Министерство образования и науки Российской Федерации Российский государственный университет нефти и газа (национальный исследовательский университет)

имени И.М. Губкина

Факультет Автоматики и вычислительной техники Кафедра Автоматизированных систем управления

Отчёт по лабораторной работе №2

«УГЛУБЛЕННАЯ РАБОТА С SQL»

по дисциплине *Базы данных*

Выполнил: студент группы АС-23-04

Ханевский Ярослав

Проверили:

доцент кафедры АСУ, к.т.н. Волков Д.А.

ст. преп. кафедры АСУ Мухина А.Г.

Москва, 2025 г.

Ход работы:

1. Для своей таблицы выполнить запросы, содержащие блок WHERE с использованием различных операторов условий.

Создадим таблицу, включающую в себя информацию о сдаче экзаменов:

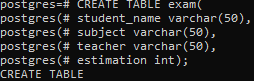


Рисунок . Создание таблицы

После заполнения таблицы получаем следующее:

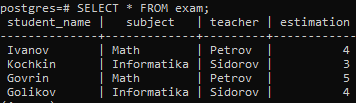


Рисунок . Заполненная таблица

Выполним запрос с оператором условия OR, выбирающий тех учеников, кто получил оценку «4» или чья фамилия – «Kochkin»:

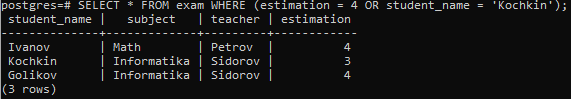


Рисунок . Оператор OR

Запрос, выбирающий данные по условию «учитель «Sidorov» и оценка выше 3», используя оператор AND:

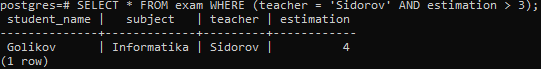


Рисунок . Оператор AND

Реализуем запрос с помощью оператора LIKE, ищущий среди всех имен учеников те, что заканчиваются на буквы «ov»:

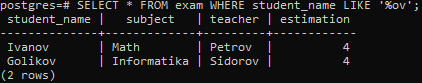


Рисунок . Оператор LIKE

Найдем тех учеников, которые сдали экзамен на 4 или 5 (оператор IN):

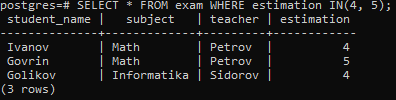


Рисунок . Оператор IN

Найдем тех учеников, которые сдали экзамен на оценку между 2 и 3 (оператор BETWEEN):

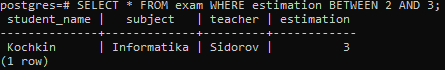


Рисунок . Оператор BETWEEN

1. Создать колонку с допустимым значением NULL. Выполнить запросы к таблице с использованием NULL. Обратить внимание на работу этого значения.

Создадим новую колонку с допустимым значением NULL (в PostgreSQL по умолчанию NULL является допустимым значением):

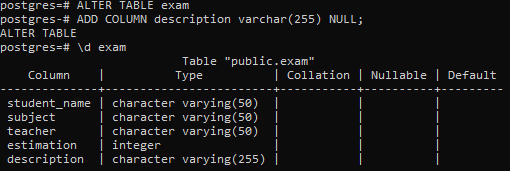


Рисунок . Новая колонка с допустимым значением NULL

Сделаем несколько запросов к таблице с использованием NULL; для начала добавим несколько строк:

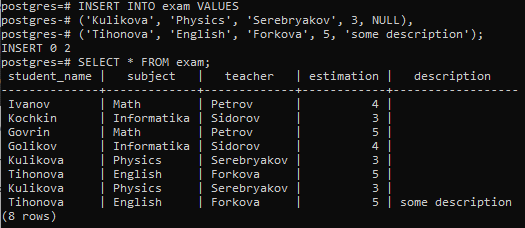


Рисунок . Добавление строки с пустым и с заполненным значением столбца

Выберем все строки из таблицы, в которых есть описание (IS NOT NULL):

