблицы отобразятся в обозревателе.



### Создание таблицы с помощью языка SQL

Теперь давайте рассмотрим процесс создания таблиц в PostgreSQL на языке SQL.

Таблицы в SQL создаются с помощью инструкции CREATE TABLE.

Для создания точно таких же таблиц и решения нашей задачи мы можем использовать следующие инструкции языка SQL.

**Примечание!** Если чуть ранее Вы создавали эти таблицы с помощью конструктора, то перед выполнением указанных ниже инструкций создания таблиц необходимо предварительно удалить эти таблицы, т.е. выполнить инструкцию DROP TABLE. В противном случае инструкции создания таблиц завершатся ошибкой, так как в пределах одной схемы не может быть таблиц с одинаковым названием.

Чтобы выполнить инструкции, открываем редактор запросов (Запросник), вводим инструкции и нажимаем «Execute».

-- Удаление таблиц

DROP TABLE IF EXISTS goods, categories;

-- Создание таблицы categories

CREATE TABLE categories (

category\_id INT NOT NULL GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

category\_name VARCHAR(100) NOT NULL

);

-- Создание таблицы goods

CREATE TABLE goods (

product\_id INT NOT NULL GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

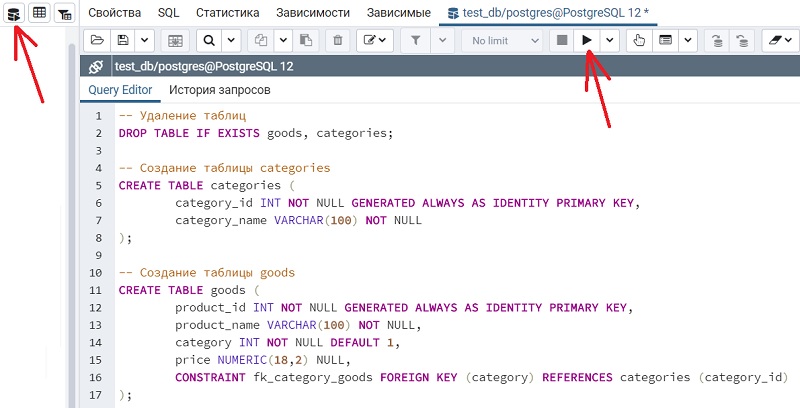
product\_name VARCHAR(100) NOT NULL,

category INT NOT NULL DEFAULT 1,

price NUMERIC(18,2) NULL,

CONSTRAINT fk\_category\_goods FOREIGN KEY (category) REFERENCES categories (category\_id)

);



После установки связи, не выходя из диалога, необходимо опре- делить для нее действия СУБД по сохранению ссылочной целост- ности. Под ссылочной целостностью понимается обеспечение тре- бования, согласно которому после выполнения операций обновле- ния и удаления значения внешнего ключа экземпляра дочерней сущности соответствовали одному из значений первичного ключа родительской сущности.

* + 1. ¶ОПРЕДЕЛЕНИЕ ПРАВИЛ СОХРАНЕНИЯ ССЫЛОЧНОЙ ЦЕЛОСТНОСТИ

Нарушение ссылочной целостности может иметь место при вы- полнении операций обновления и удаления значений ключевых ат- рибутов связи. Для каждой из перечисленных операций можно ус- тановить следующие значения, определяющие правила сохранения ссылочной целостности:

* *No Action* – отсутствие действий СУБД;
* *Restrict* – запрет операции;
* *Cascade* – каскадное выполнение операции;
* *Set null* – установка пустого значения;
* *Set default* – установка заданного значения по умолчанию.

Для установки одного из этих правил для операций обновления и удаления значений ключевых атрибутов связи в окне «*Новый внешний ключ …*» перейдите на вкладку «*Действие*» ([Рис. 2.13](#_bookmark37)). В зависимости от характера связи установите одно из перечислен- ных значений для каждой операции изменения значений первично- го ключа.

|  |
| --- |
|  |
| *Рис. 2.13 Вкладка «Действие» окна «Новый внешний ключ …»* |

В качестве примера опишем анализ ситуации, приводящей к удалению записи родительской таблицы для связи *Magazin* 

¶*Sotrudnik*. Удаление записи из таблицы *Magazin* может быть осу- ществлено при ликвидации магазина в связи с реорганизацией ком- пании. В БД после этого оказывается, что сотрудники данного ма- газина не работают ни в одном из существующих магазинов.

Это происходит по той причине, что вторичные ключи соответ- ствующих записей в таблице *Sotrudnik* ссылаются на несущест- вующую (уже удаленную) запись в таблице *Magazin*. Другими сло- вами, происходит нарушение ссылочной целостности, и это нару- шение выявляется СУБД автоматически благодаря наличию соот- ветствующих триггерных функций. Наша задача заключается в оп- ределении реакции СУБД, направленной на исправление данной ошибочной ситуации. Это действие должно быть определено с уче- том стратегии компании в области работы с кадрами.

1. Если согласно политике компании сотрудники ликвидиро- ванных магазинов увольняются, то и записи об уволенных сотруд- никах удаляются из таблицы *Sotrudnik*. Для этого в окне «*Новый внешний ключ …*» выбирается правило *CASCADE*.
2. Допустим, что в компании принято не увольнять сотрудников ликвидируемых магазинов, а распределять их по оставшимся мага- зинам компании. Для этого в окне «*Новый внешний ключ …*» выби- рается правило *SET NULL*.

В данном случае процесс может быть растянут по времени (до- пустим в течение недели) и реализован с участием персонала ин- формационной системы по следующему алгоритму.

В соответствии с правилом *SET NULL* в таблице *Sotrudnik* для записей, имеющих некоторое заданное значение *id\_mgz = n*, уста- навливается псевдозначение *id\_mgz = NULL*. После этого по мере на- значения сотрудников в другие магазины псевдозначения *NULL* за- меняются на значения атрибута вторичного ключа, указывающие на соответствующие магазины.

Для реализации описанного алгоритма в качестве правила сохранения ссылочной целостности определено правило *Set null* ([Рис. 2.13](#_bookmark37)), согласно которому выполняются следующие действия:

* ¶после выполнения операции удаления родительской записи в таблице *Magazin*, для дочерних записей в таблице *Sotrudnik* значе- ния вторичных ключей устанавливаются псевдозначение *NULL*;
* для вторичных ключей соответствующих записей таблицы *Sotrudnik* устанавливаются значения, равные значениям первич- ных ключей записей тех магазинов, в которые распределяются со- трудники ликвидированного магазина. Эта установка производится персоналом «вручную» по ходу распределения сотрудников по дру- гим магазинам.

В результате установки правила сохранения ссылочной цело- стности выражение запроса определения таблицы *Sotrudnik* долж- но видоизмениться. Для просмотра этих изменений на панели

«*Браузер объектов*» щелкните по узлу таблицы *Sotrudnik* и на панели «*Панель SQL*» изучите модифицированное выражение за- проса в части определения вторичного ключа.

После установления правила сохранения ссылочной целостно- сти *SET NULL* для связи *Magazin*  *Sotrudnik* выражение запроса *CREATE TABLE* для таблицы *Sotrudnik* приобретает вид:

CREATE TABLE "Agat"."Sotrudnik" id\_str integer NOT NULL, family\_str character varying, name\_str character varying, dolzhn character varying,

age integer, id\_mgz integer,

CONSTRAINT sotrudnik\_pkey PRIMARY KEY (id\_str), CONSTRAINT "Sotrudnik\_id\_mgz\_fkey"

FOREIGN KEY (id\_mgz)

REFERENCES "Agat"."Magazin" (id\_mgz) MATCH SIMPLE

ON UPDATE SET NULL ON DELETE SET NULL

)

По описанной схеме определите связи вашей схемы.

### ¶СОДЕРЖАНИЕ ОТЧЕТА

После выполнения предписанных в данном разделе действий составьте отчет, содержащий следующие разделы.

Цель выполнения работы.

Описание содержания выполненных работ.

Краткое описание СУБД PostgreSQL и графического интер- фейса pgAdmin III.

Создание пустой БД (в отчете привести выражение запроса на создание БД и определение БД).

Создание схемы БД (в отчете привести выражение запроса на создание схемы и определение схемы).

Создание доменов (в отчете привести выражения запросов на создание не менее 3 доменов БД и определение домена).

Создание таблиц (в отчете привести запросы на создание БД и быть готовым описать пример выражения запроса CREATE TABLE при сдаче отчета преподавателю).

Установка связей между таблицами (в отчете привести мо- дифицированные запросы CREATE TABLE на создание дочерних таблиц БД вашего варианта после установки внешних ключей свя- зей).

