Министерство науки и высшего образования РФ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский государственный университет имени А.Н.Косыгина**

**(Технологии. Дизайн. Искусство)»**

|  |  |
| --- | --- |
| **Институт**\_\_информационных технологий и цифровой трансформации  **Кафедра**\_\_автоматизированных систем обработки информации и управления | |
|  |

**ОТЧЕТ**

**об** \_\_\_учебной практике. Эксплуатационной\_\_ **практике**

**Уровень освоения основной**

**профессиональной**

**образовательной программы (ОПОП)**  академический бакалавриат

**Направление подготовки (специальность)**  09.03.01 Информатика и вычислительная техника

**Профиль (специализация)** Автоматизированные системы обработки информации и управления

**Форма обучения** очная

**Способ прохождения практики**  стационарная

**Форма проведения практики**  непрерывная

**Сроки прохождения практики с «**10**»\_**февраля**\_ 20**25**\_\_\_ г. по «**26**»\_**июня**\_\_\_ 20**25**\_\_ г.**

**Место прохождения практики \_\_**РГУ им. А.Н. Косыгина**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Отчет составил и сдал:** «26» июня 2025 г.\_\_\_\_\_\_\_Ольховский Н.С.**\_\_\_\_\_\_\_\_\_\_\_**

группа ИТА-123

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*подпись практиканта)*

**Руководитель практики от университета** \_\_Травкин Е.И.\_\_\_\_\_\_

**ОЦЕНКА работы на практике** \_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_»\_\_\_\_\_20\_\_\_ г.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись руководителя практики от университета)*

**Москва, 2025 г.**

Содержание

[Введение 3](#_Toc200498301)

[История и развитие PostgreSQL 4](#_Toc200498302)

[Особенности и преимущества PostgreSQL 5](#_Toc200498303)

[Основы языка SQL 7](#_Toc200498304)

[Создание баз данных, работа с таблицами и запросами 8](#_Toc200498305)

[Дополнительные средства PostgreSQL. 11](#_Toc200498306)

[Рынок труда и карьера в области баз данных 13](#_Toc200498307)

[Перспективы и тенденции развития 15](#_Toc200498308)

[Итоговое задание и анализ выполненной работы 16](#_Toc200498309)

[Заключение 26](#_Toc200498310)

[Список литературы 27](#_Toc200498311)

Введение

Современные информационные технологии немыслимы без эффективного хранения, управления и анализа данных. Независимо от отрасли — будь то здравоохранение, финансы, образование или электронная коммерция — базы данных лежат в основе цифровой инфраструктуры, обеспечивая целостность, доступность и согласованность информации. В этой связи особую актуальность приобретают практические знания и навыки работы с системами управления базами данных (СУБД), среди которых PostgreSQL занимает одно из ведущих мест. Она является мощной и свободной для использования СУБД с открытым исходным кодом, которая успешно используется как в небольших стартапах, так и в крупных корпоративных системах.

Задачей учебной практики является закрепление теоретических знаний, полученных в рамках изучения дисциплин, связанных с базами данных и программированием. Необходимо научиться проектировать реляционные модели, формировать запросы на языке SQL, владение которым считается одним из базовых навыков для работы с данными в любой отрасли, будь то банковская сфера, здравоохранение, логистика или телекоммуникации, создавать таблицы и связи между ними, а также обеспечивать корректную обработку и защиту информации. Немаловажной целью стало освоение инструментов, таких как pgAdmin, а также изучение механизмов резервного копирования и восстановления, что особенно важно в условиях работы с критически важными данными.

# История и развитие PostgreSQL

PostgreSQL — это одна из старейших и проверенных временем систем управления базами данных с открытым исходным кодом. Её история начинается в 1986 году в Калифорнийском университете в Беркли, где профессор Майкл Стоунбрейкер возглавлял исследовательский проект POSTGRES. Этот проект был логическим продолжением более ранней разработки — системы Ingres, одной из первых реляционных СУБД.

Целью проекта POSTGRES было устранение ограничений традиционных реляционных баз данных и внедрение новых концепций, таких как:

1. Расширяемость СУБД за счёт поддержки пользовательских типов данных и операторов;
2. Управление объектно-реляционными структурами;
3. Поддержка версионности данных и времени выполнения запросов.

Проект POSTGRES развивался в течение нескольких лет как исследовательская инициатива. В 1994 году началась трансформация системы в полноценную СУБД, пригодную для практического использования. В этот момент была добавлена поддержка языка SQL, после чего проект получил название PostgreSQL, обозначающее "POSTGRES + SQL".

Начиная с 1996 года, PostgreSQL стал развиваться как открытое программное обеспечение. Это положило начало долгому пути эволюции системы, её постоянной модернизации и расширению функциональности благодаря усилиям большого международного сообщества.

На сегодняшний день PostgreSQL — это высоко надёжная СУБД, поддерживающая как реляционные, так и объектно-ориентированные принципы хранения и обработки данных.

