GeekBrains

Базы данных

Введение в теорию реляционных баз данных

SQLite version 3.31.1

На этом уроке

1. Научимся преобразовывать наборы данных в структуры, которые соответствуют требованиям реляционной модели данных.
2. Определим основные подходы к построению реляционных баз данных.
3. Установим СУБД SQLite в качестве рабочего окружения.

Оглавление

Данные в современном мире

Реляционные базы данных

Сегодня данные — важнейший ресурс во всех областях деятельности человека. Ценность специалистов, имеющих достаточную квалификацию в этой сфере, постоянно растет. Чтобы хранить и использовать данные максимально эффективно, применяются базы данных.

Базу данных рассматриваются как набор данных, которые хранятся в соответствии с некоторой, заранее определенной, структурой. Именно наличие структуры отличает базу данных от произвольного набора значений. Есть различные подходы к определению таких структур, поэтому базы данных бывают различных типов. На нашем курсе мы познакомимся с реляционными базами данных.

Реляционные базы данных строятся на принципах реляционной модели. Термин «реляционная» говорит, что в основе лежит понятие «отношение» (*англ.* relation). Для реляционных баз данных отношением считается таблица. В свою очередь, база данных содержит набор таблиц, связанных в единую структуру.

Чтобы иметь возможность преобразовать любой набор данных к виду, который соответствует требованиям реляционной теории, используется специальный комплекс правил. Эти правила называются нормальными формами, а сам процесс преобразования данных — нормализацией.

Более подробно принципы построения реляционных баз данных и применение нормализации рассмотрим в практической части занятия.

Возможность работы с базами данных обеспечивают системы управления базами данных (СУБД). Важно понимать отличие в использовании терминов «база данных» (БД) и «система управления базами данных» (СУБД). База данных — это структурированный набор данных, а СУБД — приложение, которое обеспечивает работу с базами данных.

Но данные недостаточно только хранить, с ними надо работать — добавлять, изменять, получать требуемую информацию. Для выполнения таких операций используется структурированный язык запросов — SQL (structured query language). SQL — не язык программирования для общего применения, как, например, Java или Python, он предназначен только для работы с реляционными базами данных.

Подходы к хранению и использованию данных

Если попросить несколько неподготовленных людей структурировать один и тот же набор данных, то, скорее всего, мы получим различные варианты.

Распространённый и наглядный вариант — создание структуры в виде таблицы, но таблицы с одними и теми же данными также формируются разными способами.

Начинаем работать с данными

База данных учеников

Рассмотрим задачу по созданию структуры для хранения данных средних оценок учеников за пройденные курсы. Для каждого ученика требуется записать следующую информацию:

1. Имя и фамилия ученика.
2. Курсы, которые ученики проходили в рамках обучения.
3. Количество уроков каждого курса.
4. Оценка, полученная учеником по итогу прохождения курса.

Предварительный вариант структуры

Создадим таблицу, в строках которой мы храним все важные данные. Заметим, что записи об одном и том же ученике иногда повторяются, так как ученик может пройти несколько курсов.

Ученик	Курс	Количество уроков	Оценка
Иванов Игорь	Базы данных, поток 54 от 12.11.2020	12	4.9
Павлова Анастасия	Базы данных, поток 54 от 12.11.2020	12	5
Иванов Игорь	Linux. Рабочая станция, поток 48 от 02.10.2020	8	5
Иванов Игорь	Основы Python, поток 45 от 18.08.2020	8	4.9

Таблица 1. Предварительный вариант базы данных

В целом такой вариант выглядит неплохо. Проанализируем эту структуру на недостатки.

Рассмотрим проблемные места предварительной реализации

Составные значения в ячейках

Первое, на что стоит обратить внимание — значения в столбцах «Ученик» и «Курс» непростые, а состоят из нескольких атрибутов. Недостаток такого подхода в том, что когда записей в нашей базе данных станет много, искать по некоторым значениям будет сложнее, чем по другим.

Например, получится легко отсортировать и найти подходящего ученика по фамилии, нежели по имени. Аналогичная ситуация и с данными в столбце «Курс» — искать по дате начала обучения будет сложнее, чем по названию курса.

Таким образом, часть значений в нашей структуре — составные. Например, в таблице 2 такие значения выделяются красным цветом.

