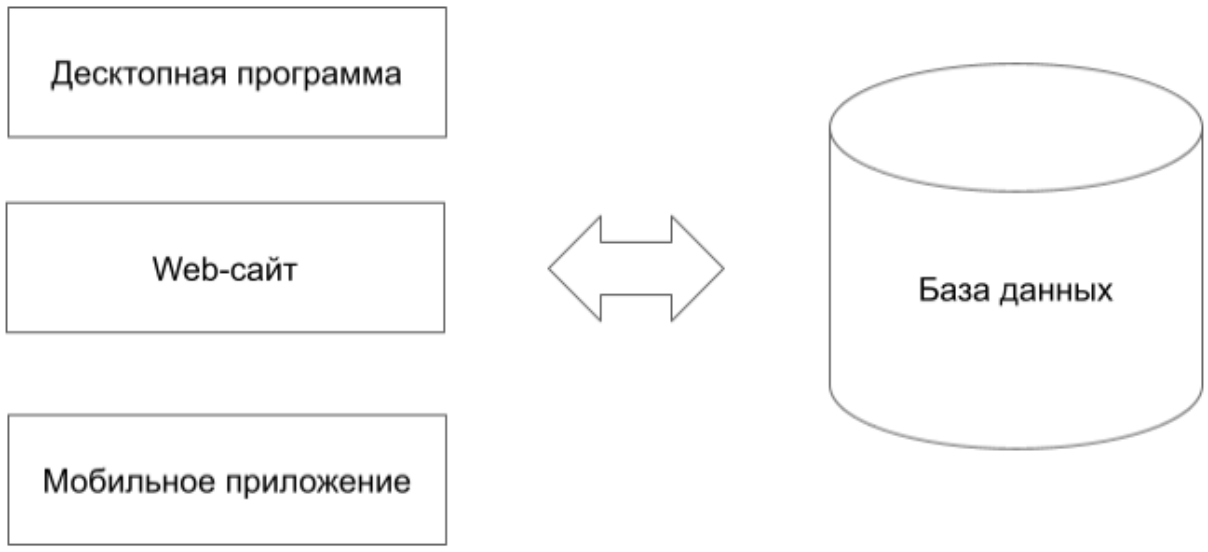
Основы реляционных баз данных (вступительная лекция)

Доброго времени суток, уважаемые студенты! В этом небольшом конспекте мы поговорим о истории развития СУБД, о реляционных и не реляционных БД

Данные и программы

Данные живут дольше, чем программы. Наши программы пока слишком недолговечны и часто меняются. Во время жизненного цикла данных в разное время их может обслуживать несколько программ. Иногда одни и те же данные обслуживает несколько программ одновременно. Поэтому в программировании принято отделять данные от кода и держать их в специализированном хранилище — **базе данных.**



База данных — это совокупность информационных материалов, организованных таким образом, чтобы их можно было найти и обработать при помощи компьютера.
Обычный текстовый файл тоже база данных, пусть и очень примитивная.

Почему недостаточно обычных файлов?

Дело в том, что файлы довольно ограничены по возможностям. При работе с большим объемом данных необходимо обеспечить их компактность. Их лучше записывать в бинарном, а не текстовом формате, возможно, применяя механизмы сжатия. Это не очень наглядно, и для восприятия в любом случае потребуется конвертация данных в формат, доступный человеку. Когда множество параллельных процессов или клиентов обращаются к файлу с целью записать или извлечь информацию из него, очень трудно обеспечить конкурентный доступ. Необходимо либо прибегать к блокировкам файла, либо создавать очередь для запросов. В файлы очень легко записывать информацию, если мы вносим ее в самый конец, однако очень не просто отредактировать запись в середине. Кроме того, чтобы что-то найти в файле, приходится его сканировать от начала до конца. Чтобы кешировать часто используемые данные в оперативной памяти, вам придется писать собственную программу. Ситуация еще больше усложняется, если наш файл гигантского объема и просто не помещается на одном компьютере.

И вот у вас уже несколько файлов, которые хранятся на нескольких компьютерах. Как искать среди них информацию? Придется писать дополнительное программное обеспечение. А что будем делать, если одновременно два клиента захотят исправить один и тот же документ, внося в него совершенно разную информацию?
Поэтому практически сразу после появления операционных систем и файлов над базами данных стали появляться программные надстройки. Они позволяли управлять, искать, пополнять и редактировать данные внутри базы данных. Решать те все проблемы, которые мы с вами обозначили. Такая надстройка стала называться системой управления базами данных или сокращенно СУБД. В нашем курсе для краткости мы будем называть базами данных совокупность как самого хранилища, так и этой программной надстройки.