# Особенности и преимущества PostgreSQL

PostgreSQL — это объектно-реляционная система управления базами данных с открытым исходным кодом, которая заслуженно считается одной из самых надёжных, мощных и гибких платформ для работы с данными. Её основные преимущества и особенности определяют высокую популярность среди разработчиков, администраторов баз данных и аналитиков по всему миру.

Среди ключевых особенностей PostgreSQL можно выделить открытый исходный код - система распространяется бесплатно и может быть модифицирована под индивидуальные нужды, что делает её особенно привлекательной для организаций с ограниченным бюджетом и разработчиков, заинтересованных в глубокой настройке. PostgreSQL также поддерживает обширный набор функций, определённых стандартами ANSI SQL, а также включает множество расширений, выходящих за рамки базовых возможностей. Пользователи могут создавать собственные типы данных, функции, операторы и даже языки программирования. Это делает её платформой, адаптируемой под любые потребности. PostgreSQL обеспечивает целостность и согласованность данных даже в условиях высокой нагрузки и параллельного доступа, что делает её надёжным решением для критически важных приложений. Механизм многоверсионности позволяет эффективно обрабатывать конкурентные запросы без блокировок чтения, повышая производительность и отклик системы. Также в последние годы PostgreSQL развился не только как реляционная, но и как универсальная платформа с поддержкой JSON, hstore и других форм полуструктурированных данных. Кроме того, PostgreSQL активно используется в самых разных отраслях — от финансов и здравоохранения до телекоммуникаций и государственного сектора. Это стало возможным благодаря её стабильности, высокой безопасности и отличной масштабируемости.

Архитектура PostgreSQL построена по принципу клиент-серверной модели. Она включает несколько ключевых компонентов:

1. Клиент: взаимодействует с пользователем или внешним приложением, передаёт команды серверу СУБД.
2. Сервер PostgreSQL (postmaster): основной управляющий процесс, координирующий работу базы данных, управление подключениями, аутентификацию и запуск других процессов.
3. Процессы-воркеры: отдельные процессы, обслуживающие клиентские соединения, выполняющие SQL-запросы, операции с транзакциями и другими задачами.
4. Система хранения: PostgreSQL использует файловую систему ОС для хранения данных. Каждая база, таблица и индекс хранятся в отдельных файлах, структурированных в рамках каталогов базы данных.
5. Библиотека libpq: интерфейс взаимодействия клиента и сервера через протокол PostgreSQL.
6. Журнал транзакций: обеспечивает отказоустойчивость и возможность восстановления после сбоев, записывая все изменения перед их внесением в базу.

Для управления и взаимодействия с PostgreSQL существует широкий спектр инструментов, обеспечивающих как консольный, так и графический интерфейс. psql — это консольный клиент для PostgreSQL, предоставляющий гибкий и мощный интерфейс командной строки. pgAdmin — официальный графический интерфейс, предоставляющий пользователям удобные средства для визуального администрирования. DBeaver, DataGrip, OmniDB — сторонние GUI-клиенты, которые поддерживают PostgreSQL и позволяют работать с ним в многобазовой среде, особенно полезны при использовании нескольких СУБД. pg\_dump и pg\_restore — стандартные утилиты PostgreSQL для создания резервных копий и восстановления баз данных.

Выбор конкретного инструмента зависит от предпочтений пользователя и задач, стоящих перед ним. Консольные инструменты особенно полезны в автоматизации, а графические — при визуальном проектировании или обучении. В нашей учебной практике был использован pgAdmin.

# Основы языка SQL

Язык структурированных запросов SQL (Structured Query Language) — это стандартизированный язык программирования, используемый для взаимодействия с реляционными базами данных. Он является основным средством управления данными: от создания структуры базы до выполнения сложных аналитических запросов. PostgreSQL, как одна из наиболее полноценных реализаций СУБД, обеспечивает поддержку практически всех аспектов SQL, а также предоставляет собственные расширения и дополнения к синтаксису.

SQL — декларативный язык, в котором пользователь описывает, *что* он хочет получить, а не *как* это должно быть реализовано. Основной единицей языка является запрос (statement), состоящий из зарезервированных ключевых слов, идентификаторов объектов, выражений, операторов и условий.

Запросы могут быть простыми (однострочными) или состоять из нескольких вложенных операторов и выражений. Структура SQL-запросов логична и понятна, что делает язык доступным даже для начинающих пользователей.

Язык SQL условно делится на несколько категорий команд:

1. DDL (Data Definition Language) — язык определения данных:
2. DML (Data Manipulation Language) — язык управления данными:
3. DCL (Data Control Language) — язык управления доступом:
4. DQL (Data Query Language) — язык запросов к данным:

Эти категории обеспечивают полный цикл работы с данными — от проектирования структуры до контроля доступа и анализа информации.

