Лабораторная Работа №8 SELECT запросы с соединением таблиц

Цель работы:

Освоить различные типы соединений таблиц (JOIN) в SQL: INNER JOIN, LEFT/RIGHT JOIN, FULL JOIN, CROSS JOIN. Научиться комбинировать данные из нескольких таблиц.

Программное обеспечение: PostgreSQL, DBeaver
Время: 2 академических часа

Подготовка базы данных

Создание таблиц с отношениями

```
SET search_path TO ваш_логин;

-- Создание таблиц для системы университета
CREATE TABLE faculties (
    faculty_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    dean VARCHAR(100)
);

CREATE TABLE departments (
    department_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    faculty_id INTEGER REFERENCES faculties(faculty_id),
    head VARCHAR(100)
);

CREATE TABLE teachers (
    teacher_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    department_id INTEGER REFERENCES departments(department_id),
    hire_date DATE,
```

```
    salary DECIMAL(10,2)
);

CREATE TABLE courses (
    course_id SERIAL PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    credits INTEGER NOT NULL,
    department_id INTEGER REFERENCES departments(department_id)
);

CREATE TABLE course_teachers (
    course_id INTEGER REFERENCES courses(course_id),
    teacher_id INTEGER REFERENCES teachers(teacher_id),
    PRIMARY KEY (course_id, teacher_id)
);

CREATE TABLE students (
    student_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    birth_date DATE,
    enrollment_year INTEGER
);

CREATE TABLE student_courses (
    student_id INTEGER REFERENCES students(student_id),
    course_id INTEGER REFERENCES courses(course_id),
    enrollment_date DATE DEFAULT CURRENT_DATE,
    grade DECIMAL(4,2),
    PRIMARY KEY (student_id, course_id)
);
```

Заполнение таблиц тестовыми данными

```
-- Факультеты
INSERT INTO faculties (name, dean) VALUES
('Информационных технологий', 'Иванов А.С.'),
('Экономический', 'Петрова М.В.'),
('Гуманитарных наук', 'Сидоров П.К.');

-- Кафедры
```

```sql
INSERT INTO departments (name, faculty_id, head) VALUES
('Программирования', 1, 'Козлов И.И.'),
('Сетевых технологий', 1, 'Новиков С.П.'),
('Бухгалтерского учета', 2, 'Морозова Е.А.'),
('Маркетинга', 2, 'Волков Д.Н.'),
('Иностранных языков', 3, 'Зайцева О.П.');

-- Преподаватели
INSERT INTO teachers (first_name, last_name, department_id, hire_date,
salary) VALUES
('Андрей', 'Смирнов', 1, '2018-03-15', 85000.00),
('Мария', 'Кузнецова', 1, '2020-09-10', 75000.00),
('Сергей', 'Попов', 2, '2017-11-20', 90000.00),
('Елена', 'Васильева', 2, '2019-06-05', 80000.00),
('Дмитрий', 'Петров', 3, '2015-08-30', 82000.00),
('Ольга', 'Соколова', 4, '2021-02-14', 78000.00),
('Алексей', 'Михайлов', 5, '2016-04-22', 79000.00),
('Наталья', 'Федорова', NULL, '2022-01-10', 70000.00); -- Преподаватель
без кафедры

-- Курсы
INSERT INTO courses (title, credits, department_id) VALUES
('Базы данных', 4, 1),
('Веб-программирование', 5, 1),
('Сетевые технологии', 4, 2),
('Бухгалтерский учет', 3, 3),
('Маркетинговые исследования', 4, 4),
('Английский язык', 2, 5),
('Французский язык', 2, 5),
('Управление проектами', 3, NULL); -- Курс без кафедры

-- Назначение преподавателей на курсы
INSERT INTO course_teachers (course_id, teacher_id) VALUES
(1, 1), (1, 2),  -- Два преподавателя на курс "Базы данных"
(2, 1),
(3, 3),
(4, 5),
(5, 6),
(6, 7), (7, 7), -- Один преподаватель на два курса
(8, 8); -- Курс без кафедры ведет преподаватель без кафедры
```

```sql
-- Студенты
INSERT INTO students (first_name, last_name, birth_date,
enrollment_year) VALUES
('Иван', 'Иванов', '2002-05-15', 2021),
('Петр', 'Петров', '2003-03-20', 2021),
('Мария', 'Сидорова', '2002-11-08', 2021),
('Анна', 'Козлова', '2003-07-30', 2022),
('Алексей', 'Николаев', '2004-01-25', 2022),
('Екатерина', 'Морозова', '2001-12-12', 2020),
('Дмитрий', 'Захаров', '2002-09-18', 2021),
('Ольга', 'Васнецова', '2003-04-05', 2022);

-- Запись студентов на курсы
INSERT INTO student_courses (student_id, course_id, enrollment_date,
grade) VALUES
(1, 1, '2023-09-01', 4.5),
(1, 2, '2023-09-01', 4.2),
(1, 6, '2023-09-01', 5.0),
(2, 1, '2023-09-01', 3.8),
(2, 3, '2023-09-01', 4.0),
(3, 4, '2023-09-01', 4.7),
(3, 5, '2023-09-01', 4.3),
(4, 6, '2023-09-01', 4.8),
(4, 7, '2023-09-01', 4.5),
(5, 2, '2023-09-01', 4.1),
(5, 3, '2023-09-01', 3.9),
(6, 1, '2023-09-01', 4.9),
(6, 4, '2023-09-01', 4.6);
-- Студент 7 не записан ни на один курс
-- Курс 8 не имеет записанных студентов
```

