GeekBrains

Базы данных

Вложенные запросы и объединение UNION

SQLite 3.31.1

На этом уроке

1. Научимся обращаться к данным нескольких таблиц в одном запросе, используя вложенные запросы.
2. Узнаем, как объединять данные различных таблиц посредством оператора UNION.

Оглавление

Введение в многотабличные запросы

На примере базы данных students.db рассмотрим одну задачу. Например, для ученика **Игорь Иванов** надо вывести отчёт, состоящий из таких столбцов: фамилия ученика, имя ученика, название курса, средняя оценка за практические задания курса.

Используя изученные приёмы, получим требуемую информацию из таблицы оценок grades. Посмотрим на её содержимое:

```
sqlite> .header on
sqlite> .mode column
sqlite> SELECT * FROM grades;
student_id  stream_id   grade
----------  ----------  -------------
1           1           4.9
2           1           5
1           3           5
1           2           4.9
sqlite>
```

Чтобы получить оценки ученика, надо узнать его идентификатор. Идентификатор ученика мы получим, выполнив запрос к таблице учеников students:

```
sqlite> SELECT id FROM students WHERE surname = 'Иванов' AND name = 'Игорь';
id
----------
1
sqlite>
```

Зная идентификатор ученика, найдём его оценки:

```
sqlite> SELECT * FROM grades WHERE student_id = 1;
student_id  stream_id   grade
----------  ----------  -------------
1           1           4.9
1           2           4.9
1           3           5
sqlite>
```

Мы получили оценки, но пока непонятно, за какие именно курсы. Нам известны только идентификаторы потоков (1, 2, 3). Чтобы определить соответствие между потоками и курсами, отправим запрос к таблице потоков streams:

```
sqlite> SELECT id, course_id FROM streams WHERE id IN (1, 2, 3);
id          course_id
----------  ----------
1           2
2           3
3           1
sqlite>
```

И, зная идентификаторы курсов, найдём их названия из таблицы курсов аналогичным способом:

```
sqlite> .mode list
sqlite> SELECT id, name FROM courses WHERE id IN (2, 3, 1);
id|name
1|Базы данных
2|Linux. Рабочая станция
3|Основы Python
sqlite>
```

С учётом этих данных определим, что Игорь Иванов получил оценку 4.9 за курс по Linux, 4.9 за курс по основам Python и 5 баллов по курсу «Базы данных».

Требуемую информацию мы получили, но для этого пришлось выполнить несколько запросов и свести полученную информацию самостоятельно.

Чтобы получить всю информацию сразу, надо создать так называемый многотабличный запрос. Это запрос, где выполняется выборка из нескольких таблиц. Для этого используются разные способы. Сегодня мы рассмотрим два — использование вложенных запросов и объединение запросов посредством оператора UNION.

Вложенные запросы

Чтобы уверенно составлять многотабличные запросы, требуется некоторый опыт. Новичкам рекомендуется метод поэтапного построения запроса с использованием так называемых заглушек. Рассмотрим применение такого подхода на примере нашего предыдущего задания — для ученика Игоря Иванова надо вывести отчёт, который состоит из таких столбцов: фамилия ученика, имя ученика, название курса, средняя оценка за практические задания курса.

Оценки находятся в таблице grades. Возьмём из неё значения оценок, а остальные значения временно заменим заглушками — строками ‘student_surname’, ‘student_name’ и ‘course_name’. На их месте выведем те значения, которых нет в таблице оценок. Выглядеть наш запрос будет следующим образом:

```
sqlite> .mode column
sqlite> SELECT 'student_surname', 'student_name', 'course_name', grade FROM
grades;
'student_surname'  'student_name'  'course_name'  grade
-----------------  --------------  -------------  -------------
student_surname    student_name    course_name    4.9
student_surname    student_name    course_name    5
student_surname    student_name    course_name    5
student_surname    student_name    course_name    4.9
sqlite>
```

Пока это не то, что нам подходит. Однако уже есть основа для дальнейшей работы. Следующий шаг — замена заглушек на код, посредством которого мы получим требуемые значения. Чтобы удобнее работать с кодом, выделим для каждого значения отдельную строку:

```
SELECT
  'student_surname',
  'student_name',
  'course_name',
  grade
FROM grades;
```

