GeekBrains

Базы данных

Объединение JOIN

SQLite 3.31.1

На этом уроке

1. Узнаем об основных подходах к объединению таблиц посредством JOIN.
2. Научимся строить многотабличные запросы на JOIN.

Оглавление

Объединение таблиц через JOIN

На предыдущем занятии мы рассмотрели создание многотабличных запросов с применением вложенных запросов, а также посредством оператора UNION. Сегодня поговорим об объединении данных различных таблиц в рамках одного запроса через оператор объединения JOIN.

Типы объединения JOIN

Есть несколько типов такого объединения:

- перекрёстное объединение (CROSS JOIN);
- внутреннее объединение (INNER JOIN);
- левое внешнее объединение (LEFT JOIN);

- правое внешнее объединение (RIGHT JOIN);
- полное внешнее объединение (FULL JOIN).

Не все они реализованы в СУБД SQLite, но и ограниченной реализации будет достаточно для решения типовых задач по построению многотабличных запросов.

Перекрёстное объединение (CROSS JOIN)

Начнём с CROSS JOIN. Этот тип объединения применяется на практике крайне редко, но понимание того, как он работает, будет ключом к пониманию логики работы других типов объединения JOIN.

Синтаксически запрос с CROSS JOIN выглядит таким образом:

```
SELECT * FROM 'Левая таблица' CROSS JOIN 'Правая таблица';
```

Таблицу, которая в запросе указывается слева от ключевого слова JOIN, называется левой таблицей. Соответственно та, что указывается справа, называется правой таблицей. Применим CROSS JOIN для таблиц учеников и оценок на примере нашей базы данных students.db:

```
SELECT * FROM students CROSS JOIN grades;
```

Таблица students в этом запросе считается левой таблицей объединения, а таблица grades — правой. Посмотрим, что мы получим при выполнении такого кода:

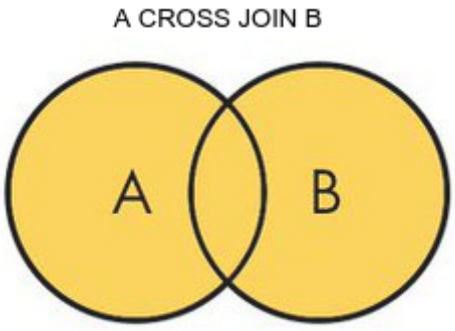
```
sqlite> .header on
sqlite> .mode column
sqlite> SELECT * FROM students CROSS JOIN grades;
id          surname     name        student_id  stream_id   grade
----------  ----------  ----------  ----------  ----------  -------------
1           Иванов      Игорь       1           1           4.9
1           Иванов      Игорь       2           1           5
1           Иванов      Игорь       1           3           5
1           Иванов      Игорь       1           2           4.9
2           Павлова     Анастасия   1           1           4.9
2           Павлова     Анастасия   2           1           5
2           Павлова     Анастасия   1           3           5
2           Павлова     Анастасия   1           2           4.9
3           Васильева   Ирина       1           1           4.9
3           Васильева   Ирина       2           1           5
3           Васильева   Ирина       1           3           5
3           Васильева   Ирина       1           2           4.9
sqlite>
```

В результате получилось 12 строк, почему именно такое количество? Если мы посмотрим в таблицы учеников и оценок, то увидим, что у нас в первой таблице три записи об учениках, а во второй — четыре записи об оценках учеников по итогам прохождения различных курсов:

```
sqlite> SELECT * FROM students;
id          surname     name
----------  ----------  ----------
1           Иванов      Игорь
2           Павлова     Анастасия
3           Васильева   Ирина
sqlite> SELECT * FROM grades;
student_id  stream_id   grade
----------  ----------  -------------
1           1           4.9
2           1           5
1           3           5
1           2           4.9
sqlite>
```

Двенадцать строк — это результат объединения данных каждой строки, относящихся к левой таблице (students), с данными каждой строки правой таблицы (grades), то есть 3 * 4 = 12. Логически такое объединение представляется диаграммой ниже.

Диаграмма 1. CROSS JOIN



При использовании CROSS JOIN мы получаем в результате объединение каждой строки левой таблицы A с каждой строкой правой таблицы B.