История развития СУБД

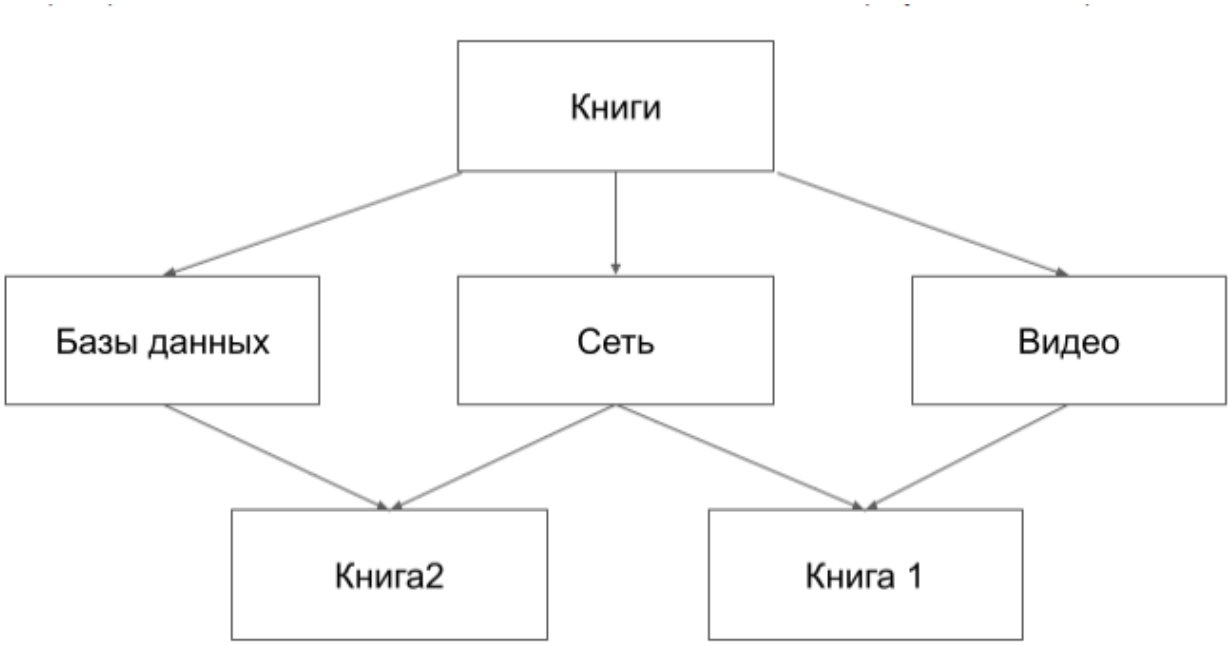
Современные базы данных за последние 60 лет прошли длительный путь развития. Давайте кратко пробежимся по истории СУБД, от первых иерархических баз данных до современных
NoSQL-решений.

Иерархические базы данных

Первые базы данных были иерархическими. Это, наверное, вообще первое, что приходит в голову программистам.
Иерархия — это дерево, состоящее из узлов, у которых может быть несколько потомков. При помощи такой структуры хорошо описываются иерархические структуры организаций и производств. Примеров иерархий очень много и они постоянно находятся у нас перед глазами. На экране вы можете видеть иерархию транспортной системы. Есть вершина — транспорт, от которого расходятся узлы с видами транспорта и последующей детализацией специализации ТС.
Иерархии очень наглядны и хорошо описываются деревьями, у которых прекрасно изученный мат. аппарат.

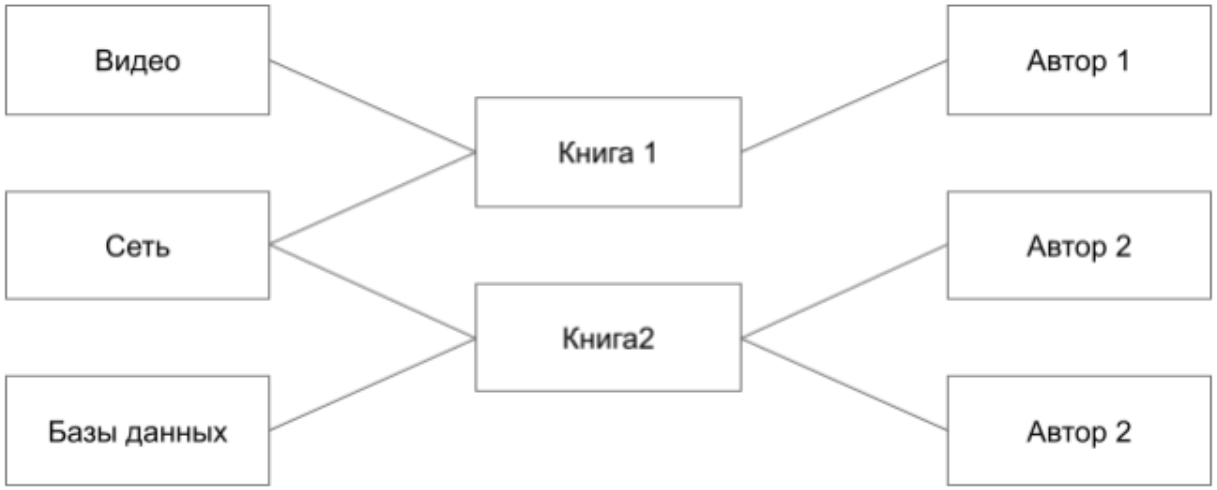
Главное их достоинство — высокая скорость обработки операций. Первые компьютеры не отличались высокой производительностью: чем проще организована база данных, тем быстрее она работает.

