Министерство цифрового развития, связи и массовых коммуникаций РФ

Уральский технический институт связи и информатики (филиал) ФГБОУ ВО "Сибирский государственный университет телекоммуникаций и информатики" в г. Екатеринбурге

(УрТИСИ СибГУТИ)

**Казанцев М.Ю.**

**ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ**

Методические указания по выполнению практических работ

студентов очной и заочной формы обучения

«Информатика и вычислительная техника»

Методические указания для выполнения практической работы №6

# Екатеринбург 2024

**Практическая работа №6**

**«Оператор JOIN. Работа с временными таблицами»**

**Цель практической работы:**

Цель этой практической работы - изучить применение оператора JOIN в SQL для извлечения данных из нескольких связанных таблиц, научиться создавать временные таблицы и выполнять сложные SQL-запросы для анализа и обработки данных в базе данных PostgreSQL с использованием pgAdmin 4.

**Теоретический материал:**

Понимание JOIN и его различных типов — INNER JOIN, LEFT JOIN, RIGHT JOIN и FULL JOIN — позволит эффективно извлекать данные из связанных таблиц, что является важной частью работы с базами данных.

INNER JOIN используется для выборки записей, имеющих совпадения в обеих таблицах. Это наиболее распространенный тип соединения, который возвращает только те строки, у которых есть соответствующие записи в обеих соединяемых таблицах. Например, следующий запрос возвращает список студентов и названия курсов, на которые они записаны:

SQL

*SELECT Students.Name AS StudentName, Courses.CourseName*

*FROM Students*

*INNER JOIN Enrollments ON Students.StudentID = Enrollments.StudentID*

*INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID;*

LEFT JOIN возвращает все записи из левой таблицы и совпадающие записи из правой таблицы. Если совпадения нет, результат будет содержать NULL для правой таблицы. Этот вид соединения полезен, когда необходимо получить все записи из одной таблицы с дополнительной информацией из другой, даже если такой информации нет.

RIGHT JOIN работает аналогично LEFT JOIN, но возвращает все записи из правой таблицы и совпадающие записи из левой. Используется реже, но также может быть полезен в определенных ситуациях.

FULL JOIN объединяет результаты LEFT JOIN и RIGHT JOIN, возвращая все записи, когда есть совпадения в одной из таблиц. Если в одной из таблиц нет совпадения, результат содержит NULL.

Создание временных таблиц позволяет объединять данные для последующего анализа, не изменяя исходные таблицы. Временные таблицы существуют только в рамках текущей сессии и удаляются автоматически после завершения сессии. Это удобно для временной обработки данных, например, при подготовке отчетов:

SQL

*CREATE TEMPORARY TABLE StudentTeachers AS*

*SELECT Students.Name AS StudentName, Teachers.Name AS TeacherName, Courses.CourseName*

*FROM Students*

*INNER JOIN Enrollments ON Students.StudentID = Enrollments.StudentID*

*INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID*

*INNER JOIN Teachers ON Courses.CourseID = Teachers.CourseID;*

**Примеры полезных SQL запросов:**

1. Получение списка всех студентов и курсов, включая тех, кто не записан ни на один курс:

SQL

*SELECT Students.Name AS StudentName, Courses.CourseName*

*FROM Students*

*LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID*

*LEFT JOIN Courses ON Enrollments.CourseID = Courses.CourseID;*

2. Определение количества курсов, проводимых каждым преподавателем:

SQL

*SELECT Teachers.Name, COUNT(DISTINCT Courses.CourseID) AS CourseCount*

*FROM Teachers*

*LEFT JOIN Courses ON Teachers.CourseID = Courses.CourseID*

*GROUP BY Teachers.Name;*

3. Поиск факультетов, предлагающих более трех курсов:

SQL

*SELECT Departments.DepartmentName*

*FROM Departments*

*INNER JOIN Courses ON Departments.DepartmentID = Courses.DepartmentID*

*GROUP BY Departments.DepartmentName*

*HAVING COUNT(Courses.CourseID) > 3;*

4. Вывод студентов, которые обучаются только на одном факультете:

SQL

*SELECT Students.Name*

*FROM Students*

*INNER JOIN Enrollments ON Students.StudentID = Enrollments.StudentID*

*INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID*

*GROUP BY Students.Name*

*HAVING COUNT(DISTINCT Courses.DepartmentID) = 1;*

5. Получение информации о преподавателях и количестве студентов на их курсах:

SQL

*SELECT Teachers.Name, COUNT(Enrollments.StudentID) AS StudentCount*

*FROM Teachers*

*INNER JOIN Courses ON Teachers.CourseID = Courses.CourseID*

*LEFT JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID*

*GROUP BY Teachers.Name;*

6. Определение факультета с наибольшим количеством уникальных студентов:

SQL

*SELECT Departments.DepartmentName, COUNT(DISTINCT Enrollments.StudentID) AS UniqueStudentCount*

*FROM Departments*

*INNER JOIN Courses ON Departments.DepartmentID = Courses.DepartmentID*

*INNER JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID*

*GROUP BY Departments.DepartmentName*

*ORDER BY UniqueStudentCount DESC*

*LIMIT 1;*

**Задания:**

Создайте базу данных под названием SchoolDB и внутри нее создайте четыре таблицы: Students, Courses, Enrollments и Teachers. Каждая таблица должна содержать соответствующие поля и связи между ними. После этого заполните таблицы примерными данными. В таблице Students должны быть указаны студенты, в Courses — доступные курсы, в Enrollments — записи о зачислении студентов на курсы, а в Teachers — информация о преподавателях. В каждой таблице должно быть минимум по 20 записей.