Синтаксис SQL в PostgreSQL основан на стандарте ANSI SQL, но дополнен специфическими расширениями, упрощающими написание сложных запросов. Все команды SQL нечувствительны к регистру, но принято писать ключевые слова заглавными буквами для лучшей читаемости.

# **Создание баз данных**, р**абота с таблицами и запросами**

Процесс работы с базами данных в PostgreSQL начинается с этапа проектирования, который играет ключевую роль в создании надёжной, масштабируемой и легко поддерживаемой системы хранения данных. Грамотно спроектированная база данных обеспечивает не только целостность и согласованность информации, но и повышает производительность при выполнении запросов, упрощает администрирование и сопровождается меньшими издержками в долгосрочной перспективе.

Первым шагом проектирования является построение реляционной модели данных. Это включает в себя определение предметной области, выявление сущностей (например, «студент», «курс», «преподаватель»), атрибутов, характеризующих эти сущности (например, имя, дата рождения, номер зачетной книжки), а также взаимосвязей между ними. На этом этапе важно определить, какие связи существуют: один-к-одному, один-ко-многим или многие-ко-многим. Например, студент может быть записан на несколько курсов, и каждый курс может иметь множество студентов — это отношение многие-ко-многим, для которого потребуется дополнительная промежуточная таблица.

Для улучшения структуры данных используется нормализация — процесс разбиения таблиц таким образом, чтобы устранить избыточность и повысить логическую целостность. Это позволяет исключить повторяющиеся данные и минимизировать возможность возникновения аномалий при обновлении или удалении записей.

После завершения логического проектирования выполняется физическое создание структуры базы данных при помощи языка SQL. С помощью операторов DDL в PostgreSQL создаются таблицы, устанавливаются ограничения (например, первичный ключ, внешний ключ, уникальные значения, not null), задаются типы данных для каждого поля. Таким образом, формируется основа базы, на которую затем накладываются данные.

Наполнение таблиц происходит посредством SQL-команд INSERT INTO, которые позволяют добавлять строки данных. При необходимости данные могут быть изменены или удалены с помощью UPDATE и DELETE. При этом важно следить за сохранением целостности информации, особенно если между таблицами существуют связи по внешним ключам. PostgreSQL автоматически предотвращает удаление связанных записей, если это нарушит логическую непротиворечивость базы данных, однако иногда для обеспечения гибкости используют каскадное удаление (ON DELETE CASCADE), которое должно быть применено осознанно.

Основной способ извлечения информации из базы данных — это выполнение SQL-запросов. Оператор SELECT позволяет получать данные, удовлетворяющие тем или иным условиям. Используя WHERE, можно фильтровать записи по заданным критериям. Команды ORDER BY, GROUP BY, HAVING обеспечивают сортировку, группировку данных и фильтрацию групп, что особенно важно при использовании агрегатных функций (например, COUNT, AVG, SUM, MIN, MAX). Эти функции позволяют собирать сводную информацию и анализировать данные, как, например, общее число заказов, среднюю стоимость товаров или максимальную оценку студента.

Одним из важнейших аспектов является работа с объединениями таблиц. Команды JOIN позволяют связать данные из нескольких таблиц по общим признакам. PostgreSQL поддерживает различные виды соединений: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, каждый из которых решает свою задачу в зависимости от логики запроса и структуры данных. Кроме того, большое значение имеют подзапросы — вложенные SELECT, которые используются в качестве фильтров или источников данных для основного запроса. Это позволяет строить гибкие, многоуровневые выборки.

Важной частью проекта является поддержание целостности данных. PostgreSQL предлагает множество встроенных механизмов для обеспечения этого: ограничения (CONSTRAINT), правила ссылочной целостности, автоматическое применение ограничений при изменении или удалении записей. Это помогает защитить данные от логических ошибок и несогласованных изменений.

Таким образом, весь процесс — от проектирования до написания запросов — представляет собой единый цикл разработки базы данных, в котором применяются как логические, так и технические подходы. Полученные в ходе учебной практики знания и навыки по проектированию, созданию, наполнению и запросам к базам данных стали фундаментом для дальнейшего углублённого изучения систем управления базами данных и подготовки к профессиональной деятельности в этой области.

# **Дополнительные средства PostgreSQL. Оптимизация, администрирование, безопасность**

Современные базы данных выходят далеко за рамки простого хранения и выборки информации. PostgreSQL как одна из самых мощных и гибких СУБД открытого кода предлагает не только классический инструментарий для работы с данными, но и расширенные средства, значительно повышающие возможности по их обработке, защите, администрированию и восстановлению.

Одним из таких средств являются представления. Представление (VIEW) представляет собой виртуальную таблицу, содержащую результат выполнения SQL-запроса. Это мощный инструмент для организации удобных слоёв доступа к данным, особенно в случаях, когда необходимо скрыть сложность исходной схемы базы, ограничить доступ к конкретным столбцам или строкам, а также повысить читаемость и повторное использование сложных запросов. Представления позволяют разработчику сконцентрироваться на логике приложения, не перегружая запросы избыточной технической детализацией.