Определение правил ссылочной целостности связи (в отчете привести модифицированные запросы CREATE TABLE на создание дочерних таблиц БД вашего варианта после установки правил со- хранения ссылочной целостности).

Выводы.

## ЛАБОРАТОРНАЯ РАБОТА №3. МОДИФИКАЦИЯ БД И ВЫПОЛНЕНИЕ МАНИПУЛЯЦИЙ С ДАННЫМИ БАЗЫ

Цель работы – освоение навыков по созданию дополнительных объектов в БД и заполнению созданной базы данными, а также вы- полнение простейших манипуляций с этими данными.

После изучения перечисленного материала выполните следую- щие модификации с БД и операции:

* создания индексов для атрибутов, на основе которых (по вашему усмотрению) целесообразно проводить поисковые операции;
* создания последовательностей на основе первичных ключей;
* заполнения базы данными с помощью оператора INSERT;
* обновления строк таблиц БД с помощью оператора UPDATE;
* удаления строк таблиц БД с помощью оператора DELETE;
* создания представлений с помощью оператора CREATE VIEW.

### СОЗДАНИЕ ИНДЕКСОВ

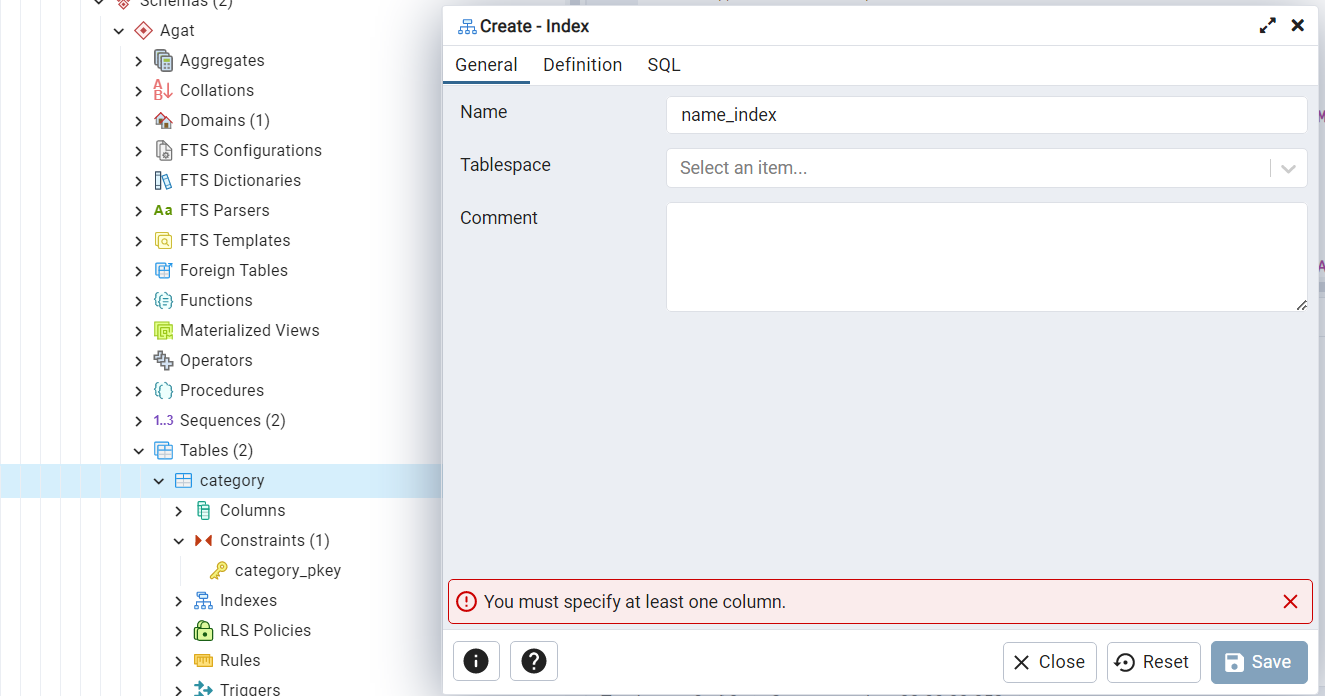
Индексы в *PostgreSQL* можно определить двумя способами:

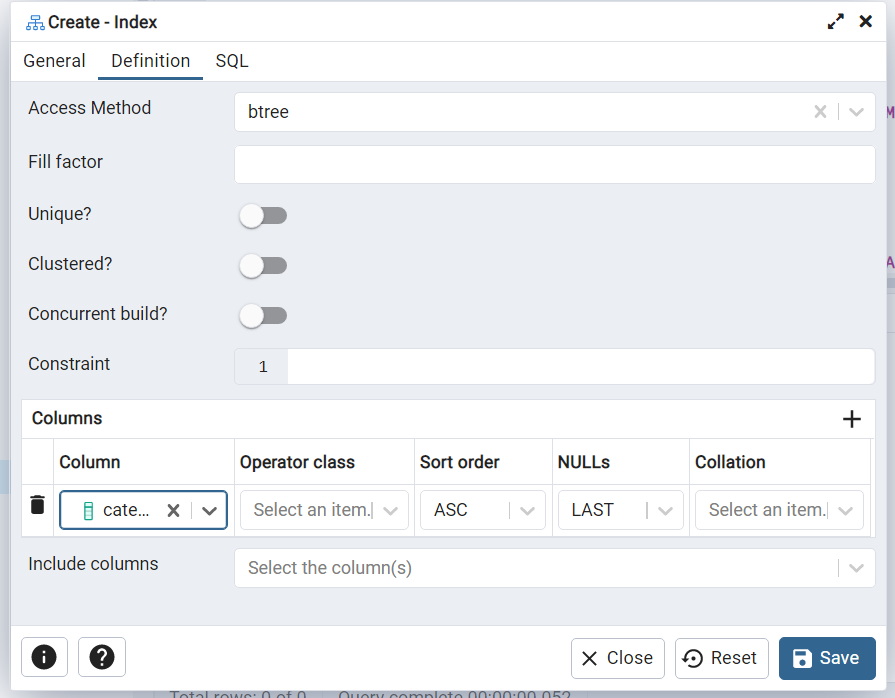
* набором выражения запроса *CREATE INDEX* (см. раздел 4.7) в редакторе запросов *Query*, и его выполнения путем нажатия кноп- ки «*Выполнить пользовательские SQL-запросы*» на панели ин- струментов в *pgAdmin*;
* формированием выражения запроса создания индекса БД в диалоге «*Новый индекс* …» клиентского приложения *pgAdmin*.

|  |
| --- |
| Создание индекса с помощью редактора запросов *Query* описывается просто и заключается в наборе текста запроса на панели «*Редактор SQL*» и нажатии кнопки «*Выполнить запрос*», как по- казано на [Рис. 2.14](#_bookmark41). |
| *Рис. 2.14 Окно «Query» с запросом CREATE INDEX* |

Создание индекса по второму способу осуществляется в следующем порядке.

1. Вызовите диалог «*Новый индекс* …» путем выбора пункта «*Новый объект/Новый индекс …*» КМ для назначенной таблицы БД ([Рис. 2.15](#_bookmark42)).





1. В вызванном диалоге на вкладке «Свойства» определите имя индекса (например, ind\_family\_name\_str).
2. В данном примере индекс создается на основе атрибутов family\_str и name\_str. Для этого перейдите на вкладку «Колонки» и в выпадающем списке «Колонка» выберите первый атрибут вы- бранной таблицы, нажмите кнопку «Добавить». Затем в выпадающем списке «Колонка» выберите второй атрибут выбранной таблицы и снова нажмите кнопку «Добавить».
3. Далее после нажатия кнопки «ОК» на панели «Браузер объектов» в узле «Индексы» для таблицы «Sotrudnik» появится пункт с названием индекса.

### СОЗДАНИЕ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

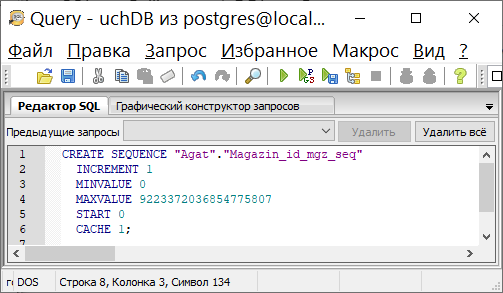
Перед выполнением данной работы изучите раздел 4.8 учебного пособия [1]. Определите не менее трех ключевых атрибутов таблиц вашей БД, для которых будут созданы последовательности.