Практической ценности такие данные, как правило, не имеют, но понимание работы CROSS JOIN даст возможность перейти к рассмотрению других типов объединения JOIN.

Неявное объединение

Вариант написания, при котором мы убираем CROSS JOIN из команды и перечисляем в части FROM таблицы учеников и оценок через запятую, называется неявным объединением. То есть оператор JOIN указывается неявным образом.

```
SELECT * FROM students, grades;
```

Выполним запрос и проверим результат:

```
sqlite> SELECT * FROM students, grades;
id          surname     name        student_id  stream_id   grade
----------  ----------  ----------  ----------  ----------  -------------
1           Иванов      Игорь       1           1           4.9
1           Иванов      Игорь       2           1           5
1           Иванов      Игорь       1           3           5
1           Иванов      Игорь       1           2           4.9
2           Павлова     Анастасия   1           1           4.9
2           Павлова     Анастасия   2           1           5
2           Павлова     Анастасия   1           3           5
2           Павлова     Анастасия   1           2           4.9
3           Васильева   Ирина       1           1           4.9
3           Васильева   Ирина       2           1           5
3           Васильева   Ирина       1           3           5
3           Васильева   Ирина       1           2           4.9
sqlite>
```

Мы получили результат, полностью аналогичный предыдущему запросу с CROSS JOIN — 12 строк. Каждая строка первой таблицы объединяется с каждой строкой второй таблицы. То есть неявное объединение таблиц без условия, а именно без части запроса WHERE, работает как CROSS JOIN. Но мы можем определить условие, что нас интересуют только строки, где идентификатор ученика таблицы students совпадает со значением столбца student_id таблицы grades:

```
SELECT * FROM students, grades WHERE students.id = grades.student_id;
```

В таком случае стала доступна практически ценная информация — данные об оценках учеников по пройденным курсам:

```
sqlite> SELECT * FROM students, grades WHERE students.id = grades.student_id;
id          surname     name        student_id  stream_id   grade
----------  ----------  ----------  ----------  ----------  -------------
1           Иванов      Игорь       1           1           4.9
2           Павлова     Анастасия   2           1           5
1           Иванов      Игорь       1           3           5
```

```
1           Иванов      Игорь       1           2           4.9
sqlite>
```

Мы получили только те строки, где есть соответствие идентификатора пользователя id значению внешнего ключа student_id. В реальных проектах и литературе в запросах будет встречаться неявное объединение, но рекомендуется применять синтаксис с явным указанием ключевого слова JOIN.

Внутреннее объединение (INNER JOIN)

Перепишем предыдущий вариант запроса с явным использованием JOIN:

```
SELECT *
FROM students JOIN grades
WHERE students.id = grades.student_id;
```

В результате выполнения запроса мы получим аналогичный результат:

```
sqlite> SELECT *
   ...> FROM students JOIN grades
   ...> WHERE students.id = grades.student_id;
id          surname     name        student_id  course_id   grade
----------  ----------  ----------  ----------  ----------  -------------
1           Иванов      Игорь       1           1           4.9
2           Павлова     Анастасия   2           1           5
1           Иванов      Игорь       1           3           5
1           Иванов      Игорь       1           2           4.9
sqlite>
```

Такой тип объединения называется внутренним, так как в результате появятся те строки двух таблиц, между которыми найдено соответствие, определённое в части запроса WHERE. При использовании JOIN рекомендуется определять условие объединения таблиц, выбирая часть выражения ON:

```
SELECT *
FROM students JOIN grades
ON students.id = grades.student_id;
```

Хотя во многих случаях применение WHERE и ON в условии объединения даст одинаковый результат, СУБД обработает такие запросы различными способами. Рекомендуемый вариант — использование ключевого слова ON.