Получим фамилию и имя ученика в запросе к таблице students по идентификатору ученика student_id. Поместим запросы для фамилии и имени вместо заглушек. Для отделения этих вспомогательных запросов от основного запроса воспользуемся скобками. А чтобы улучшить формат вывода, относящийся к значениям результатов вспомогательных запросов, определим алиасы:

```
SELECT
 (SELECT surname FROM students WHERE id = student_id) AS student_name,
 (SELECT name FROM students WHERE id = student_id) AS student_surname,
  'course_name',
  grade
FROM grades;
```

В этом варианте команды мы применили вложенные запросы для подстановки значений фамилии и имени пользователя. Проверим результат выборки:

```
sqlite> SELECT
   ...>  (SELECT surname FROM students WHERE id = student_id) AS student_name,
   ...>  (SELECT name FROM students WHERE id = student_id) AS student_surname,
   ...>   'course_name',
   ...>   grade
   ...> FROM grades;
student_name  student_surname  'course_name'  grade
------------  ---------------  -------------  -------------
Иванов        Игорь            course_name    4.9
Павлова       Анастасия        course_name    5
Иванов        Игорь            course_name    5
Иванов        Игорь            course_name    4.9
sqlite>
```

Чтобы получить название курса на месте заглушки ‘course_name’, когда в таблице оценок есть только идентификатор потока stream_id, надо:

- сначала найти подходящий поток в таблице streams;
- затем по идентификатору курса course_id в записи потока найти название курса:

```
SELECT
  (SELECT surname FROM students WHERE id = student_id) AS student_name,
  (SELECT name FROM students WHERE id = student_id) AS student_surname,
  (SELECT name FROM courses WHERE id =
    (SELECT course_id FROM streams WHERE id = stream_id)
  ) AS course_name,
  grade
FROM grades;
```

В условиях вложенных запросов мы определили id таблицы вложенного запроса, а student_id и stream_id — таблицы основного запроса grades. Чтобы не путаться, какой таблице принадлежит тот или иной столбец, указываем таблицу явным образом:

```
SELECT
  (SELECT surname FROM students WHERE students.id = grades.student_id)
    AS student_name,
  (SELECT name FROM students WHERE students.id = grades.student_id)
    AS student_surname,
  (SELECT name FROM courses WHERE courses.id =
    (SELECT course_id FROM streams WHERE streams.id = grades.stream_id))
    AS course_name,
  grade
FROM grades;
```

Проверим работу запроса:

```
sqlite> SELECT
   ...>   (SELECT surname FROM students WHERE students.id = grades.student_id)
   ...>     AS student_name,
   ...>   (SELECT name FROM students WHERE students.id = grades.student_id)
   ...>     AS student_surname,
   ...>   (SELECT name FROM courses WHERE courses.id =
   ...>     (SELECT course_id FROM streams WHERE streams.id = grades.stream_id))
   ...>     AS course_name,
   ...>   grade
   ...> FROM grades;
student_name  student_surname  course_name             grade
------------  ---------------  ----------------------  -------------
Иванов        Игорь            Linux. Рабочая станция  4.9
Павлова       Анастасия        Linux. Рабочая станция  5
Иванов        Игорь            Базы данных             5
Иванов        Игорь            Основы Python           4.9
sqlite>
```

Мы получили все требуемые данные, но надо отфильтровать оценки только ученика Игоря Иванова. Сделаем это по его идентификатору:

```
SELECT
  (SELECT surname FROM students WHERE students.id = grades.student_id)
    AS student_name,
  (SELECT name FROM students WHERE students.id = grades.student_id)
    AS student_surname,
  (SELECT name FROM courses WHERE courses.id =
    (SELECT course_id FROM streams WHERE streams.id = grades.stream_id))
    AS course_name,
  grade
FROM grades
```

```
WHERE grades.student_id = 1;
```

Такой вариант запроса будет работать, но, согласно условию, искать пользователя надо по его фамилии и имени, а не по идентификатору. Чтобы это сделать, достаточно заменить значение идентификатора ученика на вложенный запрос к таблице пользователей:

```
SELECT
  (SELECT surname FROM students WHERE students.id = grades.student_id)
    AS student_name,
  (SELECT name FROM students WHERE students.id = grades.student_id)
    AS student_surname,
  (SELECT name FROM courses WHERE courses.id =
    (SELECT course_id FROM streams WHERE streams.id = grades.stream_id))
    AS course_name,
  grade
FROM grades
WHERE grades.student_id =
  (SELECT id FROM students WHERE surname = 'Иванов' AND name = 'Игорь' );
```