Последовательности в *PostgreSQL* можно определить разными способами:

* набором выражения запроса *CREATE SEQUENCE* (см. раздел 4.8) в редакторе запросов *Query* и его выполнения путем нажатия кнопки «*Выполнить пользовательские SQL-запросы*» на панели инструментов в *pgAdmin*;
* введением набора параметров для формирования выражения запроса создания последовательности БД с помощью графического интерфейса *pgAdmin*.

1. Создание последовательности с помощью редактора запросов *Query* заключается в наборе текста запроса на панели «*Редактор SQL*» и нажатии кнопки «*Выполнить запрос*» ([Рис. 2.16](#_bookmark44)).
2. Создание последовательности по второму способу осуществ- ляется в таком порядке:

* вызовите диалог «*Новая последовательность* …» путем выбора пункта «*Новый объект/Новая последовательность …*» КМ для назначенной вами таблицы (например, *Magazin*) БД;

¶

*Рис. 2.16 Окно «Query» с введенным выражением запроса CREATE SEQUENCE*

* в вызванном диалоге в поле «*Имя*» вкладки «*Свойства*» оп- ределите имя последовательности (*Magazin\_id\_mgz\_seq* – последо- вательность для ключевого атрибута *id\_mgz* таблицы *Magazin*) и перейдите на вкладку «*Определение*». В полях этой вкладки набе- рите значения полей как показано на [Рис. 2.17](#_bookmark45);
* после нажатия кнопки «*ОК*» в узле «*Последовательности*» схемы появится пункт с именем последовательности.

Создайте все намеченные вами последовательности одним из описанных выше методом.

Созданные последовательности необходимо связать с соответст- вующими идентификаторами или атрибутами первичных ключей. Это можно сделать при создании таблицы, а также для уже создан- ной таблицы.

1. Связывание последовательности с идентификатором при соз- дании таблицы реализуется следующим образом:

* вызовите диалог «*Новая таблица* …»;
* после введения имени таблицы перейдите на вкладку «*Колон- ки*» и нажмите на кнопку «*Добавить*»;

¶

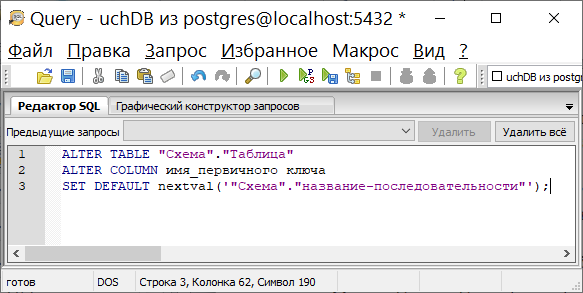
|  |
| --- |
|  |
| *Рис. 2.17 Окно «Новая последовательность* …*»* |

* в окне «*Новая колонка*» в поле «*Имя*» введите название ат- рибута идентификатора, а в выпадающем списке «*Тип данных*» выберите «*integer*»;
* перейдите на вкладку «*Определение*» и в поле «*Значение по умолчанию*» введите текст «*nextval*('«*Схема*».«*Имя последователь- ности*»'::*regclass*)»;
* далее создайте другие атрибуты таблицы.

1. Для связывания идентификатора уже существующей таблицы с последовательностью наберите в редакторе «*Query*» выражение запроса, приведенное на [Рис. 2.18](#_bookmark46).

Существует третий, более практичный способ создания иден- тификатора таблицы с автоматически увеличивающимся значени- ем. Этот способ заключается в присвоении идентификатору при создании таблицы типа данных «*serial*». Преимущества данного способа создания последовательности заключаются в следующем:

* во-первых, при этом последовательность с параметрами по умолчанию создается автоматически. В этом можно убедиться, вы- брав на панели «*Браузер объектов*» раздел «*Последовательности*» и нажав на кнопку «*Обновить выбранный объект*» на пане ли инструментов;
* во-вторых, в этом случае созданная последовательность автоматически связывается с идентификатором.



*Рис. 2.18 Окно «Query»*

### ЗАПОЛНЕНИЕ ТАБЛИЦ БД С ПОМОЩЬЮ ОПЕРАТОРА INSERT

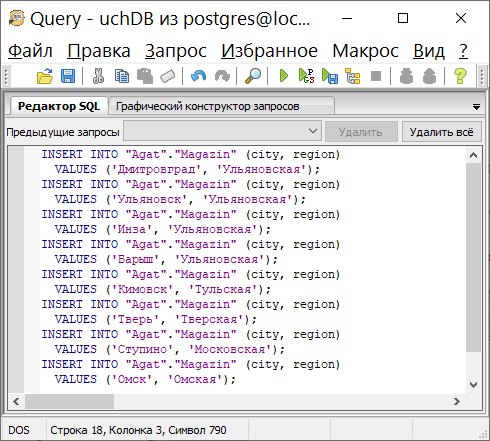
Для выполнения операций манипуляции в созданной базе данных необходимо заполнить ее таблицы данными, что реализуется путем введения в них записей или строк.

* набором выражений запросов INSERT в редакторе запросов Query, вызываемого путем нажатия кнопки «Выполнить пользовательские SQL-запросы» на панели инструментов в pgAdmin;
* формирования выражения запросов на ввод записей БД с помощью графического интерфейса pgAdmin.
* Ввод записей с помощью редактора запросов Query заключается в наборе выражений запросов на панели «Редактор SQL» и нажатии кнопки «Выполнить запрос» ([Рис. 2.19](#_bookmark48)).

insert into "Agat"."category" (category\_name) values ('Первая категория')

insert into "Agat"."product" (product\_name, category, price)

values ('Третий товар', 2, 22.3);



*Рис. 2.19 Окно «Query» с введенными запросами INSERT*

Обратите внимание на то обстоятельство, что в перечне атрибутов таблицы Magazin отсутствует ключевой атрибут id\_mgz. Это объясняется тем, что данный ключевой атрибут определен в качестве последовательности и в «ручном» вводе его значений нет необходимости, т. к. они вычисляются и вставляются в таблицу с по- мощью СУБД автоматически.

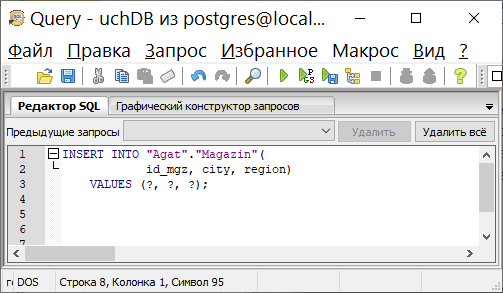
Для просмотра заполненной таблицы выделите узел таблицы на дереве объектов и на панели инструментов pgAdmin III щелкните по кнопке  («Просмотр данных в выбранном объекте»).

Ввод записей в таблицы по второму способу осуществляется в следующем порядке.

1. ¶Вызовите редактор запросов *Query* путем выбора пункта

«*Скрипты/Скрипт INSERT*» КМ для выбранной вами таблицы (в нашем случае – это таблица *Magazin*).

1. В вызванном редакторе на вкладке «*Редактор SQL*» вы об- наружите шаблон выражения запроса, в котором требуется заме- нить знаки '*?* ' значениями соответствующих атрибутов записи за- полняемой таблицы ([Рис. 2.20](#_bookmark49)).



*Рис. 2.20 Окно «Query» с шаблоном запроса INSERT*

В данном примере шаблон запроса вставки записи сформирован для таблицы *category*, первичный ключ (*category\_id*) которой определен в виде последовательности. В подобных случаях предлагается скор ректировать выражение запроса путем удаления наименования атрибута последовательности (*id\_mgz*) и соответствующего знака '*?* '.

В редакторе запросов *Query* можно продублировать шаблоны в требуемом количестве и после замены знаков '*?* ' значениями соот- ветствующих атрибутов получить множество запросов *INSERT* для заполнения всей таблицы, которое показано на [Рис. 2.19](#_bookmark48).

Выбрав один из способов заполнения таблиц, введите данные во все таблицы базы, чтобы в ней можно было выполнить осталь- ные операторы манипуляции данными.