Задания по типам JOIN

1. INNER JOIN - внутреннее соединение

```sql
-- Пример: Получить список курсов с названием кафедры
SELECT c.title, d.name as department_name
FROM courses c
```

```
INNER JOIN departments d ON c.department_id = d.department_id
ORDER BY c.title;
```

Задания:

1. Получить список преподавателей с названиями их кафедр
2. Найти всех студентов и курсы, на которые они записаны (вывести имя студента и название курса)
3. Получить список кафедр с названиями факультетов
4. Найти преподавателей и курсы, которые они ведут
5. Получить студентов с оценками по курсам (только тех, у кого есть оценки)

2. LEFT JOIN - левое внешнее соединение

```
-- Пример: Все кафедры и их преподаватели (даже если на кафедре нет преподавателей)
SELECT d.name as department, t.first_name, t.last_name
FROM departments d
LEFT JOIN teachers t ON d.department_id = t.department_id
ORDER BY d.name, t.last_name;
```

Задания:

1. Показать все курсы и преподавателей, которые их ведут (даже если у курса нет преподавателя)
2. Вывести всех студентов и курсы, на которые они записаны (включая студентов без курсов)
3. Показать все кафедры и курсы, которые к ним относятся (включая кафедры без курсов)
4. Найти всех преподавателей и их кафедры (включая преподавателей без кафедры)
5. Вывести все факультеты и их кафедры (даже если у факультета нет кафедр)

3. RIGHT JOIN - правое внешнее соединение

```
-- Пример: Все преподаватели и их кафедры (включая преподавателей без кафедры)
SELECT t.first_name, t.last_name, d.name as department
FROM departments d
RIGHT JOIN teachers t ON d.department_id = t.department_id
ORDER BY d.name NULLS LAST;
```

Задания:

1. Показать все курсы и студентов, которые на них записаны (включая курсы без студентов)
2. Вывести всех преподавателей и курсы, которые они ведут (включая преподавателей без курсов)
3. Показать все кафедры и факультеты (включая кафедры без факультета)
4. Найти все оценки и студентов, которые их получили
5. Вывести все курсы и кафедры, которые их предлагают (включая курсы без кафедры)

4. FULL JOIN - полное внешнее соединение

```
-- Пример: Все курсы и все кафедры (соответствие где есть)
SELECT c.title as course, d.name as department
FROM courses c
FULL JOIN departments d ON c.department_id = d.department_id
ORDER BY c.title NULLS LAST, d.name NULLS LAST;
```

Задания:

1. Показать всех преподавателей и все кафедры (полное соответствие)
2. Вывести всех студентов и все курсы (полное соответствие через таблицу student_courses)
3. Показать все курсы и всех преподавателей (через таблицу course_teachers)
4. Найти все кафедры и все факультеты
5. Вывести всех студентов и все оценки (через таблицу student_courses)

5. CROSS JOIN - декартово произведение

```
-- Пример: Все возможные комбинации факультетов и кафедр
SELECT f.name as faculty, d.name as department
FROM faculties f
CROSS JOIN departments d
ORDER BY f.name, d.name;
```