Проверим, как работает финальная версия запроса:

```
sqlite> SELECT
   ...>   (SELECT surname FROM students WHERE students.id = grades.student_id)
   ...>     AS student_name,
   ...>   (SELECT name FROM students WHERE students.id = grades.student_id)
   ...>     AS student_surname,
   ...>   (SELECT name FROM courses WHERE courses.id =
   ...>     (SELECT course_id FROM streams WHERE streams.id = grades.stream_id))
   ...>     AS course_name,
   ...>   grade
   ...> FROM grades
   ...> WHERE grades.student_id =
   ...>   (SELECT id FROM students WHERE surname = 'Иванов' AND name = 'Игорь'
);
student_name  student_surname  course_name             grade
------------  ---------------  ----------------------  -------------
Иванов        Игорь            Linux. Рабочая станция  4.9
Иванов        Игорь            Основы Python           4.9
Иванов        Игорь            Базы данных             5
sqlite>
```

Мы получили требуемую информацию одним запросом. Не будем пока затрагивать вопрос оценки эффективности такого подхода, другие способы построения многотабличных запросов рассмотрим далее по курсу. Сейчас важно понять, что, если потребуется, мы включим вложенный запрос в любое место основного запроса, где допускается использование выражения.

Для закрепления материала рассмотрим другие примеры использования вложенного запроса в частях SELECT, FROM и WHERE основного запроса.

Вложенные запросы в части SELECT

Схематически вложенный запрос в части SELECT основного запроса определяется следующим образом:

```
SELECT
  ('Вложенный запрос')
FROM
  'Имя таблицы'
WHERE ['Условие'];
```

Рассмотрим пример задачи, которая решается запросом подобной структуры. Надо вывести номер потока, название курса и дату начала занятий для потоков, начинающихся с ноября 2020 года. Основной запрос построим к таблице потоков, а в часть SELECT поместим вложенный запрос к таблице курсов, чтобы получить название курса по значению course _id.

```
SELECT
  number,
  (SELECT name FROM courses WHERE courses.id = streams.course_id) AS
course_name,
  start_date FROM streams
WHERE start_date >= '2020-11-01';
```

Выполним запрос и проверим результат:

```
sqlite> SELECT
   ...>   number,
   ...>   (SELECT name FROM courses WHERE courses.id = streams.course_id) AS
course_name,
   ...>   start_date FROM streams
   ...> WHERE start_date >= '2020-11-01';
number      course_name  start_date
----------  -----------  ----------
54          Базы данных  2020-11-12
sqlite>
```

Вложенные запросы в части FROM

Структурно запросы такого типа выглядят следующим образом:

```
SELECT
```

```
  'Столбец 1',
  'Столбец 2',
  'Столбец n'
FROM
  ('Вложенный запрос')
WHERE ['Условие'];
```

Для иллюстрации варианта запроса с вложенностью в части FROM используем более сложный пример с агрегированием данных. Пусть надо определить максимальную и минимальную среднюю оценки за все пройденные курсы. Запрос в этом случае выглядит так:

```
SELECT
  MAX(final_grade),
  MIN(final_grade)
FROM
  (SELECT student_id, AVG(grade) AS final_grade
   FROM grades
   GROUP BY student_id);
```

Проверим работу запроса:

```
sqlite> SELECT
   ...>   MAX(final_grade),
   ...>   MIN(final_grade)
   ...> FROM
   ...>   (SELECT student_id, AVG(grade) AS final_grade
   ...>    FROM grades
   ...>    GROUP BY student_id);
MAX(final_grade) MIN(final_grade)
---------------- ----------------
5.0              4.93333333333333
sqlite>
```

Вложенные запросы в части WHERE

Вложенные запросы также применяются в части условия WHERE:

```
SELECT
  'Столбец 1',
  'Столбец 2',
  'Столбец n'
FROM
  'Имя таблицы'
WHERE ('Вложенный запрос');
```

Практический пример такого варианта — поиск всех потоков по названию курса:

```
SELECT
  number,
  start_date
FROM streams
WHERE course_id = (SELECT id FROM courses WHERE name = 'Базы данных');
```

Посмотрим, как работает такой запрос на практике:

```
sqlite> SELECT
   ...>   number,
   ...>   start_date
   ...> FROM streams
   ...> WHERE course_id = (SELECT id FROM courses WHERE name = 'Базы данных');
number      start_date
----------  ----------
54          2020-11-12
sqlite>
```

Объединение данных через UNION

Посредством оператора UNION объединим два или более запроса на выборку данных. Эти запросы могут обращаться как к одной таблице, так и к различным таблицам. Шаблон объединения с использованием UNION выглядит следующим образом:

```
SELECT
  'Столбец 1',
  'Столбец 2'
FROM 'Таблица'
[WHERE 'Условие']

UNION

SELECT
  'Столбец 1',
  'Столбец 2'
FROM 'Таблица'
[WHERE 'Условие'];
```

Рассмотрим применение такого способа объединения. Пусть надо найти идентификатор ученицы Анастасии Павловой, а также идентификаторы всех учеников, у которых оценка за курсы равна пяти.