Внутреннее объединение JOIN также синтаксически определяется и как INNER JOIN, эти варианты взаимозаменяемы и дадут одинаковые результаты:

```
SELECT *
FROM students INNER JOIN grades
ON students.id = grades.student_id;
```

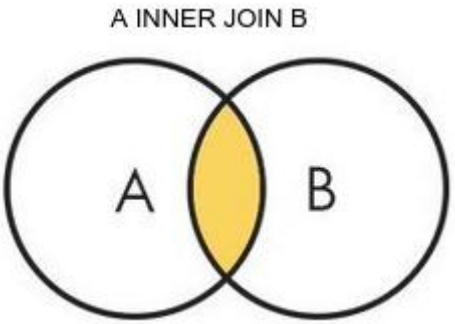
Проверим:

```
sqlite> SELECT *
   ...> FROM students INNER JOIN grades
   ...> ON students.id = grades.student_id;
id          surname     name        student_id  course_id   grade
----------  ----------  ----------  ----------  ----------  -------------
1           Иванов      Игорь       1           1           4.9
2           Павлова     Анастасия   2           1           5
1           Иванов      Игорь       1           3           5
1           Иванов      Игорь       1           2           4.9          sqlite>
```

Выбираются также определённые столбцы в части SELECT:

```
sqlite> SELECT surname, name, stream_id, grade
   ...> FROM students INNER JOIN grades
   ...> ON students.id = grades.student_id;
surname     name        stream_id   grade
----------  ----------  ----------  -------------
Иванов      Игорь       1           4.9
Павлова     Анастасия   1           5
Иванов      Игорь       3           5
Иванов      Игорь       2           4.9
sqlite>
```

Диаграмма 2. Внутреннее объединение INNER JOIN



При использовании внутреннего объединения INNER JOIN в результате мы получим только те данные из левой и правой таблиц, которые соответствуют условию, заданному в части ON запроса. На практике внутреннее объединение таблиц применяется наиболее часто.

Левое внешнее объединение (LEFT JOIN)

Рассмотрим самый используемый на практике вариант внешнего объединения — левое внешнее объединение. Базовый вариант запроса выглядит следующим образом:

```
SELECT *
FROM 'Левая таблица' LEFT JOIN 'Правая таблица'
ON 'Условие объединения';
```

Применим LEFT JOIN для объединения таблиц учеников и оценок:

```
SELECT *
FROM students LEFT JOIN grades
ON students.id = grades.student_id;
```

Получим следующий результат:

```
sqlite> SELECT *
   ...> FROM students LEFT JOIN grades
   ...> ON students.id = grades.student_id;
id          surname     name        student_id  course_id   grade
----------  ----------  ----------  ----------  ----------  -------------
1           Иванов      Игорь       1           1           4.9
1           Иванов      Игорь       1           2           4.9
1           Иванов      Игорь       1           3           5
2           Павлова     Анастасия   2           1           5
sqlite>
```

Мы получили тот же результат, что и при использовании INNER JOIN в примере выше — четыре строки, которые показывают список учеников с оценками. Тем не менее внутреннее и внешнее объединения работают различным образом, но чтобы это увидеть, надо добавить в базу данных нового ученика. Сделаем это посредством команды INSERT:

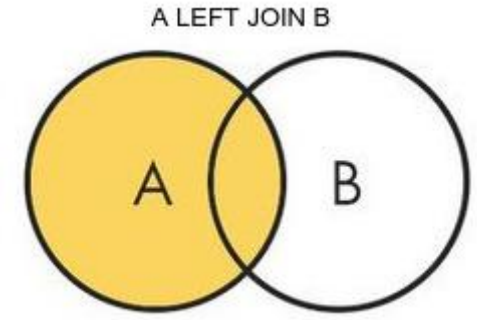
```
sqlite> INSERT  INTO students (surname, name) VALUES ('Васильева', 'Ирина');
sqlite> SELECT * FROM students;
id          surname     name
----------  ----------  ----------
1           Иванов      Игорь
2           Павлова     Анастасия
3           Васильева   Ирина
sqlite>
```

Мы добавили ученицу Васильеву Ирину, и у неё пока нет записей в таблице оценок. То есть она не закончила пока ни одного курса. Повторим наш запрос с LEFT JOIN:

```
sqlite> SELECT *
   ...> FROM students LEFT JOIN grades
   ...> ON students.id = grades.student_id;
id          surname     name        student_id  course_id   grade
----------  ----------  ----------  ----------  ----------  -------------
1           Иванов      Игорь       1           1           4.9
1           Иванов      Игорь       1           2           4.9
1           Иванов      Игорь       1           3           5
2           Павлова     Анастасия   2           1           5
3           Васильева   Ирина
sqlite>
```

Результат запроса изменился, теперь мы получили в отчёт всех учеников, но данные в столбцах из таблицы оценок для Ирины Васильевой пустые. В этом состоит основное отличие внешнего объединения от внутреннего — при использовании левого внешнего объединения в отчёте окажутся все записи из левой таблицы (students), независимо от того, будет ли найдено соответствие с правой таблицей (grades) по условию объединения в части запроса ON. Такая логика запроса используется, например, если нам по условию надо вывести в отчёт всех учеников, в том числе тех, кто пока не закончил ни одного курса. Внутреннее объединение INNER JOIN в этом случае выдаст только тех учеников, у кого есть записи в таблице оценок.