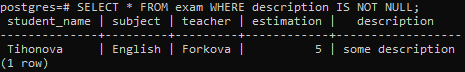


Рисунок . Использование SELECT с NOT NULL

1. Для своей таблицы выполнить запросы, содержащие блок GROUP BY с использованием различных функций агрегирования и фильтрации (HAVING):

Создадим и заполним новую таблицу, в которой указаны данные о проданных товарах:

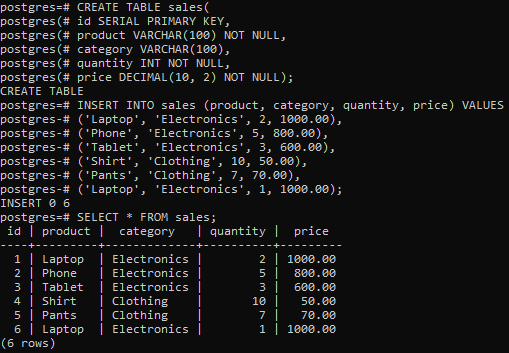
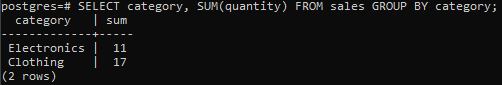


Рисунок . Новая таблица с проданными товарами

Используем функцию GROUP BY для подсчёта общего количества проданных товаров в каждой категории:



Применим функции агрегирования MAX и MIN для поиска максимальной и минимальной цены соответственно в каждой из групп категорий продуктов:

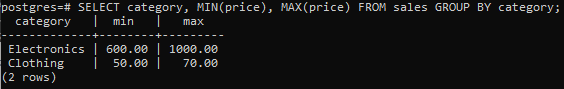


Рисунок . Поиск максимальной и минимальной цены в каждой группе

Теперь добавим фильтрацию (HAVING) таким образом, чтобы отображались те категории, общее количество проданных товаров в которых больше 10:

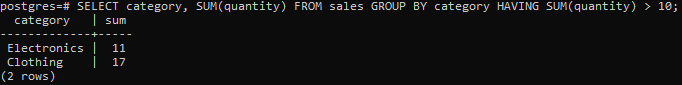


Рисунок . Использование фильтрации для поиска групп, сумма проданных товаров которых больше 10

Найдем среднее значение цены в каждой группе с помощью функции AVG:

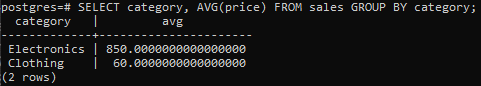


Рисунок . Среднее значение цены среди групп

1. Для своей таблицы выполнить запросы, содержащие блок ORDER BY.

Выполним сортировку (ORDER BY) для предыдущей таблицы по цене от большей к меньшей (DECS):

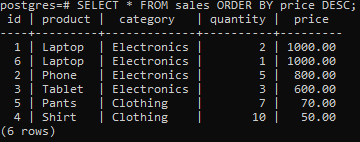


Рисунок . Сортировка данных по убыванию

Проведем сортировку по нескольким столбцам, теперь от меньших значений к большим:

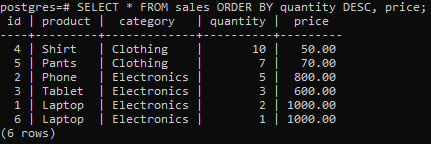


Рисунок . Сортировка нескольких столбцов по возрастанию

1. Для своей таблицы выполнить запросы, содержащие блок LIMIT.

Осуществим запрос, выбирающий первые 3 строки таблицы с помощью блока LIMIT:

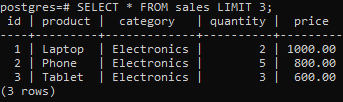


Рисунок . Использование блока LIMIT

Теперь пропустим первые 2 строки и выведем из таблицы 3 последующие строки:

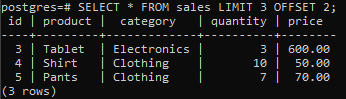


Рисунок . Блок LIMIT, пропускающий указанное количество строк

1. Добавить первичный ключ для таблицы. Добавить авто инкремент, удалить первичный ключ.