Ученик	**Курс**	**Количество уроков**	**Оценка**
Иванов Игорь	Базы данных, поток 54 от 12.11.2020	12	4.9
Павлова Анастасия	Базы данных, поток 54 от 12.11.2020	12	5
Иванов Игорь	Linux. Рабочая станция, поток 48 от 02.10.2020	8	5
Иванов Игорь	Основы Python, поток 45 от 18.08.2020	8	4.9

Таблица 2. Составные (неатомарные) значения базы данных учеников

Неоднозначные зависимости столбцов таблицы

Заметим, что в отношениях между столбцами таблицы есть некоторые логические несоответствия.

«Курс» и «Оценка» — атрибуты ученика, а «Количество уроков» относится не к «Ученику», а к «Курсу».

Ученик	**Курс**	**Количество уроков**	**Оценка**
Иванов Игорь	Базы данных, поток 54 от 12.11.2020	12	4.9
Павлова Анастасия	Базы данных, поток 54	12	5

	от 12.11.2020		
Иванов Игорь	Linux. Рабочая станция, поток 48 от 02.10.2020	8	5
Иванов Игорь	Основы Python, поток 45 от 18.08.2020	8	4.9

Таблица 3. Атрибуты (зависимости) ученика. Красным цветом выделены данные, которые зависят от курса

Ученик	**Курс**	**Количество уроков**	**Оценка**
Иванов Игорь	Базы данных, поток 54 от 12.11.2020	12	4.9
Павлова Анастасия	Базы данных, поток 54 от 12.11.2020	12	5
Иванов Игорь	Linux. Рабочая станция, поток 48 от 02.10.2020	8	5
Иванов Игорь	Основы Python, поток 45 от 18.08.2020	8	4.9

Таблица 4. Атрибуты (зависимости) курса, красным цветом выделены данные, которые зависят от ученика

Разные зависимости в рамках одной таблицы неизбежно приведут к дублированию (избыточности) данных — для всех записей с одинаковым курсом проставится то же самое значение количества уроков.

Транзитивные зависимости, избыточность данных

Проанализируем содержимое столбца «Курс». Мы видим, что название курса, номер потока и дата начала обучения — атрибуты ученика. Но в то же время название курса и дата начала занятий рассматриваются и как атрибуты потока. Следовательно, мы всегда можем узнать название курса и дату начала занятий, зная только номер потока.

Ученик > (*Название курса, Номер потока, Дата начала обучения, Оценка*)

Название курса и дату начала обучения мы всегда узнаём по номеру потока:

Номер потока > (*Название курса, Дата начала обучения*)

В итоге получим следующую структуру:

Ученик > (**Номер потока** > (*Название курса, Дата начала обучения*), *Оценка*)

Возможность получить значения некоторых атрибутов на основе других называется транзитивной зависимостью. В этом случае мы получаем значения атрибутов «Название курса» и «Дата начала занятий» по значению атрибута «Номер потока», поэтому это транзитивная связь.

На практике наличие транзитивных зависимостей также приводит к дублированию данных. Текущее представление данных в столбце «Курс» наглядно, но избыточно. Так, в таблице достаточно представить номер потока, а для отображения атрибутов потоков потребуется создать другое отношение (таблицу).

Избыточность данных определяется наличием дублированных значений. Такое дублирование приведёт к неэффективному использованию пространства на носителях, а также к усложнению операций с данными — одни и те же данные придётся добавлять или изменять сразу в нескольких местах.

Проблема однотипного ввода одинаковых значений

Во многих случаях одни и те же данные вводятся различными способами. Например, ученика можно ввести как «Игорь Иванов», «Иванов Игорь», «Иванов И.». Все эти записи относятся к одному человеку, но использовать такие данные довольно сложно.

Ученик	Курс	Количество уроков	Оценка
Иванов И.	Базы данных, поток 54 от 12.11.2020	12	4.9
Павлова Анастасия	БД, 12 ноября 2020 г., поток 54	12	5
Игорь Иванов	Linux. Рабочая станция, поток 48 от 02.10.2020	8	5
Иванов Игорь	Основы Python, поток 45 от 18.08.2020	8	4.9

Таблица 5. Пример ввода одинаковых данных, записанных в различном формате

Решаем проблемы методом приведения данных к нормальным формам

Для решения проблем, рассмотренных выше, реляционная теория предлагает метод, который основывается на приведении данных к нормальным формам.

Каждая из нормальных форм — набор требований, которым соответствуют данные. Всего сейчас определено восемь нормальных форм, но, с практической точки зрения, мы рассмотрим только первые три. Логика связей данных в современных приложениях редко требует приведения к нормальным формам выше третьей.