* + 1. ¶ОБНОВЛЕНИЕ ТАБЛИЦ С ПОМОЩЬЮ ОПЕРАТОРА UPDATE

Как и в случае выполнения запросов *INSERT*, обновление таблиц можно выполнить двумя способами:

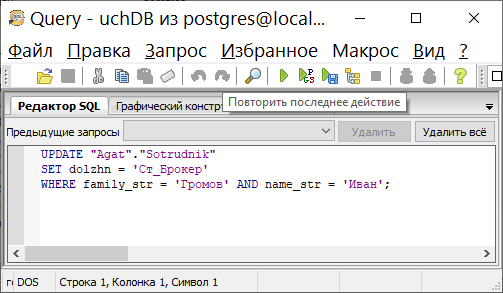
* набором выражений запросов обновления записей (см. под- раздел 5.6.1) в редакторе запросов *Query*, вызываемого путем нажа- тия кнопки «*Выполнить пользовательские SQL-запросы*» на па- нели инструментов в *pgAdmin*;
* набором выражения запроса обновления записей БД с помо- щью графического интерфейса *pgAdmin*.

Обновление записей с помощью редактора запросов *Query* вы- полняется путем набора выражений запросов на панели «*Редактор SQL*» и нажатии кнопки «*Выполнить запрос*» ([Рис. 2.21](#_bookmark51)).

update "Agat"."product"

set price=0.0

where category=2



*Рис. 2.21 Окно «Query» с введенным запросом UPDATE*

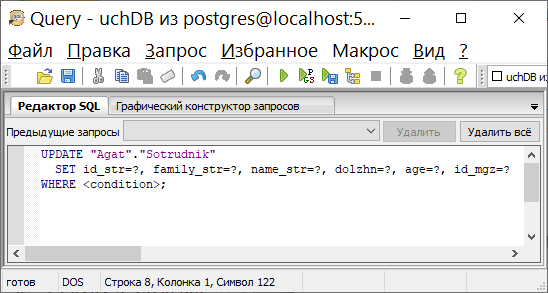
Для просмотра обновленной таблицы базы данных щелкните по кнопке  («*Просмотр данных в выбранном объекте*») пане- ли инструментов *pgAdmin III*.

Обновление записей в таблицы по второму способу осуществ- ляется в следующем порядке.

1. ¶Вызовите редактор запросов *Query* путем выбора пункта

«*Скрипты/Скрипт UPDATE*» КМ для выбранной вами таблицы (в нашем случае – это таблица *Sotrudnik*).

1. В вызванном редакторе на вкладке «*Редактор SQL*» вы об- наружите шаблон выражения запроса ([Рис. 2.22](#_bookmark52)), в котором тре- буется:

* в предложении *SET* оставить обновляемый атрибут (-ы) и за- менить знак (-и) ' *?* ' значением (-ями) соответствующего (-их) атри- бута (-ов) записи заполняемой таблицы;
* в предложении *WHERE* вставить условие выборки записи (в нашем примере, *family\_str = 'Громов' AND name\_str = 'Иван'*).

*Рис. 2.22 Окно «Query» с шаблоном запроса UPDATE*

Выбрав один из способов заполнения таблиц, в отчете по рабо- те продемонстрируйте три примера выполнения запроса на обнов- ление записей вашей БД.

* + 1. УДАЛЕНИЕ СТРОК ТАБЛИЦ С ПОМОЩЬЮ ОПЕРАТОРА

DELETE

Перед выполнением данной работы изучите подраздел 5.6.3 учебного пособия [1]. Как и в случае выполнения запросов *INSERT*, удаление строк таблиц можно выполнить двумя способами:

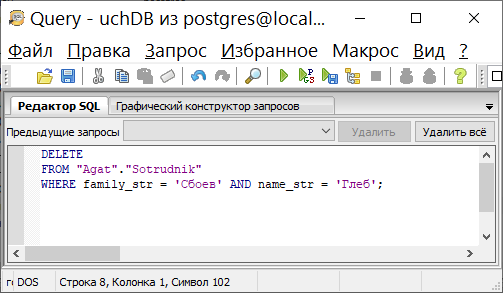
* ¶набором выражений запросов удаления записей (см. подраз- дел 5.6.3) в редакторе запросов *Query*, вызываемого путем нажатия кнопки «*Выполнить пользовательские SQL-запросы*» на панели инструментов в *pgAdmin*;
* набором выражения запроса удаления записей БД с помощью графического интерфейса *pgAdmin*.

Удаление записей с помощью редактора запросов *Query* вы- полняется путем набора выражений запросов на панели «*Редактор SQL*» и нажатии кнопки «*Выполнить запрос*» ([Рис. 2.23](#_bookmark54)).

delete

from "Agat"."category"

where category\_id=3



*Рис. 2.23 Окно «Query» с введенным запросом DELETE*

Для просмотра таблицы базы данных, из которой удалена стро- ка, щелкните по кнопке («*Просмотр данных в выбранном объек- те*») панели инструментов *pgAdmin III*.

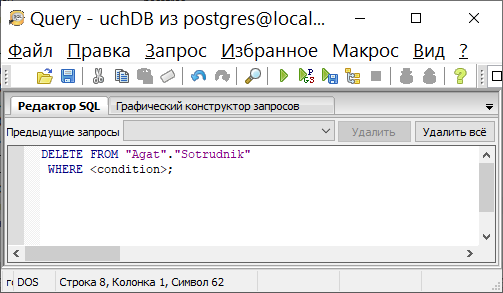
Удаление записей в таблицы по второму способу осуществляет- ся в следующем порядке.

1. Вызовите редактор запросов *Query* путем выбора пункта

«*Скрипты/Скрипт DELETE*» КМ для выбранной вами таблицы (в нашем случае – это таблица *Sotrudnik*).

1. В вызванном редакторе на вкладке «*Редактор SQL*» вы об- наружите шаблон выражения запроса ([Рис. 2.24](#_bookmark55)), в котором требу-

¶ется в предложении *WHERE* вставить условие выборки записи (в нашем примере, *family\_str = 'Сбоев' AND name\_str = 'Глеб'*).



*Рис. 2.24 Окно «Query» с шаблоном запроса DELETE*

Выбрав один из способов заполнения таблиц, в отчете по рабо- те продемонстрируйте три примера выполнения запроса на обнов- ление записей вашей БД.

* + 1. СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ

Представления, создаваемые с помощью оператора *SELECT*, воспринимаются пользователями как обычные таблицы БД. Но в БД этой таблицы не существует, просто при его вызове выполняется оператор *SELECT*, который формирует набор данных, извлекаемый из разных таблиц или из части одной таблицы, и представляются пользователю как данные таблицы.

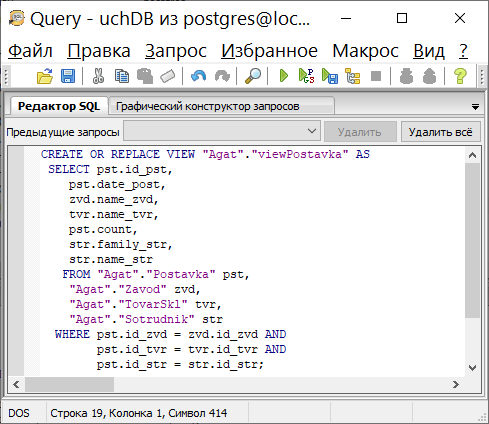
Как и в предыдущих случаях формирования запросов манипуляции данными, создание представлений таблиц также можно выполнить двумя способами:

* набором выражений запросов на создание представлений (см. подраздел 4.6) в редакторе запросов *Query*, вызываемого путем нажатия кнопки «*Выполнить пользовательские SQL-запросы*» на панели инструментов в *pgAdmin*;
* набором выражения запроса удаления записей БД с помощью графического интерфейса *pgAdmin*. Вызов окна «*Новое представ- ление* …» осуществляется путем выбора пункта «*Представления/ Новое представление* …» КМ.

Оба способа предполагают ручной набор выражений запросов на создание представлений, поэтому предлагается воспользоваться для этой цели привычным редактором запросов *Query*.

Пример формирования запроса на создание представления с помощью редактора запросов *Query* приведен на [Рис. 2.25](#_bookmark57).

Создайте не менее двух представлений для вашего варианта с помощью редактора запросов *Query*.



*Рис. 2.25 Запрос, создающий представление viewPostavka*

SELECT category.category\_id,

category.category\_name,

product.product\_id,

product.product\_name,

product.category,

product.price

FROM "Agat".category,

"Agat".product

WHERE product.category = category.category\_id;

¶Все запросы, созданные в ходе выполнения данной лаборатор- ной работы, сохраните в соответствующих файлах. Сохранение вы- ражений запросов в файлах осуществляется следующим образом.

1. Условием для сохранения выражения запроса в файле это вы- ражение должно быть набрано в редакторе запросов *Query*.
2. Затем путем выполнения команды *Файл*/*Сохранить как*/*Зап- рос(текст)* вызовите диалог «*Сохранить файл запросов как*» и с ее помощью определите каталог и имя файла.