С помощью ADD CONSTRAINT добавляем первичный ключ (PRIMARY KEY), которым будут столбцы с названием и количеством товаров:

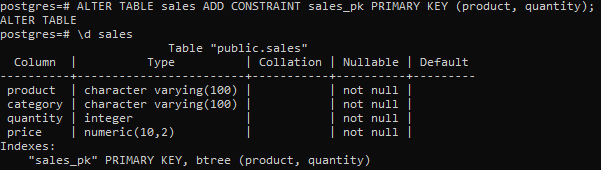


Рисунок . Добавление первичного ключа (PRIMARY KEY)

Добавляем авто инкремент (SERIAL), который обеспечивает автоматическую идентификацию новых строк:

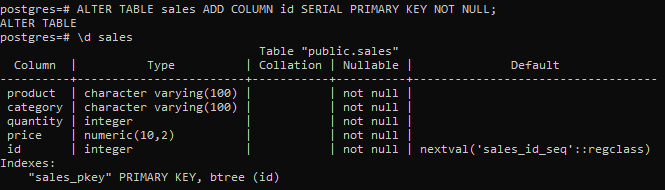


Рисунок . Добавление авто инкремента (SERIAL)

Теперь удаляем первичный ключ с помощью DROP CONSTRAINT:

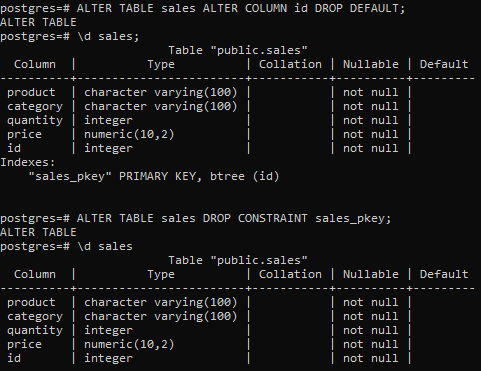


Рисунок . Удаление первичного ключа

1. Самостоятельно разобрать типы данных DATETIME, DATE и TIMESTAMP. Привести примеры.

Тип DATE используется для хранения только даты (год, месяц, день) без времени. Формат: YYYY-MM-DD (год-месяц-день).

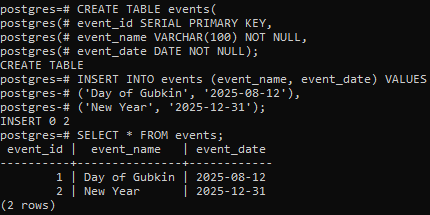


Рисунок . Тип DATE

Тип TIME используется для хранения только времени (часы, минуты, секунды) без даты. Формат: HH:MI:SS (часы:минуты:секунды).

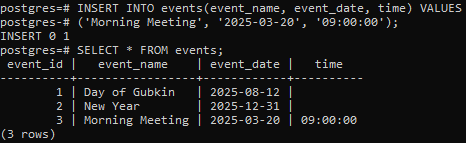


Рисунок . Тип Time

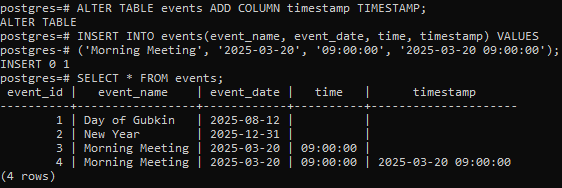
Тип TIMESTAMP используется для хранения даты и времени вместе. Формат: YYYY-MM-DD HH:MI:SS (год-месяц-день часы:минуты:секунды). 

Рисунок . Тип TIMESTAMP

1. Обратить внимание на приложение зарезервированных слов.
2. Для своих таблиц создать несколько внешних ключей, создать индекс, воспользоваться возможностями каскадного удаления и обновления.