Убираем составные значения, приводим данные к первой нормальной форме

Чтобы решить проблему составных значений, разделим их все на простые (атомарные). Для ученика разделим имя и фамилию на отдельные столбцы, аналогично поступим и с данными о курсе.

Фамилия ученика	Имя ученика	Курс	Номер потока	Дата начала обучения	Количество уроков	Оценка
Иванов	Игорь	Базы данных	54	12.11.2020	12	4.9
Павлова	Анастасия	Базы данных	54	12.11.2020	12	5
Иванов	Игорь	Linux. Рабочая станция	48	02.10.2020	8	5
Иванов	Игорь	Основы Python	45	18.08.2020	8	4.9

Таблица 6. Приведение данных к первой нормальной форме

Полученная структура удовлетворяет требованиям первой нормальной формы:

1. Все значения в таблице должны быть простыми.
2. Не должно быть повторения строк таблицы с одинаковыми данными — идентичными значениями.

Упрощаем зависимости между значениями, приводим данные ко второй нормальной форме

Следующий шаг — упрощение зависимостей в структуре таблицы. В нашем случае мы определили, что атрибуты «Фамилия ученика», «Имя ученика», «Курс», «Номер потока», «Дата начала обучения», «Оценка» относятся к ученику, а атрибут «Количество уроков» — к курсу.

Задача выполняется разбиением исходной таблицы на две новые, в каждой из которых все атрибуты относятся к (зависят от) одной сущности. В результате мы получим таблицу оценок и таблицу курсов.

Фамилия ученика	Имя ученика	Курс	Номер потока	Дата начала обучения	Оценка
Иванов	Игорь	Базы данных	54	12.11.2020	4.9
Павлова	Анастасия	Базы данных	54	12.11.2020	5
Иванов	Игорь	Linux. Рабочая станция	48	02.10.2020	5
Иванов	Игорь	Основы Python	45	18.08.2020	4.9

Таблица 7. Приведение ко второй нормальной форме, таблица оценок

Курс	Количество уроков
Базы данных	12
Linux. Рабочая станция	8
Основы Python	8

Таблица 8. Приведение ко второй нормальной форме, таблица курсов

Полученная структура удовлетворяет требованиям второй нормальной формы:

1. Структура должна находиться в первой нормальной форме.
2. В каждой из таблиц все столбцы атрибутов должны относиться к (зависеть от) одной ключевой сущности.

Решаем проблему транзитивных связей и унифицируем ввод значений, приводим данные к третьей нормальной форме

Для устранения транзитивных зависимостей вынесем атрибуты потоков в отдельную таблицу. А также вынесем все повторяющиеся атрибуты в отдельные таблицы. Затем свяжем эти таблицы по ключевым столбцам.

В результате получим структуру, состоящую из четырёх таблиц: «Курсы», «Потоки», «Ученики» и «Оценки». Обратим внимание, что в таблицах курсов, потоков и учеников добавился служебный ключевой столбец. Он используется, чтобы была возможность ссылаться на соответствующие значения в других таблицах. Ключ строки также рассматривается как некоторый условный номер, или, другими словами, идентификатор записи, уникальный в пределах таблицы. В таблице не указывается несколько записей с одинаковым ключевым значением.

Ключ курса	**Название курса**	**Количество уроков**
1	Базы данных	12
2	Linux. Рабочая станция	8
3	Основы Python	8

Таблица 9. Приведение к третьей нормальной форме, таблица курсов

Ключ потока	**Номер потока**	**Ключ курса**	**Дата начала обучения**
1	45	3	18.08.2020
2	48	2	02.10.2020
3	54	1	12.11.2020

Таблица 10. Приведение к третьей нормальной форме, таблица потоков

Ключ ученика	**Фамилия**	**Имя**
1	Иванов	Игорь
2	Павлова	Анастасия

Таблица 11. Приведение к третьей нормальной форме, таблица учеников

Ключ ученика	**Ключ потока**	**Оценка**
1	1	4.9
2	1	5

1	3	5
1	2	4.9

Таблица 12. Приведение к третьей нормальной форме, таблица оценок

Такая структура соответствует требованиям третьей нормальной формы:

1. Структура таблиц должна соответствовать второй нормальной форме.
2. В таблицах должны отсутствовать транзитивные зависимости.
3. Все повторяющиеся неключевые атрибуты выносятся в отдельные таблицы.

