Содержание:

Что такое базы данных

База данных (БД) — это программа, которая позволяет хранить и обрабатывать информацию в структурированном виде.
БД это отдельная независимая программа, которая не входит в состав языка программирования. В базе данных можно сохранять любую информацию, чтобы позже получать к ней доступ.

Базы данных активно используются для динамических сайтов со значительными объемами данных — часто это интернет-магазины, порталы, корпоративные сайты. Такие сайты обычно разработаны с помощью серверного языка программирования (как пример, PHP) или на основе CMS (как пример, WordPress), и не имеют готовых страничек с данными по аналогии с HTML-сайтами. Странички динамических сайтов формируются «на лету» в результате взаимодействия скриптов и баз данных после соответствующего запроса клиента к веб-серверу.

Система управления базами данных

В контексте баз данных стоит рассмотреть понятие СУБД. Система управления базами данных – это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д. В общем СУБД – это система, позволяющая создавать базы данных и манипулировать сведениями из них. А осуществляет этот доступ к данным СУБД посредством специального языка – SQL.

SQL - это язык программирования, используемый в большинстве реляционных баз данных для запроса, обработки и определения данных, а также контроля доступа. SQL был впервые разработан в IBM в 1970-х годах, и Oracle выступил в качестве основного участника, что привело к внедрению стандарта SQL ANSI. SQL дал толчок выпуску многочисленных расширений от таких компаний, как IBM, Oracle и Microsoft. Хотя в настоящее время SQL все еще широко используется, но уже начали

появляться и новые языки программирования.

По характеру использования СУБД делят на однопользовательские (предназначенные для создания и использования БД на персональном компьютере) и многопользовательские (предназначенные для работы с единой БД нескольких компьютеров, объединенных в локальные сети).

Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Это примеры СУБД типа клиент-сервер, именно такие СУБД встречаются чаще всего в контексте понятия хостинга. Их особенности:

- расположение СУБД на сервере с базами данных;
- непосредственный доступ к БД;
- централизованная обработка клиентских запросов на обработку данных;
- высокий уровень надежности, доступности и безопасности;
- повышенная нагрузка на сервер.

В свою очередь, для удобства работы с СУБД используются специальные веб-приложения, которые позволяют посредством графического интерфейса выполнять администрирование сервера баз данных, запускать специальные команды, а также работать с контентом таблиц и баз данных — действия, которые при отсутствии веб-приложения подлежат выполнению средствами консоли.

Эволюция баз данных

Базы данных значительно изменились с момента их появления в начале 1960-х годов. Исходными системами, которые использовались для хранения и обработки данных, были навигационные базы данных – например, иерархические базы данных (которые опирались на древовидную модель и допускали только отношение «один-ко-многим») и базы данных с сетевой структурой (более гибкая модель, допускающая множественные отношения). Несмотря на простоту, эти ранние системы были негибкими. В 1980-х годах стали популярными реляционные базы данных, в 1990-х годах за ними последовали объектно-ориентированные базы данных.

Совсем недавно вследствие роста Интернета и возникновения необходимости более быстрой обработки неструктурированных данных появились базы данных

NoSQL. В настоящее время облачные базы данных и автономные базы данных открывают новые возможности в отношении способов сбора, хранения, использования данных и управления ими.

Примеры использования баз данных

Базы данных нужны для хранения информации. Чтобы получить полное понимание необходимости использования БД в современном веб-программировании, необходимо ответить на три вопроса:

1. Какую информацию и зачем мы храним?
2. В каком виде и как надо хранить эту информацию?
3. Как и каким способом можно получить доступ к этой информации?

Предположим, вы решили сделать сайт, где каждый пользователь может вести личный дневник наблюдения за погодой в своем городе.

Такой сайт должен иметь как минимум одну форму ввода со следующими полями: город, дата, температура, облачность, погодное явление, и так далее.

Каждый день наблюдатель записывает показания погоды в эту форму, чтобы когда-нибудь в будущем вернуться на сайт и посмотреть, какая была погода месяц или даже год назад.

Из этого примера следует, что программист каким-то образом должен сохранять данные из формы для дальнейшего использования.

Кроме обычного просмотра дневника погоды за месяц в виде таблицы, можно сделать и более сложный проект.

Например, чтобы электронный дневник чем-то качественно отличался от своего бумажного аналога, будет неплохо добавить туда возможности для простого анализа: показать какой день был самым холодным в ноябре или какой продолжительности была самая длинная серия пасмурных дней.

Получается, что данные надо не просто как-то хранить, но и иметь возможность их обрабатывать и анализировать. Именно для этих целей и существуют базы данных.

Как хранится информация в базах данных

В основе всей структуры хранения лежат три понятия:

- База данных;
- Таблица;
- Запись.