По результатам выполнения лабораторных работ при предъяв- лении отчетов преподавателю вы должны быть в готовыми наби- рать выражения запросов «вручную», поэтому внимательно изу- чайте выражения запросов, даже если при выполнении работ вы пользовались графическим интерфейсом *pgAdmin III*.

* + 1. СОДЕРЖАНИЕ ОТЧЕТА

После выполнения предписанных в данном разделе действий составьте отчет, содержащий следующие разделы.

1. Цель выполнения работы
2. Описание содержания выполненных работ.
   1. Создание индексов (в отчете привести запросы на создание не менее 3 индексов).
   2. Создание последовательностей (в отчете привести запросы на создание не менее 3 последовательностей)
   3. Заполнение таблиц БД с помощью операторов *INSERT* (в от- чете и созданной БД должно содержаться не менее 5 записей в каж- дой таблице).
   4. Обновление строк таблиц с помощью оператора *UPDATE* (в отчете привести не менее 3 примеров запросов на обновление таб- лиц).
   5. Удаление строк таблиц с помощью оператора *DELETE* (в от- чете привести не менее 3 примеров запросов на удаление записей).
   6. Создание представлений (в отчете привести запросы на соз- дание не менее 3 представлений).
3. Выводы.

## ЛАБОРАТОРНАЯ РАБОТА №4. ВЫПОЛНЕНИЕ ОДНОТАБЛИЧНЫХ ЗАПРОСОВ SELECT

Цель работы – освоение возможностей однотабличного операто- ра *SELECT*. Перед выполнением данной лабораторной работы изучи- те разделы 5.1, 5.2 и 5.3 учебного пособия [1], а также работу [3].

После изучения перечисленного материала выполните следую- щие варианты однотабличного оператора *SELECT*, использующие:

* предложение *WHERE* с различными вариантами условий вы- борки записей;
* агрегатные функции, обеспечивающие вычисление итоговых значений операций над всеми записями набора данных;
* запросы с группировкой.
  + 1. ВЫБОРКА ДАННЫХ С ПОМОЩЬЮ ОПЕРАТОРА SELECT Перед выполнением работ данного пункта изучите раздел 5.1

учебного пособия [1].

Выборку данных из таблиц можно выполнить в следующем по- рядке.

|  |
| --- |
|  |
| *Рис. 2.26 Окно «Query» с шаблоном запросом SELECT* |

1. ¶Вызовите редактор запросов *Query* путем выбора пункта

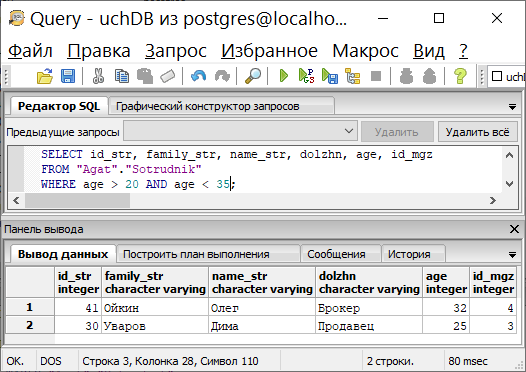
«*Скрипты/Скрипт SELECT*» КМ для выбранной вами таблицы (в нашем случае – это таблица *Sotrudnik*).

1. Для выполнения простейшего варианта запроса, не содержа- щего условий выборки записей, в вызванном редакторе на вкладке

«*Редактор SQL*» вы обнаружите шаблон выражения запроса, в ко- тором отсутствую знаки ' *?* ' ([Рис. 2.26](#_bookmark61)).

В отчете по работе приведите один пример выполнения запро- са на выборку данных из одной таблицы вашей БД.

Как правило, запросы *SELECT* содержат предложения *WHERE* с условиями выборки. Для их выполнения по описанному способу сформированный шаблон приходится модифицировать путем до- бавления предложения *WHERE* с условиями выборки.

В отчете по работе приведите по одному примеру выполнения запросов *SELECT*, содержащих предложения *WHERE* со следующи- ми условиями выборки:

*Рис. 2.27 Окно «Query» с выражением запроса SELECT с предложением WHERE*

* ¶сравнения значения одного выражения со значением другого выражения для каждой строки данных (см. пункт 5.1.4.1 [1]);
* проверки на принадлежность диапазону (см. пункт 5.1.4.2 [1]);
* проверки на членство в множестве (см. пункт 5.1.4.3 [1]);
* проверки на соответствие шаблону (см. пункт 5.1.4.4 [1]);
* проверки на значение *NULL* (см. пункт 5.1.4.5 [1]).

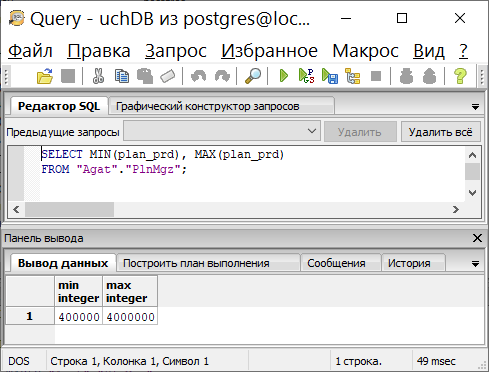
На [Рис. 2.27](#_bookmark62) изображен пример выполнения операции выборки с условием отбора со сравнением. В приведенном примере условие отбора включает два сравнения, связанные логическим оператором *AND* (по предложению *WHERE age > 20 AND age < 35* выводятся записи при одновременном выполнении двух сравнений *age > 20* и *age < 35*).

* + 1. АГРЕГАТНЫЕ ФУНКЦИИ

Далее, после изучения раздела 5.2 учебного пособия [1] выпол- ните запросы *SELECT*, содержащие агрегатные функции. Для ус- воения навыков работы с агрегатными функциями выполните за- просы *SELECT* с применением следующих функций:

* *AVG()*, вычисляющей среднее значение столбца (см. подраз- дел 5.2.1 [1]);
* *SUM()*, вычисляющей суммы значений столбца (см. подраздел 5.2.2 [1]);
* *MIN()*, вычисляющей минимальное значение столбца (см. подраздел 5.2.3 [1]);
* *MAX()*, вычисляющей максимальное значение столбца (см. подраздел 5.2.3 [1]);
* *COUNT()*, вычисляющей количество значений в столбце (см. подраздел 5.2.4 [1]).

В отчете по работе приведите по одному запросу *SELECT* на каждую агрегатную функцию. Для этого можно воспользоваться либо редактором запросов *Query*, либо графическим интерфейсом *pgAdmin*. Один из примеров выполнения такого запроса с агрегат- ной функцией приведен [Рис. 2.28](#_bookmark64).

¶

*Рис. 2.28 Выражение запроса c агрегатной функцией*

* + 1. ЗАПРОСЫ С ГРУППИРОВКОЙ

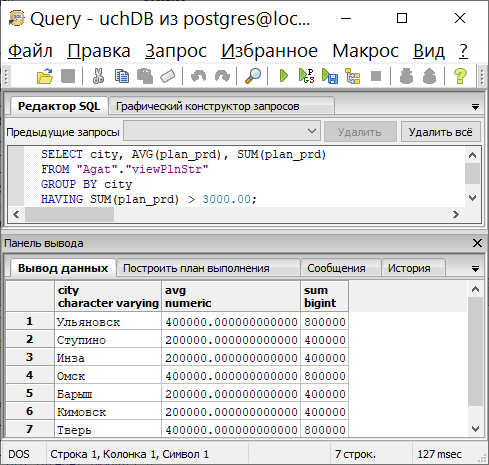
Для освоения навыков работы по формированию запросов с группировкой изучите раздел 5.3 учебного пособия [1]. Затем вы- полните запросы *SELECT* с группировкой, содержащие предложе- ния:

* *GROUP BY* (см. подраздел 5.3.1 [1]);
* *HAVING* (см. подраздел 5.3.2 [1]);
* *ORDER BY* (см. подраздел 5.3.3 [1]).

В отчете по работе опишите выполнение запросов *SELECT* (по одному примеру для каждого из перечисленных предложений). Для этого можно воспользоваться либо редактором запросов *Query*, ли- бо графическим интерфейсом *pgAdmin*.

Один из примеров выполнения запроса с агрегатными функ- циями, выполняемых для групп записей с общим значением атри- бута *city*, приведен [Рис. 2.29](#_bookmark66). В результирующий набор выводятся

¶не все вычисленные значения агрегатных функций *AVG* и *SUM*, а только те, которые удовлетворяют условию, содержащемуся в предложении *HAVING*.