PostgreSQL также предоставляет богатый функционал по созданию пользовательских функций и процедур. С помощью встроенного языка PL/pgSQL можно создавать сложную бизнес-логику, исполняемую непосредственно на стороне сервера. Это особенно полезно, когда требуется выполнение однотипных операций или вложенных условий в рамках одной транзакции. Аналогично, триггеры и события позволяют автоматизировать реакции базы данных на изменения. Триггеры могут срабатывать до или после операций вставки, обновления или удаления, обеспечивая, например, автоматическое логирование действий пользователей или проверку условий целостности, выходящих за пределы стандартных ограничений.

Особое внимание в PostgreSQL уделяется индексации, которая является ключевым фактором оптимизации производительности при работе с большими объёмами данных. Помимо обычных деревьев, система поддерживает хеш-индексы, GIN (для полнотекстового поиска), GiST (для работы с географической информацией), BRIN и другие типы. Каждый из них применяется в зависимости от характера данных и требований к запросам. Грамотно построенные индексы значительно ускоряют выполнение операций выборки, особенно по условиям поиска, фильтрации и соединения.

Оценка и анализ производительности запросов в PostgreSQL осуществляется через команду EXPLAIN, которая выводит план выполнения SQL-запроса. Она позволяет разработчику понять, каким образом СУБД читает таблицы, использует индексы, какие операции являются «узкими местами» и где возможна оптимизация. В сочетании с ключевыми метриками и статистикой использования, EXPLAIN становится незаменимым инструментом для устранения проблем в производительности.

Администрирование базы данных включает в себя управление пользователями, ролями и их правами доступа. PostgreSQL предлагает гибкую систему авторизации, в которой можно создавать как отдельные учётные записи, так и объединять их в роли с различными правами. Это позволяет ограничивать доступ к определённым данным или операциям, исходя из принципа минимально необходимого уровня доступа. Механизмы безопасности реализуются через конфигурационные файлы, политики подключения, шифрование соединений и журналирование действий.

Неотъемлемой частью сопровождения любой информационной системы является резервное копирование. PostgreSQL предлагает надёжные инструменты для создания и восстановления резервных копий. Среди них — pg\_dump, который позволяет создавать дампы отдельных баз или объектов в виде SQL-скриптов или в бинарном формате, и pg\_restore, который применяется для восстановления этих данных. В дополнение к этому предусмотрены инструменты для полной архивации базы (base backup) и журналирования транзакций (WAL), что позволяет реализовать точечное восстановление до конкретного состояния в истории.

# Рынок труда и карьера в области баз данных

В эпоху цифровизации и активного роста объёмов данных профессии, связанные с базами данных, приобретают всё большую значимость. PostgreSQL как современная, мощная и свободная СУБД (система управления базами данных) нашла широкое применение в компаниях различного масштаба: от стартапов до государственных организаций и международных корпораций. Это способствует устойчивому спросу на специалистов, владеющих инструментами администрирования, проектирования и программирования в PostgreSQL и SQL.

Востребованность специалистов по PostgreSQL

PostgreSQL активно используется в разработке веб-приложений, аналитических систем, платформ хранения данных и микросервисной архитектуре.

Открытый исходный код и отсутствие лицензий на использование, совместимость с облачными провайдерами, широкая экосистема и стабильное развитие приводят к устойчивому спросу на специалистов, владеющих PostgreSQL. По данным вакансий на карьерных платформах (hh.ru, LinkedIn, Indeed), позиции с этим навыком стабильно присутствуют в категориях «ИТ», «Разработка», «Аналитика данных».

Работодатели чаще всего ищут специалистов на следующие должности:

1. Администратор баз данных (DBA) — отвечает за настройку, резервное копирование, безопасность и мониторинг производительности баз данных;
2. Разработчик SQL/PostgreSQL — занимается написанием запросов, созданием функций, процедур, триггеров, проектированием схем данных;
3. Бэкенд-разработчик — активно использует PostgreSQL как основную СУБД для взаимодействия с приложением;
4. Системный архитектор — включает PostgreSQL в архитектуру высоконагруженных и распределённых систем;
5. Аналитик данных (Data Analyst) или BI-инженер — использует SQL и PostgreSQL для извлечения, анализа и визуализации данных.

К основным требованиям работодателей относятся:

* Знание языка SQL и опыт работы с PostgreSQL;
* Умение проектировать реляционные базы данных;
* Опыт написания сложных запросов, процедур, триггеров;
* Навыки администрирования, настройки прав доступа, резервного копирования;
* Знание инструментов pgAdmin, psql, опыт работы с EXPLAIN;
* Опыт работы с большими объёмами данных, знание принципов оптимизации.

Начав с позиции младшего аналитика или помощника администратора, специалист может развиться в одного из следующих профессионалов: ведущий разработчик PostgreSQL, технический архитектор, руководитель отдела аналитики/данных, эксперт по безопасности БД, консультант или DevOps-инженер с фокусом на БД.