Основной недостаток иерархической структуры базы данных — невозможность реализовать
отношения «многие ко многим». Например, если мы создаём каталог книг, одна книга может относиться сразу к нескольким разделам.



Сетевые базы данных

Была разработана новая модель данных — сетевая. Она расширила иерархическую модель, позволяя одной записи участвовать в нескольких отношениях «предок-потомок».

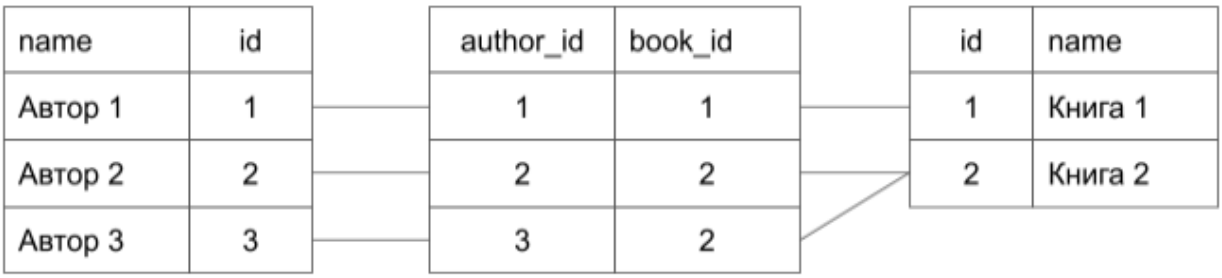


В связи с развитием социальных сетей эти базы данных получили второе дыхание в виде графовых СУБД, которые относятся к современному NoSQL-течению. Только в современной интерпретации ценность приобретают не сами данные, а связи между узлами. Тем не менее, что примечательно, многие идеи, которые появляются в новых популярных NoSQL-базах, были придуманы или опробованы в прошлом.

Просто на тот момент это было либо экономически нецелесообразно, либо мода и влияние больших компаний толкало рынок в сторону других моделей. Конечно, у сетевых баз данных имелись недостатки: подобно своим иерархическим предкам, сетевые базы данных были очень жесткими. Наборы отношений и структура записей должны были быть заданы наперед. Изменение структуры базы данных обычно означало ее полную перестройку.

Реляционные базы данных

Следующим шагом стало развитие реляционных баз данных. Реляционная модель данных была попыткой упростить структуру базы данных. В ней отсутствовала явная структура «предок-потомок», а все данные были представлены в виде простых таблиц, разбитых на строки и столбцы.



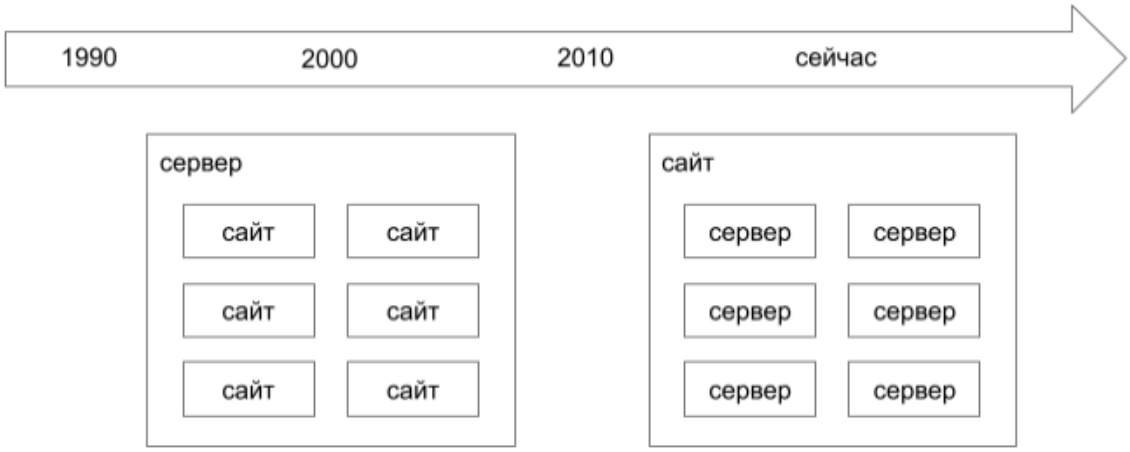
Теоретические основы новой реляционной модели данных впервые были описанны доктором Коддом в 1970 году. Поначалу его работа предоставляла лишь академический интерес. Однако, спустя 10 лет, основываясь на его работе, реляционные базы данных создали сначала Oracle, затем IBM, а потом и множество других компаний.
Реляционные СУБД прочно вошли в компьютерный мир и актуальны до сих пор. Это самый
распространенный вид баз данных. Львиная доля курса будет посвящена именно ему.
За 40 лет было множество попыток заменить реляционные баз данных чем-то более новым и
прогрессивным. Долгое время на смену СУБД пророчили приход XML и объектно-ориентированных баз данных. Однако эти технологии так и не стали массовыми, хотя продукты существуют и по сей день. XML вышел из моды из-за своей избыточности. ООП-базы данных, которые решают проблему несовместимости реляционных баз данных, основанных на множествах, и ООП-программ, основанных на деревьях, тоже не стали слишком популярны. Наиболее популярные СУБД на сегодняшний день — Oracle, MS SQL и DB2 среди коммерческих, MySQL, PostgreSQL и Firebird среди свободных.