*Рис. 2.29 Окно «Query» с группировкой по условию выборки HAVING и результатом его выполнения*

* + 1. СОДЕРЖАНИЕ ОТЧЕТА

После выполнения предписанных в данном разделе действий составьте отчет, содержащий следующие разделы.

1. Цель выполнения работы.
2. Описание содержания выполненных работ.
   1. Выполнение запросов выборки данных из таблиц с помо- щью оператора *SELECT* (по каждому из перечисленных ниже видов оператора *SELECT* в отчете привести выражение запроса):
      1. ¶Выполнение оператора *SELECT* с предложением *WHERE*, в котором сравниваются значения двух выражений;
      2. Выполнение оператора *SELECT* с предложением *WHERE*, в котором осуществляется проверка на принадлежность диапазону;
      3. Выполнение оператора *SELECT* с предложением *WHERE*, в котором осуществляется проверка на членство в множестве;
      4. Выполнение оператора *SELECT* с предложением *WHERE*, в котором осуществляется проверка на соответствие шаблону;
      5. Выполнение оператора *SELECT* с предложением *WHERE*, в котором осуществляется проверка проверки на значение *NULL*.
   2. Выполнение запросов *SELECT* с агрегатными функциями:
      1. Выполнение агрегатной функции *AVG()*;

2.2.1 Выполнение агрегатной функции *SUM()*;

2.2.1 Выполнение агрегатной функции *MIN()*;

2.2.1 Выполнение агрегатной функции *MAX()*;

2.2.1 Выполнение агрегатной функции *COUNT()*.

* 1. Выполнение запросов *SELECT*, содержащие агрегатные функции, использующие операцию группировки:
     1. Выполнение запросов с группировкой *GROUP BY*;
     2. Выполнение запросов с группировкой с предложением

*HAVING*;

* + 1. Выполнение запросов с группировкой с предложением

*ORDER BY*;

1. Выводы.

## ЛАБОРАТОРНАЯ РАБОТА №5. ВЫПОЛНЕНИЕ МНОГОТАБЛИЧНЫХ ЗАПРОСОВ SELECT

Цель работы – освоение возможностей использования многотаб- личного оператора *SELECT*. Перед выполнением лабораторной рабо- ты изучите разделы 5.4, 5.5 работы [1], а также работу [3].

После изучения перечисленного материала выполните следую- щие варианты многотабличного оператора *SELECT*, содержащие:

* вложенные запросы;
* многотабличные запросы с внутренними и внешними объеди- нениями.
  + 1. ВЫПОЛНЕНИЕ ВЛОЖЕННЫХ ЗАПРОСОВ

Для освоения навыков формирования вложенных запросов изу- чите раздел 5.4 учебного пособия [1].

Внимательно изучите схему выполнения вложенных запросов, согласно которой подчиненный запрос возвращает различные ре- зультаты для главного запроса. Проанализируйте, в чем заключа- ются особенности различных вариантов использования условий от- бора, и при составлении отчета опишите, в каких ситуациях каж- дый из них применяется.

Выполните запросы *SELECT*, определяющие следующие дейст- вия по отбору результатов вложенного запроса:

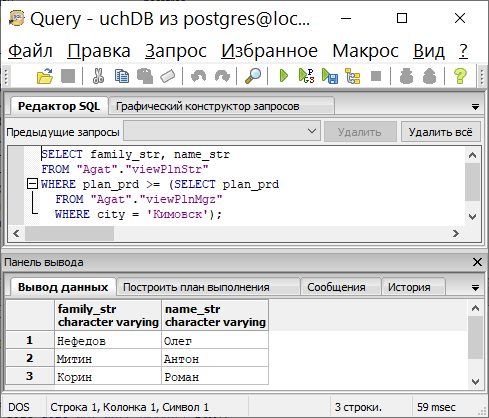
* сравнение с результатом вложенного запроса (см. пункт

5.4.2.1 [1]);

* проверку на принадлежность результатам вложенного запроса (см. пункт 5.4.2.2 [1]);
* проверку на существование (см. пункт 5.4.2.3 [1]);
* многократное сравнение (см. пункт 5.4.2.4 [1]).

В отчете по работе опишите выполнение перечисленных видов вложенных запросов и приведите по одному примеру на каждый вид запроса относительно вашей БД. Для этого можно воспользо- ваться либо редактором запросов *Query*, либо графическим интер-

¶фейсом *pgAdmin*. Пример выполнения запроса *SELECT* с вложен- ным запросом приведен на [Рис. 2.30](#_bookmark70) (см. п. 5.4.2.1 [1]).



*Рис. 2.30 Окно «Query» с запросом SELECT, выполняющим отбор вложенного запроса путем сравнения с результатом его выполнения*

* + 1. ВЫПОЛНЕНИЕ МНОГОТАБЛИЧНЫХ ЗАПРОСОВ SELECT

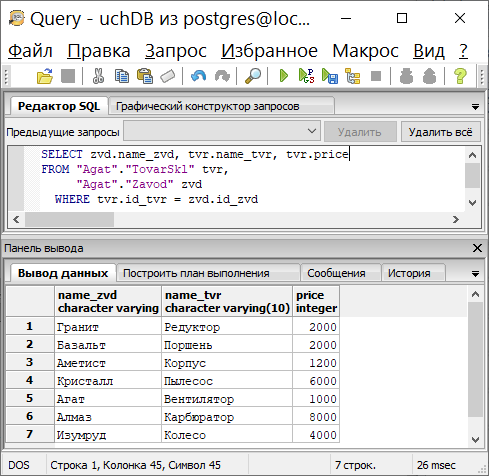
С ВНУТРЕННИМ ОБЪЕДИНЕНИЕМ ТАБЛИЦ

Перед выполнением данной части лабораторной работы изучите материал подраздела 5.5.2 учебного пособия [1] и выполните сле- дующие виды запросов *SELECT* с внутренним объединением таблиц:

* на основе точного совпадения значений двух и более столбцов (см. пункт 5.5.2.1 [1]);
* на основе отношений предок-потомок (см. пункт 5.5.2.2 [1]).

В отчете по работе опишите выполнение перечисленных видов запросов с внутренним объединением таблиц и приведите по одно-

¶му примеру на каждый вид запроса относительно вашей БД. Для этого можно воспользоваться либо редактором запросов *Query*, либо графическим интерфейсом *pgAdmin*. Пример выполнения запроса *SELECT* с внутренним объединением таблиц приведен на [Рис. 2.31](#_bookmark72) (см. п. 5.4.2.1 [1]).

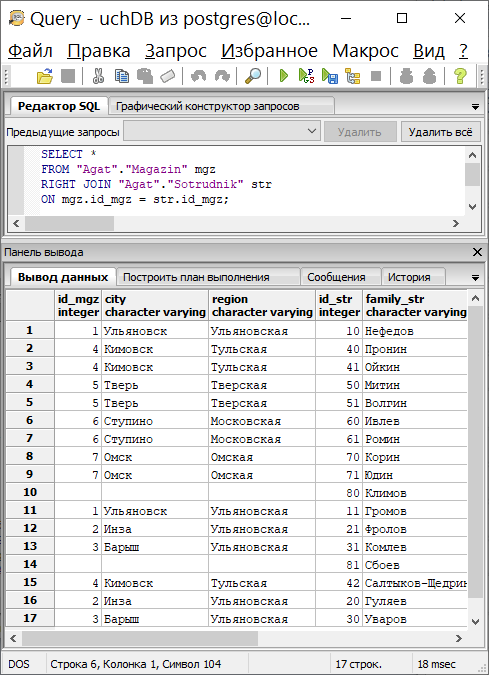


*Рис. 2.31 Окно «Query» с выражением запроса SELECT с внутренним объединением таблиц*

* + 1. ВНЕШНЕЕ ОБЪЕДИНЕНИЕ ТАБЛИЦ

Перед выполнением данной части лабораторной работы изучите материал подраздела 5.5.3 учебного пособия [1] и выполните запро- сы *SELECT* со следующими видами внешнего объединения:

* левого внешнего объединения таблиц (см. пункт 5.5.3.1 [1]);
* ¶правого внешнего объединения таблиц (см. пункт 5.5.3.2 [1]);
* полного внешнего объединения таблиц (см. пункт 5.5.3.3 [1]).



*Рис. 2.32 Окно «Query» с выражением запроса SELECT с пра- вым внешним объединением таблиц*

В отчете по работе опишите выполнение перечисленных видов запросов с внешним объединением таблиц и приведите по одному

¶примеру на каждый вид запроса относительно вашей БД. Для этого можно воспользоваться либо редактором запросов *Query*, либо гра- фическим интерфейсом *pgAdmin*. Пример выполнения запроса с правым внешним объединением таблиц приведен на [Рис. 2.32](#_bookmark74) (см. п. 5.4.2.1 [1]).