Диаграмма 3. Левое внешнее объединение LEFT JOIN



При использовании левого внешнего объединения LEFT JOIN мы получим все строки из левой таблицы A. Из правой таблицы B станут доступны данные, если найдётся соответствие по условию объединения ON, и появятся пустые значения для тех строк, по которым соответствия найдено не будет.

Правое внешнее объединение (RIGHT JOIN)

Чтобы применить правое внешнее объединение, используем синтаксис RIGHT JOIN:

```
SELECT *
FROM 'Левая таблица' RIGHT JOIN 'Правая таблица'
ON 'Условие объединения';
```

Используем такой тип объединения для таблиц учеников и оценок:

```
SELECT *
FROM students RIGHT JOIN grades
ON students.id = grades.student_id;
```

Получим следующий результат:

```
sqlite> SELECT *
   ...> FROM students RIGHT JOIN grades
   ...> ON students.id = grades.student_id;
Error: RIGHT and FULL OUTER JOINs are not currently supported
sqlite>
```

Сообщение говорит, что правое и полное внешние объединения не поддерживаются в SQLite. Но на самом деле это небольшая проблема, так как любой запрос с RIGHT JOIN переписывается с использованием LEFT JOIN, просто надо поменять местами левую и правую таблицы. Запрос, который реализует ту же логику, что и запрос выше, но использующий LEFT JOIN, выглядит так:

```
SELECT *
FROM grades LEFT JOIN students
ON students.id = grades.student_id;
```

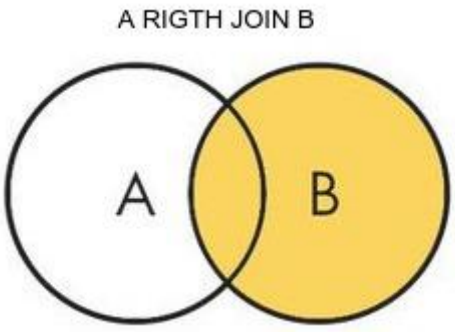
Мы поменяли RIGHT на LEFT, а также переставили местами таблицы учеников и оценок.

```
sqlite> SELECT *
   ...> FROM grades LEFT JOIN students
   ...> ON students.id = grades.student_id;
student_id  stream_id   grade          id          surname     name
----------  ----------  -------------  ----------  ----------  ----------
1           1           4.9            1           Иванов      Игорь
2           1           5              2           Павлова     Анастасия
1           3           5              1           Иванов      Игорь
1           2           4.9            1           Иванов      Игорь
sqlite>
```

Стали доступны четыре записи об оценках и соответствующие им данные из таблицы учеников. Такой же результат мы бы получили, если использовали в запросе объединение students RIGHT JOIN

grades, например, при работе в другой СУБД. В этом случае те же данные появились и при использовании INNER JOIN, потому что записи об оценках не имеют смысла, если в них не указывается, какому ученику выставлена оценка — строки в grades не имеют смысла без student_id.

Диаграмма 4. Правое внешнее объединение RIGHT JOIN



При использовании правого внешнего объединения RIGHT JOIN мы получим все строки из правой таблицы B. Из левой таблицы A станут доступны данные, если найдётся соответствие по условию объединения ON, и появятся пустые значения для строк, по которым соответствия найдено не будет. Для любого запроса можно менять тип объединения с LEFT на RIGHT и обратно, меняя местами левую и правую таблицы в объединении.

Полное внешнее объединение (FULL JOIN)

Третий тип внешнего объединения — FULL JOIN. Его работа рассматривается как одновременное применение левого и правого внешних объединений. Полное внешнее объединение не поддерживается во многих СУБД, включая SQLite, и на практике применяется редко.