Индекс популярности баз данных

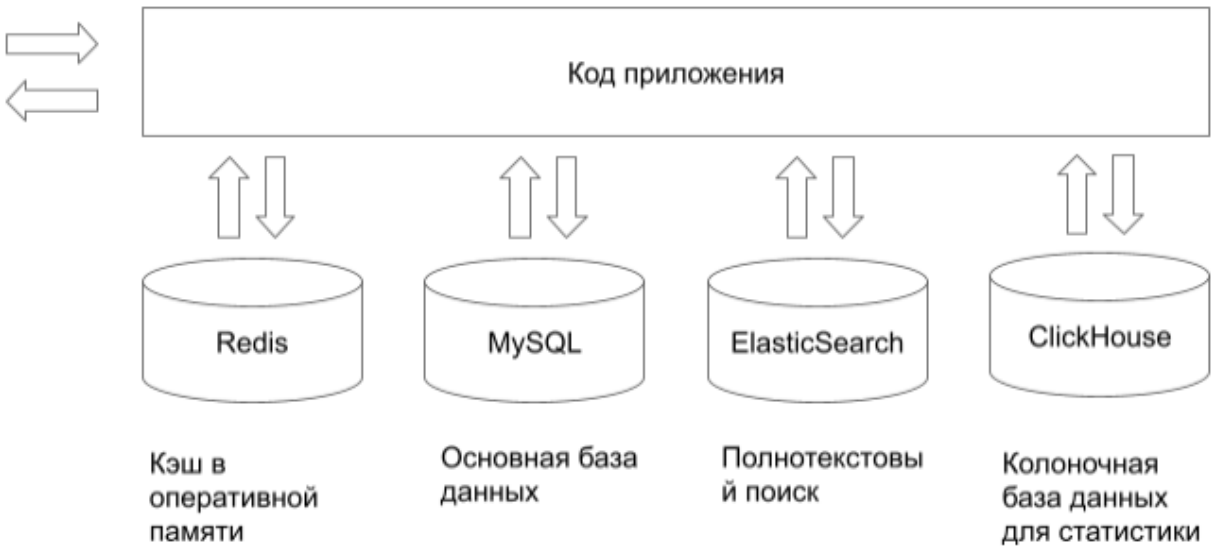
Следить за индексом популярности баз данных можно по рейтингу на сайте db-engines.com. Здесь представлены не только реляционные базы данных, но и NoSQL-решения. Тем не менее, по этому ресурсу вы можете отслеживать динамику интереса к базам данных на протяжении длительного времени.

NoSQL базы данных

После стремительного развития интернета в принципах построения баз данных произошел
переломный момент. Первые веб-сайты и сервисы были очень невелики: зачастую на одном сервере убирались сотни сайтов. Однако по мере вовлечения все новых и новых пользователей проекты начали укрупняться и очень скоро им стало не хватать не то что одного, а десятков, сотен, а затем и тысяч серверов.



Не стали помещаться на одном сервере и базы данных. Поэтому очень скоро сначала реляционные СУБД, а потом и новые игроки стали отказываться от традиционного подхода хранения данных. Стали строиться распределенные хранилища и интенсивно использоваться гораздо большие объемы оперативной памяти. Новые подходы и распределенная структура баз данных привели к ситуации, когда реализовать стандартный SQL-язык стало либо очень сложно, либо почти невозможно. В результате на рынке стали появляться специализированные СУБД, ориентированные под решение тех или иных задач, зачастую полностью расположенные в оперативной памяти, предоставляющие свой язык запросов, иногда вообще не следующий многолетней традиции SQL.