Сначала составим два запроса для получения требуемой информации отдельно. Найдём идентификатор ученицы **Анастасия Павлова**:

```
SELECT id FROM students WHERE surname = 'Павлова' AND name = 'Анастасия';
```

```
sqlite> SELECT id FROM students WHERE surname = 'Павлова' AND name =
'Анастасия';
id
----------
2
sqlite>
```

Найдём идентификаторы учеников-отличников:

```
SELECT student_id FROM grades WHERE grade = 5;
```

```
sqlite> SELECT student_id FROM grades WHERE grade = 5;
student_id
----------
2
1
sqlite>
```

Посредством оператора UNION объединим два этих запроса:

```
SELECT id FROM students WHERE surname = 'Павлова' AND name = 'Анастасия'
UNION
SELECT student_id FROM grades WHERE grade = 5;
```

В результате получим объединённый результат:

```
sqlite> SELECT id FROM students WHERE surname = 'Павлова' AND name =
'Анастасия'   ...> UNION
   ...> SELECT student_id FROM grades WHERE grade = 5;
id
----------
1
2
sqlite>
```

Заметим, что при использовании UNION мы получаем только уникальные значения из результирующего набора значений каждого запроса. Если надо получить все значения, потребуется воспользоваться вариантом UNION ALL:

```
SELECT id FROM students WHERE surname = 'Павлова' AND name = 'Анастасия'
UNION ALL
SELECT student_id FROM grades WHERE grade = 5;
```

В этом случае мы увидим в результате выборки и все дубликаты:

```
sqlite> SELECT id FROM students WHERE surname = 'Павлова' AND name =
'Анастасия'   ...> UNION ALL
   ...> SELECT student_id FROM grades WHERE grade = 5;
id
----------
2
2
1
sqlite>
```

Для объединения требуется:

- совпадение количества выводимых столбцов каждого запроса;
- одинаковые по типу выводимые данные каждого столбца.

Если эти условия не соблюдаются, выводится ошибка:

```
sqlite> SELECT id, surname FROM students WHERE surname = 'Павлова' AND name =
'Анастасия'
   ...> UNION
   ...> SELECT student_id FROM grades WHERE grade = 5;
Error: SELECTs to the left and right of UNION do not have the same number of
result columns
sqlite>
```

Практическое задание

Работаем с базой данных учителей teachers.db. Для каждого задания создайте запрос, сдать нужно только код запросов в текстовом файле. Для решения воспользуйтесь вложенными запросами и UNION.

1. Найдите потоки, количество учеников в которых больше или равно 40. В отчет выведите номер потока, название курса и количество учеников.

2. Найдите два потока с самыми низкими значениями успеваемости. В отчет выведите номер потока, название курса, фамилию и имя преподавателя (одним столбцом), оценку успеваемости.

3. Найдите среднюю успеваемость всех потоков преподавателя Николая Савельева. В отчёт выведите идентификатор преподавателя и среднюю оценку по потокам.

4. Найдите потоки преподавателя Натальи Петровой, а также потоки, по которым успеваемость ниже 4.8. В отчёт выведите идентификатор потока, фамилию и имя преподавателя. В отчёте должно быть 3 столбца - номер потока, фамилия преподавателя, имя преподавателя.

5. **Дополнительное задание.** Найдите разницу между средней успеваемостью преподавателя с наивысшим соответствующим значением и средней успеваемостью преподавателя с наименьшим значением. Средняя успеваемость считается по всем потокам преподавателя.

Глоссарий

Многотабличный запрос — запрос, в котором происходит обращение к данным нескольких таблиц, например, две таблицы и больше.

Вложенный запрос — запрос, который помещается в некоторую часть другого (основного) запроса. Часто такие запросы называются подзапросами.

Дополнительные материалы

1. Статья «Вложенные SQL-запросы».

2. Статья «Объединение запросов, оператор UNION».

Используемые источники

1. Документация SQLite, Query Language Understood by SQLite.