На этом этапе работу по преобразованию данных мы выполнили, а структура соответствует требованиям реляционной модели.

Связи между таблицами

В результате работы по приведению данных к третьей нормальной форме мы получили структуру, состоящую из четырёх таблиц.

Перечислим связи между таблицами:

1. Таблица учеников связана с таблицей оценок по ключу ученика.
2. Таблица курсов связана с таблицей потоков по ключу курса.
3. Таблица потоков связана с таблицей оценок по ключу потока.

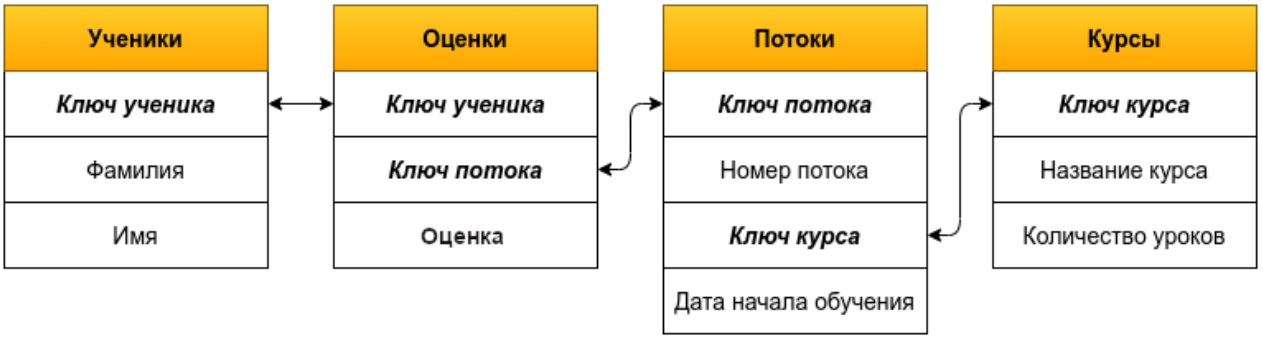

Диаграмма 1. Связи между таблицами

Тип связи между таблицами, когда для ссылок используются значения ключевых столбцов таблиц, называется связью по ключевым столбцам.

Связь по ключевым столбцам обеспечивает гарантию того, что значения могут ссылаться только на те данные, которые действительно есть в других таблицах, то есть обеспечивается ссылочная целостность данных в целом. Например, если у нас в таблице учеников есть сейчас два человека с

ключами (идентификаторами) 1 и 2, то вставляются только эти значения в столбец «Ключ ученика» таблицы оценок. Использование любых других значений приведёт к нарушению ссылочной целостности данных этих таблиц.

Преимущества при использовании правил приведения к нормальным формам

Приведение данных к нормальным формам — не только приложение теоретических правил реляционной теории, оно имеет вполне определённую практическую направленность. Перечислим преимущества, которые мы получаем при использовании нормализованных структур:

1. Исключается избыточность (дублирование) в хранении данных.
2. Обеспечивается согласованность (ссылочная целостность) данных.
3. Применяется единый подход к хранению данных независимо от области применения, что делает более простыми процессы использования, разработки и сопровождения баз данных.

Общий подход к проектированию реляционных баз данных

Пример приведения данных к третьей нормальной форме, рассмотренный выше, даёт представление об основных подходах, используемых для проектирования реляционных баз данных.

1. Сущности реального мира отображаются на таблицы.
2. Атрибуты сущностей реального мира отображаются на столбцы.
3. Данные отдельных экземпляров сущности отображаются на строки таблицы.
4. Между таблицами определяются связи по ключевым столбцам.
5. Каждая таблица должна соответствовать требованиям нормальных форм.

Такой подход универсален и применяется для данных любой предметной области.

Установка рабочего окружения

Для выполнения заданий курса надо установить рабочее окружение, позволяющее работать с реляционными базами данных. Рассмотрим некоторые из доступных вариантов.

Популярные реляционные СУБД

Среди популярных систем управления реляционными базами данных упомянем PostgreSQL, MySQL, Oracle, Microsoft SQL Server. Однако эти системы реализуются по клиент-серверной технологии,

предназначаются для применения, например, используются в высоконагруженных системах и довольно сложны в установке, настройке и эксплуатации.

Есть также встраиваемые СУБД (embedded), они не поддерживают клиент-серверную архитектуру. На сегодняшний день SQLite — самая популярная СУБД такого типа.