На экране представлены типичные представители NoSQL-базы данных. На самом деле их гораздо больше.
Redis — это очень быстрое хранилище построенное по принципу «ключ-значение». Оно полностью расположено в оперативной памяти, сервер реализован в виде однопоточного EventLoop-цикла, когда один поток опрашивает по кругу соединения в неблокирующем режиме. За счет того, что не происходит переключение процессора на другие процессы, достигается гигантская производительность порядка 100 000 RPS (это зачастую в сотни раз выше, чем в лучших реляционных базах данных).

Если на сервере не хватает оперативной памяти, чтобы разместить индекс, можно разбить данные на части — шарды и хранить несколько копий такого шарда на разных компьютерах. В результате образуется кластер, который ведет себя как единый компьютер с огромным количеством оперативной памяти. Так действуют grid-решения в **Oracle**, **MySQL**, так могут поступать и **NoSQL**-базы данных вроде ElasticSearch и **MongoDB**.

Собирать JSON-документ из нескольких таблиц может быть долго и накладно. В этом случае можно хранить не отдельные значения документа, а готовый, собранный заранее, документ. Для этого используются документо-ориентированные СУБД, примером может служить та же **MongoDB**. Часто в реляционных базах данных штатный механизм полнотекстового поиска не предусмотрен или реализован неэффективно. Поэтому можно прибегать к базам данных, специально предназначенных для полнотекстового поиска, позволяющих

регулировать любые параметры поискового механизма. Яркий представитель таких баз данных — **ElasticSearch**.

Традиционные СУБД плохо предназначены для операций в реальном времени, например для обсчета статистики. Гораздо лучше для этих целей подходит колоночная база данных. В ней запрещены операции редактирования и удаления, данные сжаты, что позволяет обеспечивать исключительно быстрый механизм агрегации. Один из представителей таких баз данных — **ClickHouse**.

Большой проблемой для баз данных, разработанных в прошлом, стала распределенная природа современных приложений. Они не только работают на нескольких серверах, но и зачастую разбросаны по нескольким дата-центрам в разных точках мира. Это грозит тем, что разные части распределенной базы данных могут терять связность и необходимы меры по поддержанию работоспособности и доступности в таких условиях. Как раз для этого предназначена база данных **Cassandra**.

Это не значит, что у новых NoSQL-баз данных вообще нет недостатков, их много и они настолько существенны, что не позволят вам отказаться от традиционных реляционных баз данных. Просто эксплуатируя NoSQL-базу данных для решения задач, под которые она заточена, можно добиться удивительных успехов. При этом важно не использовать такое решение там, где проявляются слабые стороны того или иного хранилища.

Поэтому современный сайт или приложение может использовать совокупность нескольких хранилищ. Например, мы можем запоминать результат ресурсоемкой операции для ускорения чтения, т. е., использовать кеш. Можем использовать основную базу данных для долговременного хранения. Можем предоставлять пользователям возможность искать данные по ключевому слову или фильтровать их различными способами. Чтобы время от времени перемалывать большие объемы накопленных данных, идеально подходят колоночные базы данных. На протяжении всего курса мы будем рассматривать реляционные базы данных на примере **MySQL**.

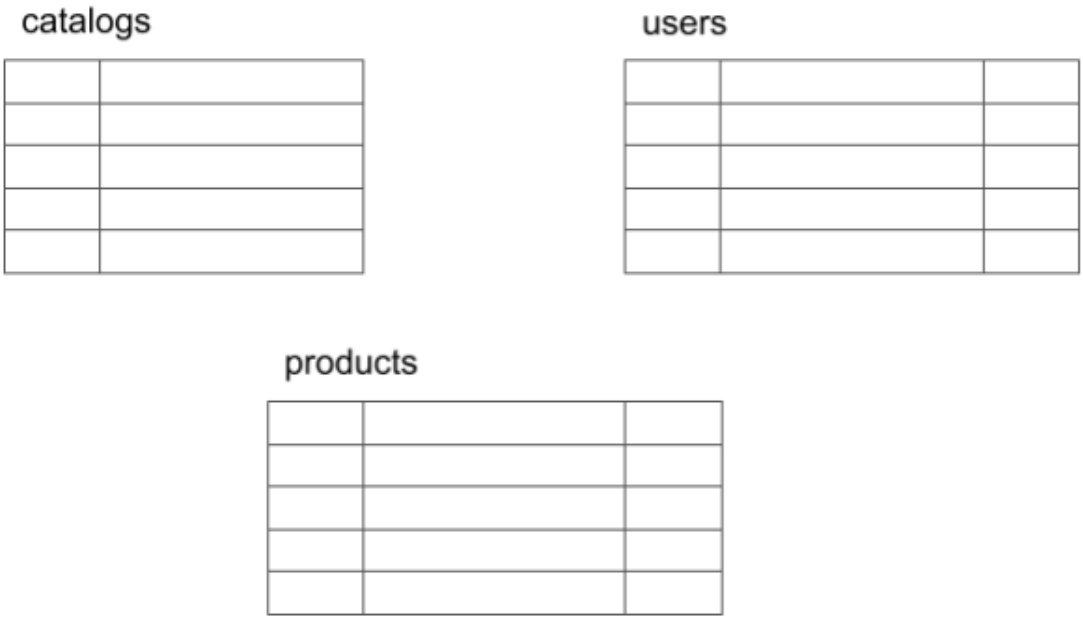
Основы реляционных баз данных

Большая часть курса будет посвящена реляционным базам данных. В них информация организована в виде прямоугольных таблиц, разделенных на строки

и столбцы, на пересечении которых содержатся значения.