Варианты полного синтаксиса для внешних объединений

Для внешних объединений применяется полный синтаксис с использованием ключевого слова OUTER — LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN. Но так как левое, правое и полное объединения всегда внешние, они никак не могут быть другими, то в большинстве случаев OUTER при написании команд опускается.

Пример команды с использованием полного синтаксиса:

```
SELECT *
FROM students LEFT OUTER JOIN grades
ON students.id = grades.student_id;
```

Пример построения запросов с более сложной логикой

Посредством объединения JOIN строятся сложные многотабличные запросы. Рассмотрим пример, где требуется по идентификатору ученика получить его фамилию, имя, пройденные курсы с номером потока и оценкой. Запрос для решения такой задачи выглядит так:

```
SELECT
  students.surname,
  students.name,
  courses.name AS course_name,
  streams.number AS stream_number,
  grades.grade
FROM students
  INNER JOIN grades
    ON students.id = grades.student_id
  INNER JOIN streams
    ON grades.stream_id = streams.id
  INNER JOIN courses
    ON streams.course_id = courses.id
WHERE students.id = 1;
```

Рассмотрим этот запрос более детально. Требуемая информация содержится в четырёх таблицах — учеников, оценок, потоков и курсов, то есть students, grades, streams, courses. Поэтому нам надо включить в объединение все эти таблицы.

То, какие типы объединения использовать для каждой таблицы, будет зависеть от логики запроса и порядка включения таблиц в объединение. В нашем примере достаточно использовать внутреннее объединение INNER JOIN, так как нас интересуют только данные, которые появятся после нахождения соответствия между записями таблиц учеников и оценок.

Составим объединение, используя в качестве условия в части ON равенство ‘внешний ключ’ = ‘первичный ключ’ для соответствующих таблиц. Хотя объединять таблицы можно не только по столбцам внешних ключей, на практике воспользуемся именно этим способом объединения, так как назначение внешнего ключа состоит в том, чтобы формальным образом определить связь между таблицами.

```
SELECT
  students.surname,
  students.name,
  courses.name AS course_name,
  streams.number AS stream_number,
  grades.grade
FROM students
  INNER JOIN grades
    ON students.id = grades.student_id
  INNER JOIN streams
    ON grades.stream_id = streams.id
```

```
  INNER JOIN courses
    ON streams.course_id = courses.id
WHERE students.id = 1;
```

В части SELECT запроса перечислим имена столбцов, данные которых надо получить, и дадим некоторым из них алиасы:

```
SELECT
  students.surname,
  students.name,
  courses.name AS course_name,
  streams.number AS stream_number,
  grades.grade
FROM students
  INNER JOIN grades
    ON students.id = grades.student_id
  INNER JOIN streams
    ON grades.stream_id = streams.id
  INNER JOIN courses
    ON streams.course_id = courses.id
WHERE students.id = 1;
```

Далее определим фильтр в части запроса WHERE, где укажем, что нас интересуют данные конкретного ученика с идентификатором 1:

```
SELECT
  students.surname,
  students.name,
  courses.name AS course_name,
  streams.number AS stream_number,
  grades.grade
FROM students
  INNER JOIN grades
    ON students.id = grades.student_id
  INNER JOIN streams
    ON grades.stream_id = streams.id
  INNER JOIN courses
    ON streams.course_id = courses.id
WHERE students.id = 1;
```

Теперь запрос готов, проверим его работу:

```
sqlite> SELECT
   ...>   students.surname,
   ...>   students.name,
   ...>   courses.name AS course_name,
```

```
   ...>   streams.number AS stream_number,
   ...>   grades.grade
   ...> FROM students
   ...>   INNER JOIN grades
   ...>     ON students.id = grades.student_id
   ...>   INNER JOIN streams
   ...>     ON grades.stream_id = streams.id
   ...>   INNER JOIN courses
   ...>     ON streams.course_id = courses.id
   ...> WHERE students.id = 1;
surname     name        course_name              stream_number  grade
----------  ----------  -----------------------  -------------  -------------
Иванов      Игорь       Linux. Рабочая станция   45             4.9
Иванов      Игорь       Основы Python            48             4.9
Иванов      Игорь       Базы данных              54             5
sqlite>
```

Мы видим, что запрос отработал корректно, стали доступны данные по ученику **Игорь Иванов**.