Для начала создадим две таблицы и настроим внешний ключ:

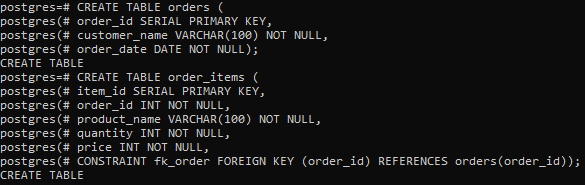


Рисунок . Создание двух таблиц и настройка внешнего ключа

Добавим данные в таблицы:

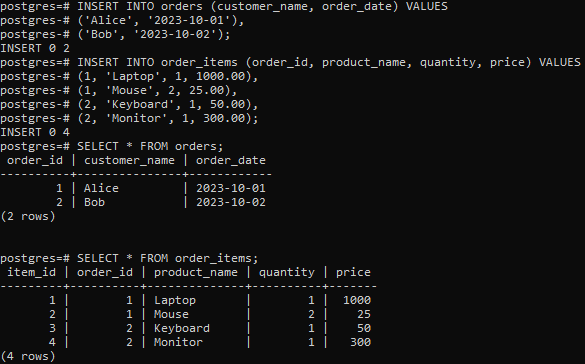


Рисунок . Добавление данных в таблицы

Попытка удалить данные из таблицы с заказами вызывает ошибку базы данных:



Рисунок . Ошибка при попытке удаления данных

Рассмотрим возможность каскадного удаления, то есть удаления всех связанных данных при удалении строки из таблицы.

Сначала удалим прошлый внешний ключ:

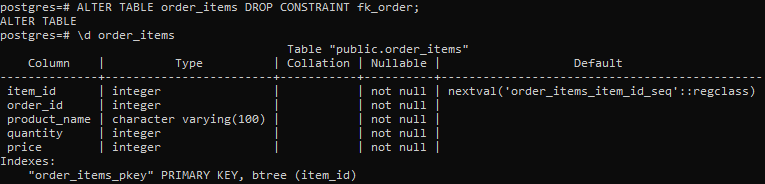


Рисунок . Удаление внешнего ключа

Добавляем возможность каскадного удаления ON DELETE и ON UPDATE:

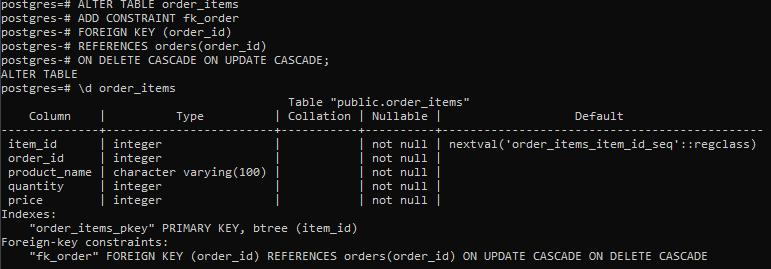


Рисунок . Создание внешнего ключа с возможностью каскадного удаления

Теперь удалим одну из строк таблицы «orders» – например, с порядковым номером 2:

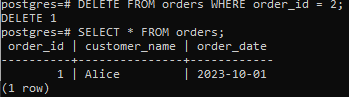


Рисунок . Удаление одной из строк «orders»

При этом удалятся все записи с order\_id = 2 в связанной дочерней таблице:

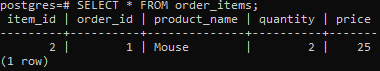


Рисунок . Проверка дочерней таблицы после удаления строки

Обновим порядковый номер строки:

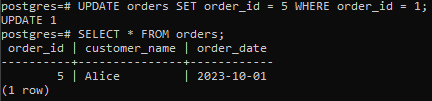


Рисунок . Обновление номера строки

Посмотрим изменения в дочерней таблице:

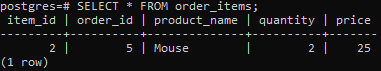


Рисунок . Изменения в дочерней таблице после обновления номера строки

Все секции с порядковым номером 1 изменились на 5.