Таблицы

База данных состоит из нескольких таблиц. Каждая таблица имеет уникальное имя, описывающее ее содержимое.



Начнем формировать базу данных, с которой мы будем работать в течение курса. Пусть это будет база данных интернет-магазина компьютерных комплектующих. Ниже представлена таблица **catalogs**.

id	name	total
1	Процессоры	15
2	Видеокарты	10
3	Материнские платы	24
4	Оперативная память	12

Строки

Каждая горизонтальная строка этой таблицы представляет отдельную физическую сущность — один каталог. Четыре строки таблицы вместе представляют все четыре каталога интернет-магазина. Все данные, содержащиеся в конкретной строке таблицы, относятся к каталогу, который описывается этой строкой.

Столбцы

Каждый вертикальный столбец таблицы catalogs представляет один элемент данных для каждого из каталогов. На пересечении строки и столбца таблицы содержится только одно значение. Например, в строке, представляющей видеокарты, в столбце name содержится название раздела. В столбце **total** этой же строки находится значение 10, сообщающее количество доступных для покупки товаров. Все значения, содержащиеся в одном и том же столбце — данные одного типа. Например, в столбце **name** содержатся только строки, в столбце **total** — только числовые значения. У каждого столбца в таблице есть свое имя, которое обычно служит его заголовком. Все столбцы в одной таблице должны иметь уникальные имена, однако разрешается присваивать одинаковые имена столбцам,
расположенным в различных таблицах. На практике такие имена столбцов, как name (имя), id (идентификатор), description (описание) и тому подобные, часто встречаются в различных таблицах одной базы данных. Столбцы таблицы упорядочены слева направо, и их порядок определяется при создании таблицы. В любой таблице всегда есть как минимум один столбец.
В отличие от столбцов, строки таблицы не имеют определенного порядка. Это значит, что если последовательно выполнить два одинаковых запроса для отображения содержимого таблицы, нет гарантии, что оба раза строки будут перечислены в одном и том же порядке. Конечно, можно попросить SQL-запрос отсортировать строки перед выводом, однако порядок сортировки не имеет ничего общего с фактическим расположением строк в таблице.

Пустая таблица

В таблице может содержаться любое количество строк. В том числе и ноль строк, в этом случае таблица называется пустой. Пустая таблица сохраняет структуру,

определенную ее столбцами, просто в ней не содержатся данные.

Первичный ключ

Поскольку строки в реляционной таблице не упорядочены, нельзя выбрать строку по ее номеру в таблице. В таблице нет первой, последней или тринадцатой строки.

В правильно построенной реляционной базе данных в каждой таблице есть столбец (или комбинация столбцов), для которого значения во всех строках различны. Этот столбец (или столбцы) называется первичным ключом (primary key) таблицы.

Первичный ключ у каждой строки уникальный. В таблице с первичным ключом нет двух совершенно одинаковых строк. Таблица, в которой все строки отличаются друг от друга, в математических терминах называется отношением (relation). Именно этому термину реляционные базы данных и обязаны своим названием, поскольку в их основе лежат отношения, т. е., таблицы с отличающимися друг от друга строками.

id	name	total
1	Процессоры	15
2	Видеокарты	10
3	Материнские платы	24
4	Оперативная память	12