Задания:

1. Создать все возможные пары студентов (сам с собой не включать)
2. Показать все возможные комбинации курсов и преподавателей
3. Создать расписание: все дни недели и все курсы

4. Все возможные комбинации факультетов и курсов
5. Декартово произведение студентов и курсов (все возможные записи)

6. Множественные JOIN - соединение 3+ таблиц

```sql
-- Пример: Студент -> Курс -> Преподаватель
SELECT s.first_name, s.last_name, c.title as course, t.first_name as
teacher_first, t.last_name as teacher_last
FROM students s
INNER JOIN student_courses sc ON s.student_id = sc.student_id
INNER JOIN courses c ON sc.course_id = c.course_id
INNER JOIN course_teachers ct ON c.course_id = ct.course_id
INNER JOIN teachers t ON ct.teacher_id = t.teacher_id
ORDER BY s.last_name, c.title;
```

Задания:

1. Получить цепочку: Факультет -> Кафедра -> Преподаватель -> Курс
2. Найти: Студент -> Курс -> Кафедра -> Факультет
3. Вывести: Преподаватель -> Курс -> Студенты (которые посещают этот курс)
4. Получить: Курс -> Преподаватели -> Их кафедры -> Факультеты
5. Найти: Студент -> Оценки -> Курсы -> Кафедры

7. JOIN с WHERE и ORDER BY

```sql
-- Пример: Курсы IT-факультета с количеством студентов
SELECT c.title, COUNT(sc.student_id) as student_count
FROM courses c
LEFT JOIN student_courses sc ON c.course_id = sc.course_id
INNER JOIN departments d ON c.department_id = d.department_id
INNER JOIN faculties f ON d.faculty_id = f.faculty_id
WHERE f.name = 'Информационных технологий'
GROUP BY c.course_id, c.title
ORDER BY student_count DESC;
```

Задания:

1. Найти всех студентов, записанных на курсы кафедры "Программирования"
2. Показать преподавателей, которые ведут более одного курса
3. Найти курсы, на которые записано более 2 студентов
4. Вывести студентов, у которых средняя оценка выше 4.5

5. Показать кафедры с количеством преподавателей и отсортировать по убыванию

8. Сложные комбинации JOIN

```
-- Пример: Студенты и их курсы с преподавателями и кафедрами
SELECT
    s.first_name || ' ' || s.last_name as student,
    c.title as course,
    t.first_name || ' ' || t.last_name as teacher,
    d.name as department,
    sc.grade
FROM students s
LEFT JOIN student_courses sc ON s.student_id = sc.student_id
LEFT JOIN courses c ON sc.course_id = c.course_id
LEFT JOIN course_teachers ct ON c.course_id = ct.course_id
LEFT JOIN teachers t ON ct.teacher_id = t.teacher_id
LEFT JOIN departments d ON c.department_id = d.department_id
ORDER BY s.last_name, c.title;
```

Задания:

1. Создать полный отчет по факультету: факультет -> кафедры -> преподаватели -> курсы -> студенты
2. Найти "одиноких" записей: преподавателей без кафедры, курсы без кафедры, студентов без курсов
3. Создать матрицу соответствия: все студенты против всех курсов (отметить записан/не записан)
4. Вывести иерархию: факультет -> кафедра -> преподаватель (в виде древовидной структуры)
5. Найти пересечения: курсы, которые посещают студенты из разных факультетов

Требования к отчету

1. **SQL-запросы** для каждого задания
2. **Результаты выполнения** (первые 10 строк каждого запроса)
3. **Объяснение** выбранного типа JOIN для каждого запроса
4. **Схема соединений** (какие таблицы и как соединялись)

Контрольные вопросы

1. **В чем разница между INNER JOIN и LEFT JOIN?**
2. **Когда использовать RIGHT JOIN вместо LEFT JOIN?**
3. **Что такое декартово произведение и когда его следует избегать?**
4. **Как работает FULL JOIN и в каких сценариях он полезен?**
5. **Какие проблемы могут возникнуть при множественных JOIN?**
6. **Как оптимизировать запросы с большим количеством JOIN?**
7. **Как обрабатываются NULL значения в разных типах JOIN?**