Важно! Выбор типа объединения важен для правильного решения поставленной задачи. Например, если понадобится вывести всех учеников с оценками либо без них, то в этом случае надо использовать левое внешнее объединение. При использовании внутреннего объединения в отчёт не попадут ученики, которые ещё не закончили ни одного курса — в нашем случае это Васильева Ирина:

```
SELECT
  students.surname,
  students.name,
  courses.name AS course_name,
  streams.number AS stream_number,
  grades.grade
FROM students
  LEFT JOIN grades
    ON students.id = grades.student_id
  LEFT JOIN streams
    ON grades.stream_id = streams.id
  LEFT JOIN courses
    ON streams.course_id = courses.id;
```

Выполним запрос и убедимся, что все пользователи попали в выборку данных:

```
sqlite> SELECT
   ...>   students.surname,
   ...>   students.name,
   ...>   courses.name AS course_name,
   ...>   streams.number AS stream_number,
   ...>   grades.grade
   ...> FROM students
   ...>   LEFT JOIN grades
```

```
   ...>     ON students.id = grades.student_id
   ...>   LEFT JOIN streams
   ...>     ON grades.stream_id = streams.id
   ...>   LEFT JOIN courses
   ...>     ON streams.course_id = courses.id;
surname     name        course_name             stream_number  grade
----------  ----------  ----------------------  -------------  -------------
Иванов      Игорь       Linux. Рабочая станция  45             4.9
Иванов      Игорь       Основы Python           48             4.9
Иванов      Игорь       Базы данных             54             5
Павлова     Анастасия   Linux. Рабочая станция  45             5
Васильева   Ирина
sqlite>
```

Практическое задание

Работаем с базой данных учителей teachers.db. Для каждого из заданий требуется создать запрос, сдать надо только код запросов в текстовом файле. Для решений воспользуйтесь объединением JOIN, не используйте вложенные запросы и объединение UNION.

1. Покажите информацию по потокам. В отчет выведите номер потока, название курса и дату начала занятий.

2. Найдите общее количество учеников для каждого курса. В отчёт выведите название курса и количество учеников по всем потокам курса.

3. Для всех учителей найдите среднюю оценку по всем проведённым потокам. В отчёт выведите идентификатор, фамилию и имя учителя, среднюю оценку по всем проведенным потокам. Важно чтобы учителя, у которых не было потоков, также попали в выборку.

4. **Дополнительное задание.** Для каждого преподавателя выведите имя, фамилию, минимальное значение успеваемости по всем потокам преподавателя, название курса, который соответствует потоку с минимальным значением успеваемости, максимальное значение успеваемости по всем потокам преподавателя, название курса, соответствующий потоку с максимальным значением успеваемости. В дополнительном задании допускается применение вложенных запросов.

Глоссарий

Перекрёстное объединение — CROSS JOIN — тип объединения, при котором каждая строка левой таблицы объединяется с каждой строкой правой таблицы.

Внутреннее объединение — JOIN, INNER JOIN — тип объединения, где в результате появляются только те данные из левой и правой таблиц, которые соответствуют условию, заданному в части ON запроса.

Левое внешнее объединение — LEFT JOIN, LEFT OUTER JOIN — тип объединения, при использовании которого в результате появляются все строки из левой таблицы. Из правой таблицы выделяются данные, если появляется соответствие по условию объединения ON, и выявляются пустые значения для тех строк, где соответствие не обнаруживается.

Правое внешнее объединение — RIGHT JOIN, RIGHT OUTER JOIN — тип объединения, при использовании которого в результате выделяются все строки из правой таблицы. Из левой таблицы появляются данные, если обнаруживается соответствие по условию объединения ON, и выводятся пустые значения для тех строк, по которым соответствия не выявляются.

Полное внутреннее объединение — FULL JOIN, FULL OUTER JOIN — тип объединения, который рассматривается как одновременное применение левого и правого внешних объединений, то есть в отчёт попадут все записи как из левой, так и из правой таблицы, удовлетворяющих условию выборки.

Дополнительные материалы

1. Статья о Joins SQLite

Используемые источники

1. Документация SQLite, SELECT.