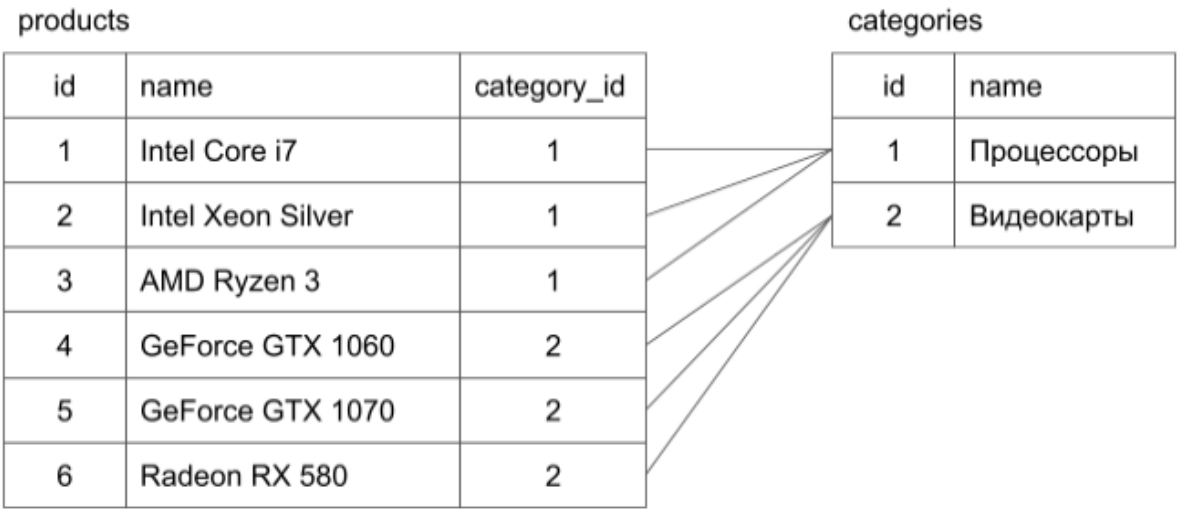
Связи между таблицами

В иерархических базах данных довольно легко выстраивать отношения «предок-потомок». В

реляционной базе данных происходит отказ от явных связей, однако, отношение «предок-потомок» между категориями и товарными позициями не утеряно.

Оно реализовано в виде одинаковых значений, хранящихся в двух таблицах, а не

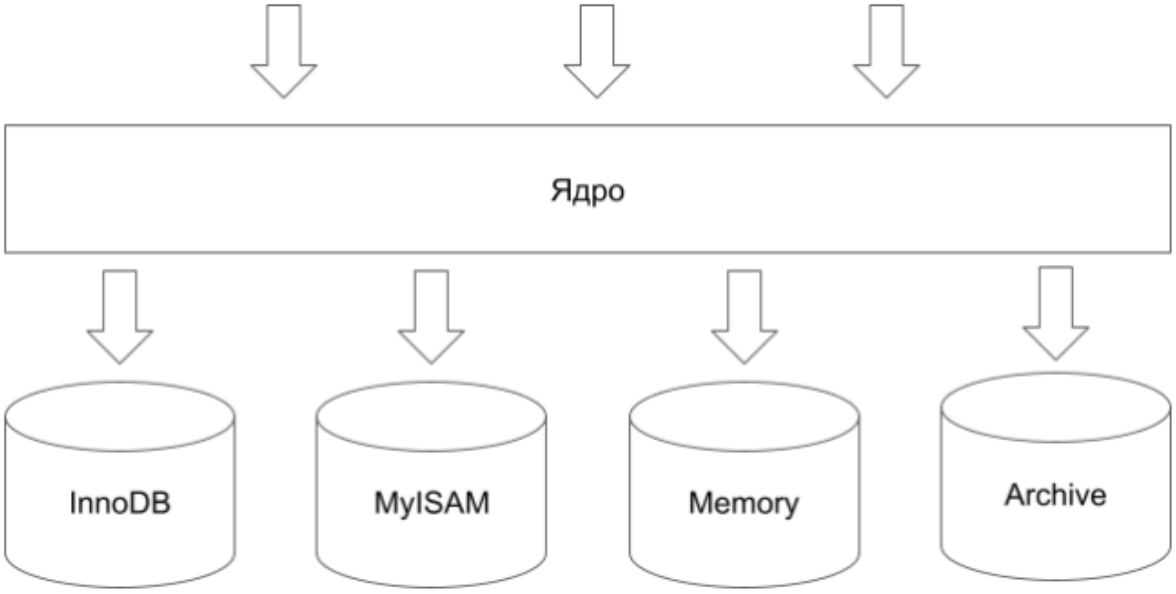
в виде явного указателя. Таким способом реализуются все отношения, существующие между таблицами реляционной базы данных.
Столбец одной таблицы, значения в котором совпадают со значениями столбца, являющегося
первичным ключом другой таблицы, называется внешним ключом (foreign key). Например, здесь на экране внешним ключом выступает столбец category_id. Одним из главных преимуществ реляционных баз данных — возможность извлекать связанные между собой данные, используя эти отношения.



Для нас наличие первичного ключа сейчас является чем-то само собой разумеющимся, хотя первые реляционные СУБД его просто не поддерживали.

Архитектура MySQL

MySQL на сегодняшний день самая популярная база данных с открытым кодом. По популярности ее превосходит только коммерческая **СУБД Oracle**. **MySQL** используется во множестве проектов, наверное, самый известный из них — электронная энциклопедия **Wikipedia**. Существует множество форков этой базы данных: **Percona**, **MariaDB**. Мы будем знакомиться с оригинальной версией **MySQL**.