Область баз данных требует постоянного обновления знаний, участия в профессиональных сообществах, чтения документации и работы над реальными проектами. PostgreSQL предлагает широкий простор для профессионального роста — от технических до управленческих ролей.

# Перспективы и тенденции развития

Современный мир предъявляет всё более высокие требования к системам управления базами данных. Рост объёмов информации, внедрение облачных решений, Big Data, искусственного интеллекта и аналитики требуют от СУБД гибкости, масштабируемости и высокой производительности. В этом контексте PostgreSQL продолжает активно развиваться и укреплять свои позиции как одно из ведущих решений с открытым исходным кодом.

PostgreSQL демонстрирует стабильный рост популярности в течение последних лет. Согласно рейтингам DB-Engines, эта СУБД входит в тройку самых востребованных в мире.

Тем не менее, PostgreSQL конкурирует и с другими популярными решениями. MySQL широко используется, особенно в веб-разработке, но уступает PostgreSQL в функциональности. Oracle, Microsoft SQL Server — мощные коммерческие СУБД, часто используются в крупных корпоративных системах. Однако благодаря открытости, сообществу и широкому набору функций PostgreSQL сохраняет лидерские позиции и остаётся перспективной СУБД на ближайшие годы.

Одной из самых заметных тенденций является переход систем в облачную инфраструктуру. PostgreSQL отлично вписывается в эту парадигму. Облачные технологии позволяют масштабировать PostgreSQL в зависимости от нагрузки, упрощают резервное копирование, обновление версий и управление отказоустойчивостью.

С ростом объёмов данных возникает потребность в аналитике, бизнес-аналитических системах и платформенных решениях для обработки больших данных. PostgreSQL активно адаптируется к этим реалиям.

# Итоговое задание и анализ выполненной работы

Учебная практика по дисциплине «PostgreSQL и язык SQL» включала в себя как теоретическую подготовку, так и практические задания, направленные на закрепление и применение полученных знаний. Центральным элементом практики стало итоговое задание, в рамках которого мы спроектировали и реализовали собственную реляционную базу данных, используя возможности PostgreSQL и языка SQL. Этот этап позволил продемонстрировать понимание основ проектирования БД, владение SQL и уверенное использование инструментов PostgreSQL.

Описание задания:

* Разработать предметную область «Онлайн торговля»;
* Определить сущности и связи между ними;
* Создать схему базы данных с помощью Power Architect;
* Наполнить таблицы тестовыми данными;
* Выполнить выборки с применением команд SELECT, JOIN, GROUP BY, HAVING и подзапросов;
* Реализовать пользовательские представления, функции;
* Настроить права доступа для пользователей;
* Выполнить экспорт и восстановление базы данных с помощью pg\_dump и pg\_restore.

Работа над итоговым заданием была разбита на несколько ключевых этапов:

1. Анализ предметной области и проектирование структуры данных  
   Определены сущности, их атрибуты, связи (один-ко-многим, многие-ко-многим), проанализированы требования к хранилищу информации.
2. Создание базы данных и таблиц с использованием SQL-команд CREATE DATABASE, CREATE TABLE реализована схема. Определены первичные и внешние ключи, типы данных, ограничения (CHECK, NOT NULL, UNIQUE).
3. Наполнение базы данных  
   С помощью команд INSERT INTO добавлены тестовые данные. Использовались также UPDATE, DELETE для демонстрации операций редактирования и удаления.
4. Формирование запросов  
   Написаны простые и сложные запросы, включая агрегатные функции (SUM, COUNT), фильтрацию, сортировку, группировку, подзапросы и объединения таблиц (JOIN всех видов).
5. Дополнительные функции  
   Созданы представления (VIEW) для удобной выборки данных, функции (FUNCTION) на языке SQL.
6. Администрирование и безопасность  
   Реализовано создание ролей и пользователей, предоставление прав доступа (GRANT, REVOKE). Выполнено резервное копирование с использованием pg\_dump, восстановление базы — через pg\_restore.