Для ON DELETE и ON UPDATE при создании внешних ключей кроме CASCADE также доступны действия:

* RESTRICT – при попытке удаления или обновления родительской записи, содержащей связанные дочерние записи, вызовется ошибка, операция удаления или обновления не будет выполнена;
* SET NULL – при удалении или обновлении родительской записи все связанные дочерние записи будут обновлены так, чтобы столбец, содержащий внешний ключ в дочерней таблице, принял значение NULL;
* NO ACTION – равнозначно RESTRICT;
* SET DEFAULT – при удалении или обновлении родительской записи все связанные дочерние записи будут обновлены так, чтобы столбец, содержащий внешний ключ в дочерней таблице, принял значение, указанное по умолчанию.

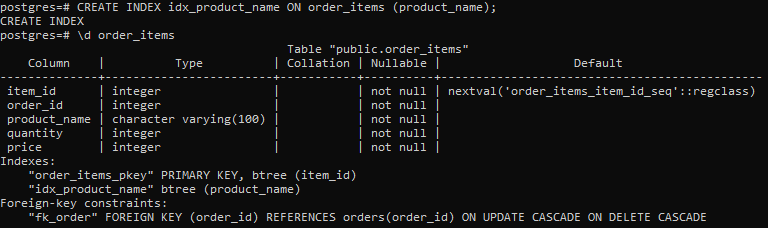
Для ускорения поиска по столбцу «product\_name» в таблице «order\_items» создадим индекс: 

Рисунок . Создание и проверка индекса

Индексы могут использоваться для поиска записей по уникальному идентификатору, для фильтрации записей по критериям, для сортировки данных, операций соединения и уникальных ограничений.

1. Для своих таблиц выполнить 10 разных запросов к нескольким таблицам, используя различные типы соединений.

Для демонстрации различных типов соединений создадим две таблицы: students (студенты) и courses (курсы):

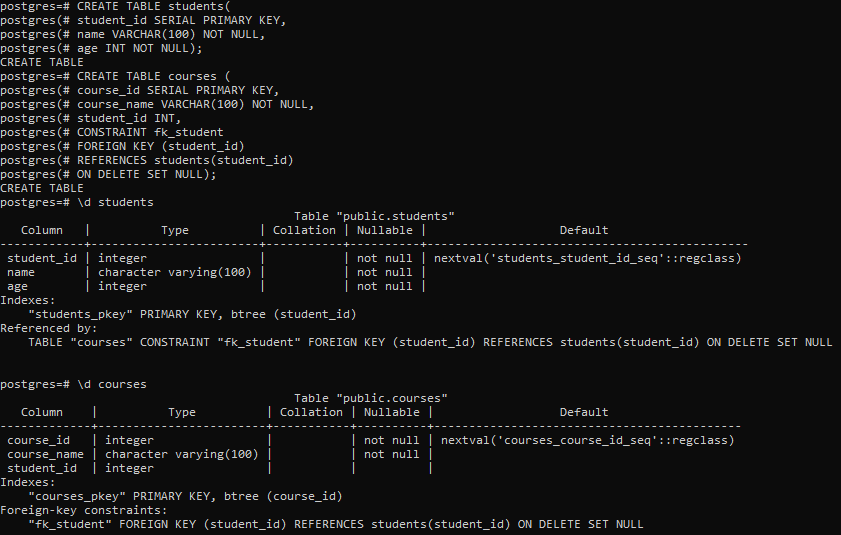


Рисунок . Создание таблиц для запросов с разными типами соединения

Добавим данные в таблицы:

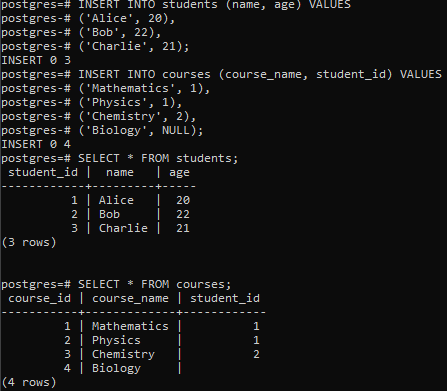


Рисунок . Добавление данных в таблицы

Осуществим декартово произведение, то есть все возможные перестановки строк из таблиц со студентами и предметами (CROSS JOIN):