Далее приступите к выполнению запросов, используя оператор INNER JOIN. Сначала получите список всех студентов и курсов, на которые они записаны. Это позволит понять, как работает INNER JOIN для выборки данных из связанных таблиц. Затем выполните запрос с использованием LEFT JOIN, чтобы извлечь список всех студентов и курсов, включая информацию о тех студентах, которые не записаны ни на один курс.

Создайте временную таблицу StudentTeachers, которая будет содержать данные о студентах, их курсах и преподавателях. Для этого используйте оператор INNER JOIN, чтобы объединить таблицы Students, Enrollments, Courses и Teachers. После создания временной таблицы выполните запрос, чтобы извлечь список студентов и их учителей.

Также найдите студентов, которые обучаются у преподавателя Ms. Johnson. Подсчитайте количество студентов, записанных на каждый курс, чтобы оценить популярность курсов. Также выведите список всех курсов, на которые не записан ни один студент, чтобы выявить курсы без студентов.

Подсказка по структуре БД:

В базе данных AdvancedSchoolDB предусмотрена таблица Students, которая содержит информацию о студентах. В этой таблице находятся три поля: StudentID, являющееся первичным ключом, Name, в котором хранится имя студента, и Age, отражающее возраст студента. Такая структура позволяет уникально идентифицировать каждого студента и хранить базовую информацию о нем.

Таблица Courses предназначена для хранения данных о курсах. Она включает поля CourseID, CourseName и DepartmentID. Поле CourseID служит первичным ключом, а DepartmentID является внешним ключом, ссылающимся на таблицу Departments. Это позволяет связать каждый курс с определенным факультетом, которому он принадлежит.

Таблица Enrollments играет роль связующей таблицы между студентами и курсами. Она содержит поля EnrollmentID, StudentID и CourseID. Поле EnrollmentID является первичным ключом таблицы, а StudentID и CourseID — внешними ключами, ссылающимися на таблицы Students и Courses соответственно. Эта таблица позволяет отслеживать, какие студенты записаны на какие курсы.

Таблица Teachers включает информацию о преподавателях и состоит из полей TeacherID, Name и CourseID. Поле TeacherID служит первичным ключом, а CourseID — внешним ключом, который связывает преподавателя с курсом, который он преподает. Таким образом, можно определить, кто из преподавателей ведет тот или иной курс.

Наконец, таблица Departments хранит данные о факультетах. Она содержит поля DepartmentID и DepartmentName, где DepartmentID является первичным ключом. Эта таблица позволяет управлять факультетами и связывать их с курсами через таблицу Courses, обеспечивая интеграцию между различными аспектами учебного процесса.

**Задание для самостоятельного выполнения:**

Создайте базу данных AdvancedSchoolDB и определите пять таблиц: Students, Courses, Enrollments, Teachers и Departments. Каждая таблица должна содержать такие поля и связи, которые позволят отразить структуру учебного заведения. Таблица Departments должна включать информацию о факультетах, к которым относятся курсы. Заполните все таблицы примерными данными, включая несколько факультетов, разнообразные курсы, студентов, преподавателей и записи об их зачислениях.

Выполните сложные SQL-запросы для извлечения данных. Используя INNER JOIN, LEFT JOIN и RIGHT JOIN, создайте запросы, которые показывают студентов и курсы, на которые они записаны, включая студентов, не записанных ни на один курс, а также курсы без студентов. Используйте FULL JOIN, чтобы собрать полную информацию о студентах и их курсах, включая тех, у кого нет совпадений в другой таблице.

Создайте временную таблицу DepartmentSummary, которая будет содержать информацию о каждом факультете, включая количество курсов, студентов и преподавателей на каждом факультете. Для этого объедините данные из всех пяти таблиц, используя соответствующие JOIN-ы. После создания временной таблицы выполните запросы для анализа данных, например, чтобы найти факультет с наибольшим количеством студентов или определить, какой преподаватель ведет больше всего курсов.

В качестве дополнительных заданий выполните следующие действия: найдите студентов, которые обучаются на курсах только одного факультета, и тех, кто записан на курсы нескольких факультетов. Определите преподавателя, у которого наибольшее количество студентов, и подсчитайте средний возраст студентов на каждом курсе.

**Контрольные вопросы:**

1. Какой запрос можно использовать, чтобы получить список всех студентов, которые записаны на курсы более чем одного факультета? Объясните логику вашего решения.

2. Как с помощью SQL-запроса определить, какие курсы преподаются на факультете с наибольшим количеством курсов? Опишите шаги, необходимые для выполнения этого задания.

3. Как создать запрос, который выводит преподавателей, имеющих наибольшее количество студентов, записанных на их курсы? Объясните, какие операторы JOIN и агрегатные функции вам понадобятся.

4. Каким образом вы можете использовать временные таблицы для анализа данных о количестве студентов на каждом факультете? Опишите процесс создания и использования таких таблиц.

5. Какой запрос позволит определить средний возраст студентов на каждом курсе? Опишите использование необходимых функций и операторов в SQL для достижения этого результата.

6. Сформулируйте запрос, который покажет факультет с наибольшим количеством студентов, используя данные из всех таблиц. Объясните, как вы будете объединять таблицы и какие функции использовать для подсчета.