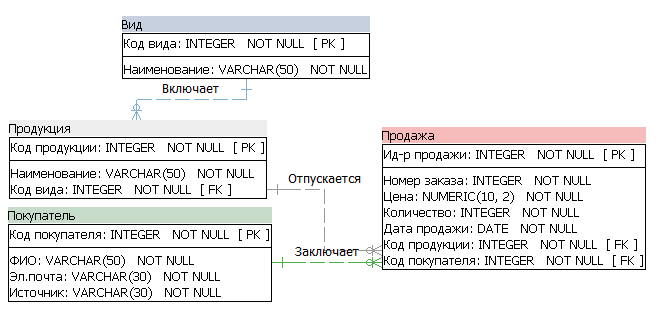
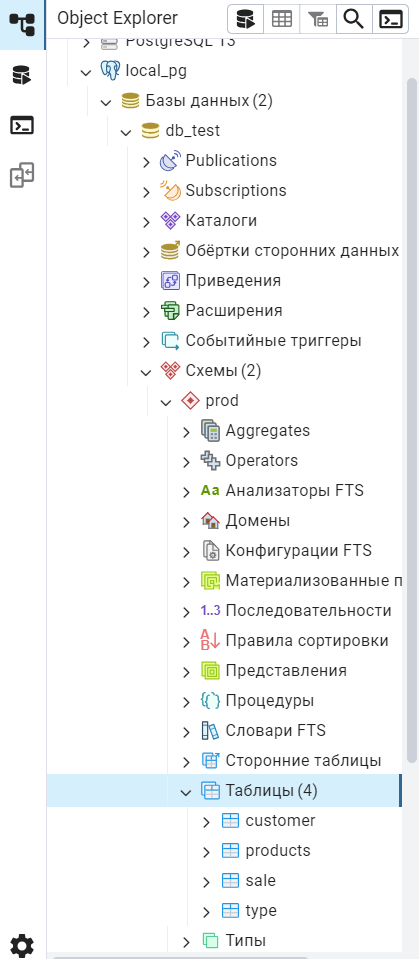


Рисунок 1. Логическая модель данных “Онлайн торговля”

Созданная схема (таблицы и другие объекты) в целевой БД отображены на рисунке 2. Наполнение таблиц на рисунке 3.



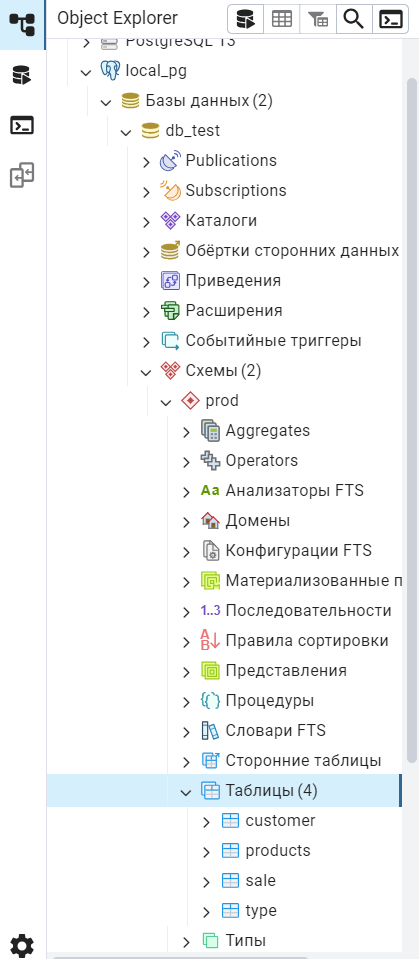


Рисунок 2. Таблицы базы данных

|  |  |
| --- | --- |
|  |  |
|  | |
|  | |

Рисунок 3. Таблицы, загруженные в базу данных

Следующее задание демонстрирует практику составления оператора запроса к таблицам базы данных, при помощи предикатов для отбора нужных данных в БД PostgreSQL.

Запрос 1: Выбрать всю продукцию из продажи с заданным id и датой. Вывести код и наименование продукции, цену, количество и стоимость каждой позиции.

*SELECT p.prod\_code, p.prod\_name, s.price, s.count,*

*CAST(s.price \* s.count AS NUMERIC(10, 2)) AS cost*

*FROM sale s*

*JOIN products p ON s.sale\_prod\_code = p.prod\_code*

*WHERE s.id = 11 AND s.date\_of\_sale = '2025-05-05';*

Запрос 2: Выбрать продукцию, проданную в августе 2025 года.

*SELECT p.prod\_code, p.prod\_name, s.price, s.count,*

*CAST(s.price \* s.count AS NUMERIC(10, 2)) AS cost*

*FROM sale s*

*JOIN products p ON s.sale\_prod\_code = p.prod\_code*

*WHERE s.date\_of\_sale BETWEEN '2025-08-01' AND '2025-08-31';*

Запрос 3: Вывести имена покупателей, которые покупали продукцию с кодом 3 или 4.

*SELECT fullname*

*FROM customer*

*WHERE buyer\_code IN (*

*SELECT sale\_buyer\_code*

*FROM sale*

*WHERE sale\_prod\_code IN (3, 4)*

*);*

Запрос 4: Вывести имена покупателей, у которых заполнен email.

*SELECT fullname*

*FROM customer*

*WHERE email IS NOT NULL OR email = '';*

Запрос 5: Вывести продажи в обратном хронологическом порядке.

*SELECT \* FROM sale ORDER BY date\_of\_sale DESC;*

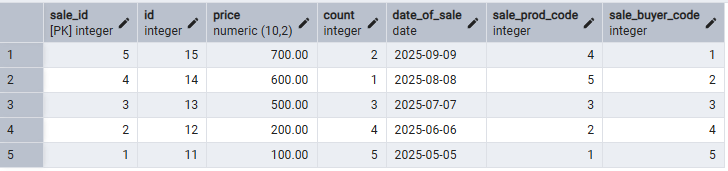


Рисунок 4. Продажи в обратном хронологическом порядке.