База данных — это высокоуровневное понятие, которое означает объединение совокупности данных, хранимых для выполнения одной цели.

Если мы делаем современный сайт, то все его данные будут храниться внутри одной базы данных. Для сайта онлайн-дневника наблюдений за погодой тоже понадобится создать отдельную базу данных.

Таблица по отношению к базе данных является вложенным объектом. То есть одна база данных может содержать в себе множество таблиц.

Аналогией из реального мира может быть шкаф (база данных) внутри которого лежит множество коробок (таблиц). Таблицы нужны для хранения данных одного типа, например, списка городов, пользователей сайта, или библиотечного каталога.

Таблицу можно представить как обычный лист в Excel-таблице, то есть совокупность строк и столбцов. Заполняя такую таблицу, пользователь определяет столбцы, у каждого из которых есть заголовок. В строках хранится информация.

В базе данных точно также: создавая новую таблицу, необходимо описать, из каких столбцов она состоит, и дать им имена.

Запись — это строка электронной таблицы. Это неделимая сущность, которая хранится в таблице. Когда мы сохраняем данные веб-формы с сайта, то на самом деле добавляем новую запись в какую-то из таблиц базы данных. Запись состоит из полей (столбцов) и их значений. Но значения не могут быть какими угодно.

Определяя столбец, программист должен указать тип данных, который будет храниться в этом столбце: текстовый, числовой, логический, файловый и т.д. Это нужно для того, чтобы в будущем в базу не были записаны данные неверного типа.

Соберем всё вместе, чтобы понять, как будет выглядеть ведение дневника погоды при участии базы данных.

1. Создадим для сайта новую базу данных и дадим ей название «weather_diary».
2. Создадим в базе данных новую таблицу с именем «weather_log» и определим там следующие столбцы:
    - Город (тип: текст);
    - День (тип: дата);
    - Температура (тип: число);
    - Облачность (тип: число; от 0 (нет облачности) до 4 (полная облачность));
    - Были ли осадки (тип: истина или ложь);
    - Комментарий (тип: текст).
3. При сохранении формы будем добавлять в таблицу weather_log новую запись, и заполнять в ней все поля информацией из полей формы.

Теперь можно быть уверенными, что наблюдения наших пользователей не пропадут, и к ним всегда можно будет получить доступ.

Как правильно спроектировать базу данных

Перед созданием базы данных надо знать, что:

- атомарность предполагает, что значение нельзя разделить на несколько атрибутов;
- под кортежем понимается запись (строка) в таблице базы данных;
- атрибут — это колонка таблицы;
- неключевой атрибут — это атрибут, не входящий в состав никакого потенциального ключа.

Есть минимум два требования, которые должны быть соблюдены при проектировании структуры БД:

1. Сохранить всю информацию после разделения её на таблицы.
2. Минимизировать избыточность того, как эта информация хранится.

Также есть рекомендации, которые помогут добиться эффективной структуры:

- используйте хотя бы третью нормальную форму;
- создавайте ограничения для входных данных;
- не храните ФИО в одном поле, также как и полный адрес;
- установите для себя правила именования таблиц и полей.

Нормальные формы — это требования, которые должны соблюдаться при правильной проектировке базы данных. Нормальных форм существует целых 6 штук, однако обычно соблюдают всего лишь 3 и для начала этого более чем достаточно.

Первая нормальная форма:

Для примера будем использовать отношение сотрудники_отделы_проекты. В нём есть информация о номере сотрудника, его фамилии, номере отдела, в котором он работает, номере телефона отдела и так далее.

Н_СОТР	ФАМ	Н_ОТД	ТЕЛ	***Н_ПРО***	ПРОЕКТ	Н_ЗАДАН
1	Иванов	1	11-22-33	*1*	Космос	1
1	Иванов	1	11-22-33	*2*	Климат	1
2	Петров	1	11-22-33	*1*	Космос	2
3	Сидоров	2	33-22-11	*1*	Космос	3
3	Сидоров	2	33-22-11	*2*	Климат	2

Это отношение, как и любое другое, автоматически находится в первой нормальной форме:

- в отношении нет одинаковых кортежей;
- кортежи не упорядочены;
- атрибуты не упорядочены и различаются по наименованию;
- все значения атрибутов атомарны.

Вторая нормальная форма:

В нашем случае у таблицы выше имеется сложный (составной) ключ {Н_СОТР, Н_ПРО}. От части ключа Н_СОТР зависят неключевые атрибуты ФАМ, Н_ОТД, ТЕЛ. От части ключа Н_ПРО зависит неключевой атрибут ПРОЕКТ. А вот атрибут Н_ЗАДАН зависит от всего составного ключа, так как сотрудник может выполнять одно задание в одном проекте.