* + 1. СОДЕРЖАНИЕ ОТЧЕТА

После выполнения предписанных в данном разделе действий составьте отчет, содержащий следующие разделы.

1. Цель выполнения работы
2. Описание содержания выполненных работ.
   1. Выполнение запросов *SELECT* с вложенными запросами (по каждому из перечисленных ниже видов оператора *SELECT* в отчете привести выражение запроса):
      1. Описание запросов *SELECT*, осуществляющих сравнение с результатом вложенного запроса;
      2. Описание запросов *SELECT*, осуществляющих проверку на принадлежность результатам вложенного запроса;
      3. Описание запросов *SELECT*, осуществляющих проверку на существование;
      4. Описание запросов *SELECT*, осуществляющих многократ- ное сравнение.
   2. Выполнение многотабличных запросов *SELECT* с внутрен- ним объединением таблиц (по каждому из перечисленных ниже ви- дов оператора *SELECT* в отчете привести выражение запроса):
      1. Внутреннее объединение таблиц на основе точного сов- падения значений двух и более столбцов;
      2. Внутреннее объединение таблиц с использованием отно- шения предок-потомок.
   3. ¶Выполнение многотабличных запросов *SELECT* с внешним объединением таблиц (по каждому из перечисленных ниже видов оператора *SELECT* в отчете привести выражение запроса):
      1. Выполнение многотабличных запросов *SELECT* с левым внешним объединением;
      2. Выполнение многотабличных запросов *SELECT* с правым внешним объединением;
      3. Выполнение многотабличных запросов *SELECT* с полным внешним объединением.
3. Выводы.

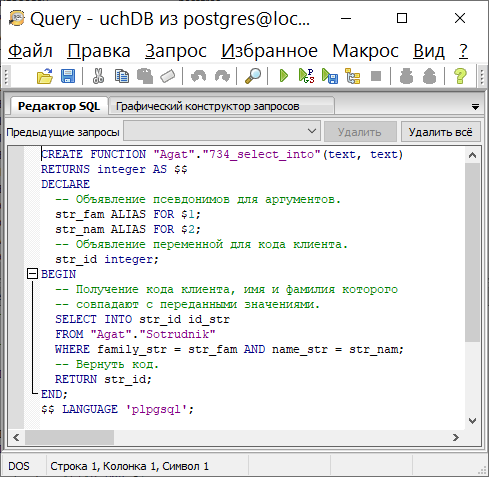
## ЛАБОРАТОРНАЯ РАБОТА №6. РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР

Цель работы – освоение различных аспектов разработки функ- ций хранимых процедур с использованием языка *PL/pgSQL*, под- держиваемого СУБД *PostgreSQL*.

Перед выполнением данной лабораторной работы внимательно изучите материалы раздела 7.1 работы [1], а также работы [4, 5, 6].

Создание и проверка работы хранимых процедур осуществля- ется по следующей схеме.

1. С помощью редактора запросов *Query* введите текст храни- мой процедуры ([Рис. 2.33](#_bookmark77)).



*Рис. 2.33 Окно «Query» с текстом хранимой процедуры*

1. ¶Нажатием кнопки «Выполнить запрос» запустите процесс компиляции. Если после этого на *Панели вывода* (в нижней части окна редактора) появляется сообщение о(б):

* ошибке, это означает, что в вашем коде содержится либо син- таксическая, либо логическая ошибка, и ваш код необходимо скор- ректировать и снова запустить на выполнение;
* завершении выполнения запроса, это означает, что процесс компиляции процедуры завершен и его объектный код сохранен в БД в качестве ее объекта ([Рис. 2.33](#_bookmark77)).

1. После удачного завершения компиляции хранимой процеду- ры на *Браузере объектов* графического интерфейса *pgAdmin* рас- кройте узел *Функции* в схеме вашей БД и нажмите на кнопку «*Об- новить выделенный объект*». В данном узле должен появиться пункт с названием скомпилированной процедуры или функции. В нашем примере – это *734\_select\_into(text, text)*.
2. Для проверки работы хранимой процедуры осуществите ее вызов по команде *SELECT* (описание процесса вызова хранимой процедуры см. в разделе 7.4 учебного пособия [1]).
   * 1. ИСПОЛЬЗОВАНИЕ КОМАНДЫ SELECT INTO В ХРАНИМЫХ ПРОЦЕДУРАХ

Перед выполнением данной части лабораторной работы изучите материалы подразделов 7.2.1, 7.2.2, 7.2.3 и пункта 7.2.4.1 учебного пособия [1]. Создайте хранимую процедуру для вашей БД, исполь- зующей команду *SELECT INTO*, по аналогу, приведенному в Лис- тинге 7.12 учебного пособия.

* + 1. ИСПОЛЬЗОВАНИЕ АТРИБУТА %TYPE

Для выполнения данной части лабораторной работы изучите материалы пункта 7.2.4.2 учебного пособия [1]. Создайте хранимую процедуру для вашей БД, использующую атрибут *%TYPE*, по ана- логу, приведенному в Листинге 7.14.

* + 1. ¶ИСПОЛЬЗОВАНИЕ АТРИБУТА %ROWTYPE

Для выполнения данной части лабораторной работы изучите материалы пункта 7.2.4.3 учебного пособия [1]. Создайте хранимую процедуру для вашей БД, использующую атрибут *%ROWTYPE* , по аналогу, приведенному в Листинге 7.15.

* + 1. ИСПОЛЬЗОВАНИЕ УСЛОВНЫХ КОМАНД

Для выполнения данной части лабораторной работы изучите материалы подраздела 7.2.5 учебного пособия [1]. Создайте храни- мые процедуры для вашей БД, использующие условные команды:

* *IF/THEN* (аналог см. в Листинге 7.17);
* *IF/THEN/ELSE* (аналог см. в Листинге 7.19);
* *IF/THEN/ELSE/IF* (аналог см. в Листинге 7.20).
  + 1. ИСПОЛЬЗОВАНИЕ ЦИКЛОВ

Для выполнения данной части лабораторной работы изучите материалы подраздела 7.2.6 учебного пособия [1]. Создайте храни- мую процедуру для вашей БД, использующую команду цикла *FOR* (аналог см. в Листинге 7.23).

* + 1. РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР, ОБЕСПЕЧИВАЮЩИХ ВЫПОЛНЕНИЕ ПРИЛОЖЕНИЯ ВАШЕЙ БД

Для выполнения данной части лабораторной работы изучите материалы раздела 7.4 учебного пособия [1]. В зависимости от ва- шего варианта создайте хранимые процедуры, обеспечивающие выполнение процессов:

* поступления товаров (см. пункт 7.4.1.1);
* продажи товаров (см. пункт 7.4.1.2);
* архивирование поступления и продаж товаров (см. подраз- дел 7.4.2);
* планирование поступления и продаж товаров (см. подраз- дел 7.4.2).

¶Обратите внимание на то обстоятельство, что в подразделе 7.4.2 процессы архивирования и планирования описаны в рамках еди- ной процедуры. Вам предстоит эти процессы разработать и опи- сать как отдельные процедуры.

* + 1. СОДЕРЖАНИЕ ОТЧЕТА

После выполнения предписанных в данном разделе действий составьте отчет, содержащий следующие разделы.

1. Цель выполнения работы.
2. Описание содержания выполненных работ.
   1. Использование команды *SELECT INTO* в хранимых проце- дурах (в отчете приведите пример хранимой процедуры, обрабаты- вающей данные таблицы БД вашего варианта и содержащей коман- ду *SELECT INTO*).
   2. Использование атрибута *%TYPE* в хранимых процедурах (в отчете приведите пример хранимой процедуры, обрабатывающей данные таблицы БД вашего варианта и содержащей атрибут типа

*%TYPE*).