СУБД SQLite

SQLite — это компактная кросс-платформенная встраиваемая СУБД, разработка которой ведётся как проект с открытым кодом. По своей структуре эта СУБД — библиотека, которая может быть встроена в приложение.

Такая реализация даёт некоторые значительные преимущества, но в то же время налагает существенные ограничения.

Сильные стороны SQLite

1. База данных сохраняется в одном файле, что предполагает отличную переносимость.
2. Хорошая производительность в системах, где не требуется конкурентная запись данных.
3. Используется также на конечных устройствах, в том числе мобильных и так называемых умных устройствах, в качестве локального хранилища данных.
4. Простая установка и использование, требуется минимум административных операций.
5. Кросс-платформенность: используется во всех распространённых ОС (операционных системах) современного типа.

SQLite хорошо подходит для применения в однопользовательских приложениях, разработке и тестировании.

Слабые стороны SQLite

1. Нет встроенной системы управления пользователями и правами.
2. Ограниченная поддержка стандартов языка.
3. Проблемы с производительностью при конкурентной записи. Нет эффективного механизма, позволяющего одновременно записывать данные по различным операциям.

SQLite не подходит:

- для многопользовательских систем;
- систем, где требуется одновременная запись по нескольким операциям.

Однако эта СУБД прекрасно подойдёт для целей обучения языку запросов SQL, который применяется для работы со всеми реляционными базами данных. Применение SQLite позволит не отвлекаться на

сложные вопросы установки, конфигурирования и использования, характерные для более «тяжёлых» клиент-серверных СУБД.

Установка SQLite

Установка SQLite несложна, ниже приводятся шаги для каждого семейства популярных десктопных ОС.

Установка SQLite на Windows

1. На странице загрузки найдите раздел **Precompiled Binaries for Windows** и скачайте файл sqlite-tools-win32-x86-XXXXXXX.zip.
2. Распакуйте архив.

В новой директории будет находиться исполняемый файл sqlite3 — это программа-клиент SQLite.

Установка SQLite на MacOS

Выберите один из вариантов ниже.

Вариант с использованием MacPorts:

```
sudo port install sqlite3
```

Вариант с использованием Brew:

```
brew install sqlite3
```

Вариант с использованием страницы загрузки:

3. На странице загрузки найдите раздел **Precompiled Binaries for Mac OS X (x86)** и скачайте файл sqlite-tools-osx-x86-3370200.zip.
4. Распакуйте архив.

В новой директории будет находиться исполняемый файл sqlite3 — это программа-клиент SQLite.

Установка SQLite на Linux

Для дистрибутивов, основанных на Debian:

```
sudo apt install sqlite3
```

Для всех дистрибутивов:

1. На странице загрузки найдите раздел **Precompiled Binaries for Linux** и скачайте файл sqlite-tools-linux-x86-XXXXXXX.zip.
2. Распакуйте архив:

```
unzip sqlite-tools-linux-x86-XXXXXXX.zip
```

3. Войдите в новую директорию и выведите содержимое:

```
cd sqlite-tools-linux-x86-XXXXXXX
ls
```

В этой директории будет находиться исполняемый файл sqlite3 — программа-клиент SQLite.

Проверка установки для Windows

Проводником откройте папку, содержащую файл sqlite3.exe и запустите его обычным образом. Игнорируйте предупреждение системы безопасности.

Проверка установки для MacOS и Linux

Выполните в терминале команду:

```
sqlite3
```

Важно! *Если вы устанавливали SQLite на Linux из архива, то при выполнении команды надо указать путь к исполняемому файлу sqlite3, или перед этим перейти в папку, где этот файл находится, и затем запустить sqlite3 следующим способом:*

```
./sqlite3
```

Если всё прошло успешно, вы увидите аналогичное сообщение, а приглашение для ввода команд изменится на sqlite>:

```
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
Connected to a transient in-memory database.
```

```
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

После этого введите команду .exit (точка плюс exit), чтобы выйти из программы.