Поэтому для приведения отношения ко второй нормальной форме из отношения сотрудники_отделы_проекты нужно выделить два отношения сотрудники_отделы и проекты, а исходное отношение оставим отношением задания.

Н_СОТР	ФАМ	Н_ОТД	ТЕЛ
1	Иванов	1	11-22-33
2	Петров	1	11-22-33
3	Сидоров	2	33-22-11

Н_СОТР	Н_ПРО	Н_ЗАДАН
1	1	1
1	2	1
2	1	2
3	1	3
3	2	2

Н_ПРО	ПРОЕКТ
1	Космос
2	Климат

Третья нормальная форма:

Отношение находится в третьей нормальной форме, когда отношение находится во второй нормальной форме и все неключевые атрибуты взаимно независимы.

Для того, чтобы устранить зависимость неключевых атрибутов, нужно произвести декомпозицию отношения ещё на несколько отношений. При этом те неключевые атрибуты, которые являются зависимыми, выносятся в отдельное отношение.

Отношение сотрудники_отделы не находится в третьей нормальной форме, так как имеется зависимость неключевых атрибутов, таких как зависимость номера телефона от номера отдела. Поэтому декомпозируем отношение

сотрудники_отделы на два отношения — сотрудники и отделы:

Н_ОТД	ТЕЛ
1	11-22-33
2	33-22-11

Н_СОТР	ФАМ	Н_ОТД
1	Иванов	1
2	Петров	1
3	Сидоров	2

База данных — это не просто набор таблиц. В неё встроено много инструментов, которые помогут с сохранностью и качеством данных.

В первую очередь БД поможет с ограничением значений, которые принимают поля.

Внешние ключи регламентируют отношения между таблицами. Благодаря им сильно упрощается контроль за структурой базы, уменьшается и упрощается код приложения. Правильно настроенные внешние ключи — это гарант того, что увеличится целостность данных за счёт уменьшения избыточности. Поэтому обязательно применяйте ограничение внешнего ключа при определении связей между таблицами.

Выражения ON DELETE и ON UPDATE внешних ключей используются для указания действий, которые будут выполняться при удалении строк родительской таблицы (ON DELETE) или изменении родительского ключа (ON UPDATE). Не пренебрегайте ими.

Стоит убедиться, что обязательность заполнения (NOT NULL) проверяется для полей, которые строго не должны оставаться пустыми.

Используйте CHECK, чтобы убедиться, что значения входят в диапазон (например, чтобы цена не была отрицательной).

Также не стоит хранить ФИО в одном поле, также как и полный адрес. Представим ситуацию, когда вам понадобится узнать, в каком городе продукт более популярен.

В таком случае, если полный адрес хранится в виде цельной строки, сделать это будет очень тяжело, ведь вам нужно будет каким-то образом выделить из этой строки город. Учитывая все возможные форматы и варианты адресов, эта задача становится практически невыполнимой. Похожая ситуация и с ФИО. Даже если кажется, что это ни к чему, храните эти данные в разных полях, и в будущем вы поблагодарите себя.

Установите для себя правила именования таблиц и полей. Сложно работать с данными, которые выглядят как-то
так: user.firstName, user.last_name, user.birthDate. Конечно, каждый программист в праве сам выбирать для себя стиль наименования, но для SQL рекомендуется выбрать наименование с подчёркиванием. Потому что не все SQL-движки одинаково работают с заглавными буквами, а помещать всё в кавычки бывает утомительно.

Ещё нужно определиться как будут называться таблицы — во множественном числе (users) или в единственном (user). Каждая базовая структура в БД обычно настроена на множественное число, поэтому и именовать таблицы стоит соответственно.

Не упускайте возможность сложить побольше обязанностей на базу данных, чтобы облегчить себе работу над приложением и думать о его структуре, а не о контроле табличных связей.

Всё приходит с опытом. Спроектируйте две-три схемы, и картинка сама сложится у вас в голове. Отталкивайтесь от задачи —некоторыми рекомендациями иногда можно пренебречь.

Различие между базой данных и электронной таблицей

Базы данных и электронные таблицы (в частности, Microsoft Excel) предоставляют удобные способы хранения информации. Основные различия между ними заключаются в следующем.

- Способ хранения и обработки данных
- Полномочия доступа к данным
- Объем хранения данных

Заключение

Электронные таблицы изначально разрабатывались для одного пользователя, и их свойства отражают это. Они отлично подходят для одного пользователя или небольшого числа пользователей, которым не нужно производить чрезвычайно сложные операции с данными. С другой стороны, базы данных предназначены для хранения гораздо больших наборов упорядоченной информации—иногда огромных объемов. Базы данных позволяют множеству пользователей в одно и то же время быстро и безопасно получать доступ к данным и запрашивать их, используя весьма сложную логику и язык.