Следующее задание демонстрирует выполнение группировки данных и вычисление агрегатных функций в операторе запроса в БД PostgreSQL

Запрос 6: Определить суммарные объемы продажи по каждой продукции

*SELECT p.prod\_name, SUM(s.count) AS total\_quantity*

*FROM products p*

*JOIN sale s ON p.prod\_code = s.sale\_prod\_code*

*GROUP BY p.prod\_name;*

Запрос 7: Определить количество разных наименований продукции по каждому покупателю

*SELECT customer.fullname, COUNT(DISTINCT p.prod\_name) AS product\_count*

*FROM customer*

*JOIN sale s ON customer.buyer\_code = s.sale\_buyer\_code*

*JOIN products p ON s.sale\_prod\_code = p.prod\_code*

*GROUP BY customer.fullname;*

Запрос 8: Покупатели с общим объемом поставок > 1000

*SELECT customer.fullname, SUM(s.price \* s.count) AS total\_value*

*FROM customer*

*JOIN sale s ON customer.buyer\_code = s.sale\_buyer\_code*

*GROUP BY customer.fullname*

*HAVING SUM(s.price \* s.count) > 1000;*

Запрос 9: Продукция с минимальной поставкой > 2

*SELECT p.prod\_name, MIN(s.count) AS min\_supply*

*FROM products p*

*JOIN sale s ON p.prod\_code = s.sale\_prod\_code*

*GROUP BY p.prod\_name*

*HAVING MIN(s.count) > 2;*

Запрос 10: выполнить соединение таблиц путём внешнего(полного) соединения.

*WITH products (prod\_code, prod\_name, prod\_group) AS*

*(SELECT '010', 'Samsung', 'Стиральная машина'*

*UNION*

*SELECT '011', 'LG', 'Пылесос'*

*UNION*

*SELECT '012', 'Bork', 'Микроволновка'*

*UNION*

*SELECT '013', 'Bosch', 'Холодильник'*

*UNION*

*SELECT '014', 'Sony', 'Телевизор'),*

*sale (sale\_id, customer, prod\_code, count) AS*

*(SELECT 1, 'Сидоров Кондратий Викторович', '010', 1*

*UNION*

*SELECT 2, 'Копылов Яков Иринеевич', '011', 2*

*UNION*

*SELECT 3, 'Быков Варлам Валерьевич', '012', 4*

*UNION*

*SELECT 4, 'Фёдоров Игнатий Ярославович', NULL, 3*

*UNION*

*SELECT 5, NULL, '013', 1*

*UNION*

*SELECT 6, 'Белов Даниил Натанович', '014', 4)*

*SELECT p.prod\_name, s.customer, s.count*

*FROM products p FULL OUTER JOIN sale s ON p.prod\_code = s.prod\_code;*

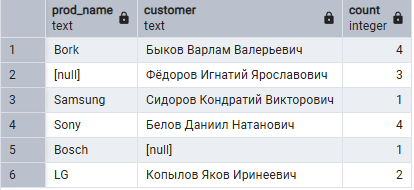


Рисунок 5. Внешнее соединение таблиц

Следующие задания нацелены на то, чтобы научиться использовать подзапросы в составе оператора запроса, получить практические навыки использования обобщенных табличных выражений.

Запрос 11. Получить список продукции, которая относится к тому же виду, что и продукция с кодом 2. Использовать предикат сравнения.

*SELECT prod\_name*

*FROM products*

*WHERE prod\_type\_code = (SELECT prod\_type\_code FROM products WHERE prod\_code = 2);*

Запрос 12. Получить список продукции, полученной покупателем Копылов ЯковИринеевич*.*

*SELECT prod\_name*

*FROM products*

*WHERE prod\_code IN (*

*SELECT DISTINCT prod\_code*

*FROM sale s JOIN products p ON s.sale\_prod\_code = p.prod\_code*

*WHERE s.sale\_buyer\_code = (*

*SELECT buyer\_code FROM customer WHERE fullname = 'Копылов Яков Иринеевич'));*

Запрос 13. Получить список продукции, полученной покупателем Копылов Яков Иринеевич. Использовать предикат EXISTS.

*SELECT prod\_name*

*FROM products*

*WHERE EXISTS (SELECT 1*

*FROM sale s JOIN products p ON s.sale\_prod\_code = p.prod\_code*

*WHERE s.sale\_buyer\_code = (SELECT buyer\_code FROM customer WHERE fullname = 'Копылов Яков Иринеевич')*

*AND products.prod\_code = s.sale\_prod\_code);*

*Запрос 14. Вывести информацию о продукции и количестве покупателей, которые ее купили.*

*SELECT prod\_name, (SELECT COUNT(DISTINCT s.sale\_buyer\_code)*

*FROM sale s*

*WHERE s.sale\_prod\_code = p.prod\_code) AS customer\_count*

*FROM products p;*

Таблицы для заданий 15-16.