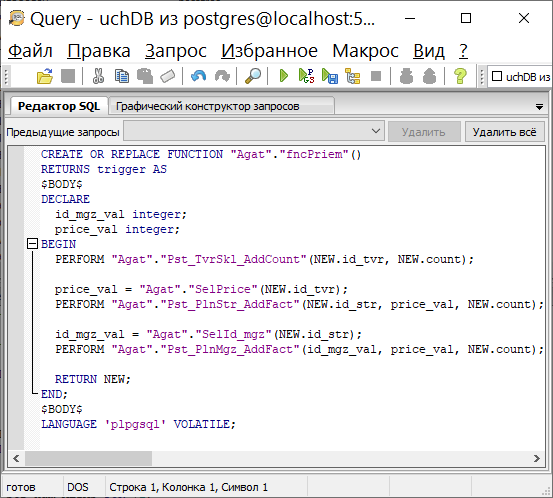
* 1. Использование атрибута *%ROWTYPE* в хранимых процеду- рах (в отчете приведите пример хранимой процедуры, обрабаты- вающей данные таблицы БД вашего варианта и содержащей атри- бут типа *%ROWTYPE*).
  2. Использование условных команд в хранимых процедурах (в отчете приведите пример хранимой процедуры, обрабатывающей данные таблицы БД вашего варианта и содержащей условные ко- манды).
  3. Использование циклов в хранимых процедурах (в отчете приведите пример хранимой процедуры, обрабатывающей данные таблицы БД вашего варианта и содержащей команды циклов).

1. Выводы.

## ЛАБОРАТОРНАЯ РАБОТА №7. РАЗРАБОТКА ТРИГГЕРНЫХ ФУНКЦИЙ И ТРИГГЕРОВ

Цель работы – освоение различных аспектов разработки триг- герных функций и триггеров с использованием языка *PL/pgSQL*, поддерживаемого СУБД *PostgreSQL*.

Перед выполнением данной лабораторной работы внимательно изучите материалы раздела 8.1 работы [1], а также работы [5, 6, 7].



*Рис. 2.34 Окно «Query» с текстом триггерной функции*

* + 1. ¶СОЗДАНИЕ ТРИГГЕРНЫХ ФУНКЦИЙ И ТРИГГЕРОВ

Создание триггерных функций осуществляется по схеме, опи- санной в предыдущей работе (см. преамбулу к разделу 2.6). Пример триггерной функции приведен на [Рис. 2.34](#_bookmark86).

Небольшое отличие процесса, описываемого в данном разделе, заключается в том, что после успешной компиляции триггерной функции на *Браузере объектов* графического интерфейса *pgAdmin* пункт с названием скомпилированной триггерной функции (в на- шем примере – это *fncPriem()*) появляется в узле *Триггерные функции*.

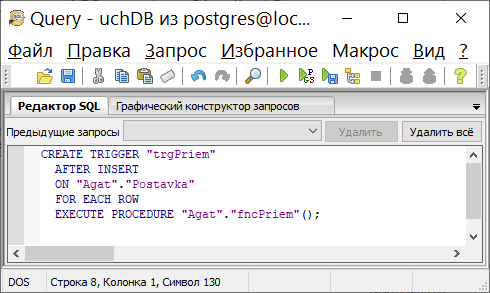
Триггерные функции отличаются от хранимых процедур также в части синтаксиса и использования специальных переменных, со- держащих информацию о сработавшем триггере. Кроме этого триг- герные функции могут содержать вызовы хранимых процедур.

Триггер в *PostgreSQL* можно создать двумя способами:

1. Набором выражения текста определения триггера (см. под- раздел 8.1.2) в редакторе запросов *Query* ([Рис. 2.35](#_bookmark88)):

* введите текст определения триггера в редакторе запросов

*Query*;

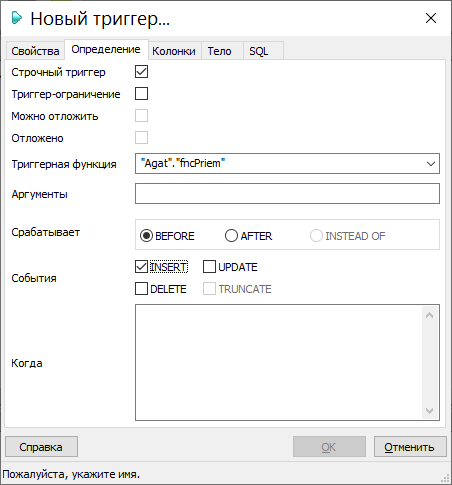


*Рис. 2.35 Окно «Query» с текстом определения триггера*

* ¶нажмите кнопку «*Выполнить запрос*»;
* убедитесь, что в узле *Триггеры* таблицы, для которой предна- значен создаваемый триггер, появилось наименование триггера.

1. С помощью графического интерфейса *pgAdmin*:

* для выделенной на *Браузере объектов* таблицы (*Postavka*) вызовите диалог «*Новый триггер* …» путем выбора пункта «*Но- вый объект/Новый триггер …*» КМ ([Рис. 2.36](#_bookmark89));



*Рис. 2.36 Окно «Новый триггер» с параметрами определения триггера*

* на вкладке «*Свойства*» окна диалога наберите наименова- ние создаваемого триггера;
* на вкладке «*Определение*» выберите чекбокс «*Строчный триггер*»;
* ¶на вкладке «*Определение*» с помощью выпадающего списка выберите требуемую триггерную функцию;
* на вкладке «*Определение*» в поле «*Срабатывает*» с помо- щью радиокнопки выберите момент срабатывания триггера (*BE- FORE* или *AFTER*);
* на вкладке «*Определение*» с помощью выпадающего списка выберите требуемую триггерную функцию;
* на вкладке «*Определение*» выберите чекбокс *INSERT*, *UPDATE*

или *DELETE*;

* нажмите на кнопку *OK* и убедитесь, что в узле *Триггеры* таб- лицы, для которой предназначен создаваемый триггер, появилось наименование триггера.
  + 1. РАЗРАБОТКА ТРИГГЕРОВ, ОБЕСПЕЧИВАЮЩИХ ВЫПОЛНЕНИЕ ПРОЦЕДУРЫ ВАШЕЙ БД

Для выполнения данной части лабораторной работы изучите материалы подраздела 8.3.2 учебного пособия [1]. В зависимости от вашего варианта создайте триггерные функции, обеспечивающие выполнение процессов либо:

* поступления товаров (см. пункт 7.4.1.1);
* продажи товаров (см. пункт 7.4.1.2);
  + 1. СОДЕРЖАНИЕ ОТЧЕТА

После выполнения предписанных в данном разделе действий составьте отчет, содержащий следующие разделы.

1. Цель выполнения работы.
2. Описание содержания выполненных работ.
   1. Описание схем разработки триггерных функций и тригге- ров.
   2. Разработка триггерных функций и триггеров, обеспечи- вающих выполнение процесса вашего варианта.
3. Выводы.

¶

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Токмаков Г. П. Базы данных: модели и структуры данных, язык SQL, программирование баз данных : Учебное пособие / Г. П. Токмаков. – Ульяновск: УлГТУ, 2021. − 362 с.
2. Электронный документ. https:/[/w](http://www.intuit.ru/studies/courses/)w[w.intuit.ru/studies/courses/](http://www.intuit.ru/studies/courses/) 953/191/lecture/4969. Владимир Туманов. [Основы проектирования](https://www.intuit.ru/studies/courses/1095/191/info) [реляционных баз данных](https://www.intuit.ru/studies/courses/1095/191/info) (дата обращения: 12.11.2020).
3. Электронный документ. https://postgrespro.ru/docs/postgrespro/9.5/ index.html - документация PostgreSQL (дата обращения: 12.11.2020).
4. Электронный документ. <http://stydopedia.ru/3x68e8.html>- Поня- тие хранимой процедуры (дата обращения: 12.11.2020).
5. Электронный документ. <http://stydopedia.ru/3x6df8.html>- Соз- дание хранимой процедуры (дата обращения: 12.11.2020).
6. Электронный документ. <http://helpiks.org/2-36754.html> - Пере- менные типа RECORD и ROWTYPE (дата обращения: 12.11.2020).
7. Электронный документ. <http://stydopedia.ru/3x68e9.html>- По- нятие триггера (дата обращения: 12.11.2020).
8. Электронный документ. <http://java-course.ru/begin/database01/>

* Базы данных на Java – введение (дата обращения: 12.11.2020).

1. Электронный документ. <http://citforum.ru/database/dblearn/db-> learn06.shtml (дата обращения: 12.11.2020).

¶

|  |
| --- |
| Учебное издание  ТОКМАКОВ Геннадий Петрович  **Базы данных:**  **язык SQL, программирование баз данных**  Учебно-методическое пособие по выполнению лабораторных работ |
| Редактор Н. А. Евдокимова |
| ЛР №020640 от 22.10.97  Подписано в печать 20.12.2022. Формат 6084/16.  Усл. печ. л. 5,23. Тираж 40 экз. Заказ 567. ЭИ № 1757. |
| Ульяновский государственный технический университет 432027, г. Ульяновск, ул. Сев. Венец, 32.  ИПК «Венец» УлГТУ, 432027, г. Ульяновск, ул. Сев. Венец, 32. |