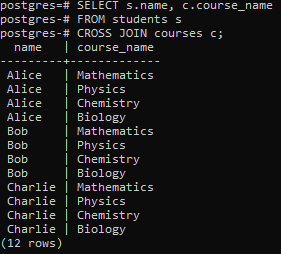


Рисунок . Декартово произведение таблиц

Объединим таблицы так, чтобы отображались только строки, где есть совпадения в обеих таблицах (INNER JOIN):

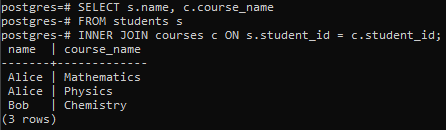


Рисунок . Внутреннее объединение

Объединим таблицы так, чтобы отображались все строки из левой таблицы и соответствующие строки из правой таблицы. Если совпадений нет, отобразится NULL для правой таблицы. (LEFT JOIN):

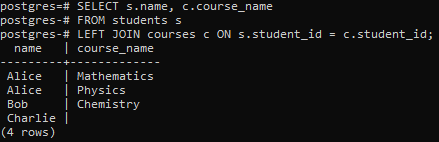


Рисунок . Левое объединение

Объединим таблицы так, чтобы отображались все строки из правой таблицы и соответствующие строки из левой таблицы. Если совпадений нет, отобразится NULL для левой таблицы. (RIGHT JOIN):

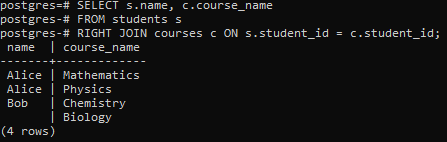


Рисунок . Правое объединение

Объединим таблицы так, чтобы отображались все строки из обеих таблиц. Если совпадений нет, отобразится NULL для недостающих данных. (FULL JOIN):

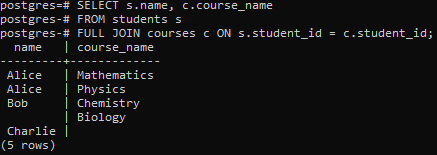


Рисунок . Полное объединение

1. Для своих таблиц выполнить 2 разных запроса к нескольким таблицам, используя оператор объединения множеств.

С помощью UNION и UNION ALL объединим таблицы:

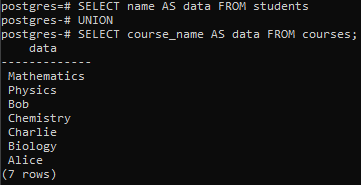


Рисунок . Объединение множеств с помощью UNION

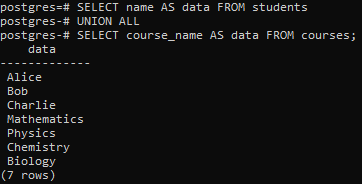


Рисунок . Объединение множеств с помощью UNION ALL

1. Используя официальную документацию PostgreSQL, выполнить произвольные запросы, содержащие функции для работы с данными:

Используем функции SQRT() (квадратный корень) и ROUND() (округление):

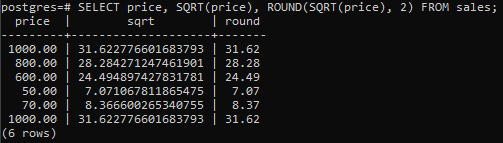


Рисунок . Вычисление и округления корня

Используем функции CONCAT() (объединение строк) и UPPER() (преобразование в верхний регистр):

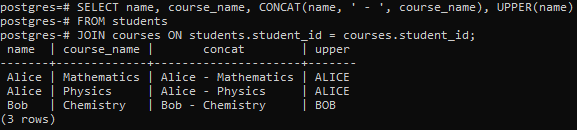


Рисунок . Объединение строк и перевод в верхний регистр