```
sqlite>.exit
```

Практическое задание

1. Приведите данные по преподавателям к третьей нормальной форме. В качестве отчёта сдайте таблицы с соответствующим содержимым в формате PDF. Рекомендуется выполнить это задание в программе, предназначенной для работы с таблицами — Microsoft Excel, OpenOffice Calc или другой, зависит от вашей рабочей системы. Затем выполните экспорт результата в файл формата PDF. Если такой программы у вас нет, создайте таблицы через сервис Google Docs или в любом текстовом редакторе.

   Дополнительные пояснения к данным:

   - *нумерация потоков сквозная;*
   - *поток определяет прохождение определенного курса некоторой группой учеников;*
   - *следующий курс для группы пойдёт уже другим потоком;*
   - *один ученик может закончить несколько курсов (потоков);*
   - *один преподаватель может вести разные курсы;*
   - *успеваемость учеников — оценка, выставленная преподавателем группе в целом по итогам выполнения проекта, также отражает вовлеченность группы в учебный процесс.*

Преподаватель	Электронная почта	Курс	Номер потока	Дата начала обучения	Количество учеников	Успеваемость учеников
Савельев Николай	saveliev.n@mai.ru	Базы данных	203	12.11.2020	35	4.8
Петрова Наталья	petrova.n@yandex.ru	Основы Python	178	02.10.2020	37	4.9
Малышева Елена	malisheva.e@google.com	Linux. Рабочая станция	165	18.08.2020	34	4.7
Савельев Николай	saveliev.n@mail.ru	Базы данных	210	03.12.2020	41	4.9

Петрова Наталья	petrova.n@yandex.ru	Linux. Рабочая станция	212	14.12.2020	40	4.8

2. Установите программу sqlite3, запустите её, выполните команду .help (точка плюс help). В качестве отчёта сдайте скриншот результата выполнения.
3. Дополнительное задание (выполняется по желанию): приведите данные по использованию социальных сетей к третьей нормальной форме.

Пользователь	Возраст	Instagram	Facebook	TikTok	YouTube	Twitter	ВКонтакте
Михаил Вершинин	23	1	2	3	1	4	1
Екатерина Павлова	16	1	3	1	2	4	4
Василий Иванов	34	2	1	4	3	1	1
Георгий Николаев	25	4	1	1	2	4	3

Пояснение к значениям в таблице:

1 — использует активно;

2 — использует редко;

3 — не зарегистрирован;

4 — зарегистрирован, но не использует.

Глоссарий

База данных — набор данных, хранящихся в соответствии с некоторой предварительно определённой структурой.

Встраиваемые СУБД — системы управления базами данных, которые не имеют реализации в виде отдельного приложения и используются для интеграции в другие системы. Распространённый вариант реализации таких СУБД — библиотека.

Избыточность данных — дублирование данных. Избыточность появляется в случае хранения одних и тех же данных в различных таблицах одной базы данных.

Нормализация — процесс последовательного применения требований нормальных форм к данным.

Нормальные формы — набор правил, выполнение которых позволит привести данные к виду, соответствующему требованиям реляционной модели.

Проект с открытым кодом, открытое программное обеспечение — распространяется на условиях «свободных» лицензий, например, GNU General Public License или BSD License. В большинстве случаев ПО с открытым кодом — свободное и используется любым желающим.

Простое (атомарное) значение — значение, не разделяемое на отдельные части без потери первоначального смыслового контекста.

Реляционная база данных — база данных, соответствующая требованиям реляционной модели данных.

Реляционная модель данных — логическая модель данных, основанная на математическом аппарате теории множеств и логики первого порядка.

Составное значение — значение, разделяемое на отдельные простые значения без потери первоначального смыслового контекста.

СУБД — система управления базами данных. На практике СУБД реализуются в виде приложения, которое разворачивается на соответствующей аппаратной платформе.

Технология клиент-сервер — архитектура, которая предполагает разделение программно-аппаратного комплекса на клиентскую и серверную часть. Клиентская часть предоставляет интерфейс для работы с системой, серверная часть обрабатывает запросы клиентов.

Транзитивная зависимость — определяется наличием атрибутов, значения которых получаются на основе других атрибутов.

Язык запросов SQL — структурированный язык запросов, который применяется для работы с реляционными базами данных.

Дополнительные материалы

1. Статья «Нормализация отношений. Шесть нормальных форм».
2. Статья «Встраиваемая СУБД».
3. SQLite, официальная документация на английском языке: SQLite Home Page.
4. SQLite, документация на русском языке: SQLite, компактная встраиваемая СУБД

Используемые источники

1. SQLite, официальная документация: SQLite Home Page
2. Статья SQLite
3. Статья «SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных»
4. Статья «База данных».
5. Статья «Реляционная база данных».
6. Статья «Реляционная модель данных».
7. Статья «Нормальная форма».