*WITH customer (buyer\_code, fullname, email, source) AS (*

*VALUES (1, 'Сидоров Кондратий Викторович', 'tonaza-bubi62@mail.ru', 'Интернет'),*

*(2, 'Копылов Яков Иринеевич', 'fobogo\_yigu1@mail.ru', 'Знакомые'),*

*(3, 'Быков Варлам Валерьевич', 'jovo\_xaxara67@mail.ru', 'Интернет'),*

*(4, 'Федоров Игнатий Ярославович', 'seb\_iciwuri84@mail.ru', 'Интернет'),*

*(5, 'Белов Донат Натаннович', 'cey\_osacuva39@mail.ru', 'Журнал')*

*),*

*sale (sale\_id, price, count, sale\_buyer\_code) AS (*

*VALUES (1, 100.00, 5, 5),*

*(2, 200.00, 4, 4),*

*(3, 500.00, 3, 3),*

*(4, 600.00, 4, 2),*

*(5, 700.00, 2, 1),*

*(2, 200.00, 4, 4),*

*(3, 500.00, 3, 3),*

*(4, 600.00, 4, 2) )*

Задание 15. Вывести список покупателей (ФИО, email), у которых общая сумма покупок ниже средней суммы покупок по всем покупателям.

*SELECT c.fullname, c.email, CAST(SUM(s.price \* s.count) AS NUMERIC(10,2)) AS total\_spent*

*FROM customer c*

*JOIN sale s ON c.buyer\_code = s.sale\_buyer\_code*

*GROUP BY c.fullname, c.email*

*HAVING SUM(s.price \* s.count) < (SELECT AVG(total) FROM (SELECT SUM(price \* count) AS total FROM sale GROUP BY sale\_buyer\_code) AS avg\_total)*

*ORDER BY total\_spent;*

Задание 16. Вывести список покупателей (ФИО, email), у которых количество покупок больше среднего по всем покупателям.

*SELECT c.fullname, c.email*

*FROM customer c*

*JOIN sale s ON c.buyer\_code = s.sale\_buyer\_code*

*GROUP BY c.fullname, c.email*

*HAVING COUNT(s.sale\_id) < (SELECT AVG(purchase\_count) FROM (SELECT COUNT(sale\_id) AS purchase\_count FROM sale GROUP BY sale\_buyer\_code) AS avg\_purchases)*

*ORDER BY c.fullname;*

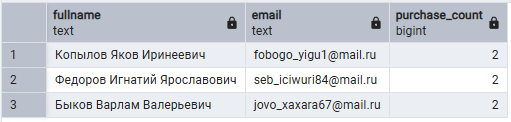


Рисунок 6. Результат выполнения задания 16

Сертификат окончания первого этапа олимпиалы IT-Планета 2025 в разделе СУБД Postgres Pro на рисунке 7.



Рис. 7. Сертификат окончания первого этапа олимпиалы IT-Планета 2025

**Заключение**

Учебная практика, посвящённая изучению PostgreSQL и языка SQL, позволила не только углубить теоретические знания, полученные в рамках учебной программы, но и приобрести важные практические навыки, необходимые для будущей профессиональной деятельности в сфере информационных технологий. В течение практики была проведена комплексная работа, охватывающая как основы проектирования и построения реляционных баз данных, так и продвинутые аспекты администрирования, оптимизации и обеспечения безопасности информационных систем.

Полученные знания и умения позволили не только понять логику работы современных реляционных СУБД, но и сформировать целостное представление о жизненном цикле данных в информационной системе — от этапа моделирования до сопровождения в эксплуатации. Выполнение итогового задания закрепило теоретические положения на практике, продемонстрировав, как важно учитывать все аспекты — от структуры запросов до резервного копирования и настройки прав доступа.

Пройденная практика способствовала формированию профессионального мышления, развитию системного подхода к решению задач, связанных с базами данных, а также повысила уверенность в работе с одной из самых популярных и надёжных СУБД на современном рынке. Полученные результаты создают прочную базу для дальнейшего обучения и профессионального роста в области информационных технологий.

Список литературы

1. Postgrespro.ru — официальный сайт PostgreSQL. Официальная документация. URL: postgrespro.ru/docs

2. Моргунов, Е. П. PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018. — 336 с.: ISBN 978-5-6041193-2-7 (Постгрес Профессиональный, электронное издание)

3. Лори Э., Коуэн Б. PostgreSQL. Основы разработки корпоративных приложений.Москва : ДМК Пресс, 2022. — 784 c.ISBN 978-5-97060-965-3

4. Левин А.Ш. Самоучитель SQL. Руководство для начинающих программистов и администраторов баз данных PostgreSQL.СПб.: Наука и техника, 2021. — 320 с.ISBN 978-5-94387-784-3