Архитектуру **MySQL** можно условно разбить на две части: это ядро, сама база данных и движки, которые реализуют тот или иной механизм баз данных. На экране представлено несколько движков, которые могут использоваться **MySQL**. По умолчанию используется **InnoDB**. Более подробно каждый из движков будет рассмотрен на следующих уроках.

Такая архитектура позволяет разрабатывать базу данных усилиями нескольких команд. Одна команда может сосредоточиться на ядре, другая — на каком-либо отдельном движке. Например, движок **InnoDB** долго разрабатывался отдельной компанией.

MySQL построена по клиент-серверной технологии: и клиент, и сервер являются программами, которые могут быть расположены на разных компьютерах или на том же самом. Сервер хранит и обслуживает базы данных, клиенты шлют ему запросы на языке **SQL** и получают в ответ результирующие таблицы с данными.

Информационная схема

Операторы SHOW и DESCRIBE являются нестандартными, другие базы данных, отличные от MySQL, их могут не предоставлять. Более того, возможности этих операторов довольно ограничены. Каждая СУБД имеет системную базу данных, в которой хранятся учетные записи, таблицы привилегий и другая информация, необходимая для управления СУБД. В MySQL такая системная база данных носит

название mysql. Структура системной базы данных для разных СУБД отличается: для того, чтобы унифицировать процесс обращения к системной базе данных, вводится специальный набор представлений, оформленных в виде базы данных INFORMATION_SCHEMA, которая доступна каждому клиенту MySQL.
База данных INFORMATION_SCHEMA является виртуальной и располагается в оперативной памяти — для неё нет физического соответствия на жёстком диске, как для других баз данных. Это означает, что невозможно выбрать базу данных INFORMATION_SCHEMA при помощи оператора USE, как и выполнить по отношению к таблицам этой базы данных запросы с участием операторов INSERT, UPDATE и DELETE.

Допускается использование только оператора SELECT:

```
SELECT * FROM INFORMATION_SCHEMA.SCHEMATA
```

Операторы SHOW и DESCRIBE короче, но не закреплены в стандарте SQL, они работают только в MySQL. Информационную схему обязаны реализовывать все реляционные СУБД, поддерживающие язык запросов SQL. С таблицами информационной схемы можно работать как с таблицами любой базы данных. Например, чтобы извлечь список таблиц базы данных shop, можно воспользоваться следующим запросом:

```
SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'shop';
```

Дополнительные материалы

Базам данных вообще и MySQL в частности посвящено большое количество литературы разной степени сложности. Помимо книг, посвященных СУБД MySQL, следует обращать внимание на книги, которые специализируются на языке запросов SQL. Это стандартный специализированный язык для общения с реляционными базами данных. Больше половины курса будет посвящена именно ему, почти все его элементы одинаковы для всех баз данных. Существуют исключения, вроде команд SHOW и DESCRIBE, которые могут быть реализованы в SQL-диалекте одной базы данных и отсутствовать в другой.

1. Шварц Б., Зайцев П., Ткаченко В., Заводны Дж., Ленц А., Бэллинг Д. MySQL. Оптимизация
производительности, 2-е издание. — Пер. с англ. — СПб.: Символ-Плюс, 2010. — 832 с.
2. Чарльз Белл, Мэтс Киндал и Ларс Талманн. Обеспечение высокой доступности систем на
основе MySQL / Пер. с англ. — М. : Издательство "Русская редакция"; СПб. : БХВ-Петербург,
3. — 624 с.
4. Чаллавала Ш., Лакхатария Дж., Мехта Ч., Патель К. MySQL 8 для больших данных. — М.: ДМК
Пресс, 2018. — 226с.
5. Поль Дюбуа. MySQL. — Пер. с англ. — М.: ООО "И.Д. Вильямс", 2007. — 1168 с.
6. Поль Дюбуа. MySQL. Сборник рецептов. — Пер. с англ. — М.: Символ-Плюс, 2004. — 1056 с.
7. Кузнецов М.В., Симдянов И.В. MySQL 5. — СПб.: БХВ-Петербург, 2006. — 1024с.
8. Дейт К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский
дом "Вильямс", 2005. — 1328 с.
9. Кляйн К., Кляйн Д., Хант Б. SQL. Справочник, 3-е издание. — Пер. с англ. — СПб:
Символ-Плюс, 2010. — 656с.
10. Линн Бейли. Head First. Изучаем SQL. — СПб.: Питер, 2012. — 592 с.
11. Грофф, Джеймс Р., Вайнберг, Пол Н., Оппель, Эндрю Дж. SQL: полное руководство, 3-е изд. :
Пер. с англ. — М.: ООО "И.Д. Вильямс", 2015. — 960 с.
12. Дейт К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL. — Пер. с англ. —
СПб.: Символ-Плюс, 2010. — 480 с.

13. Карвин Б. Программирование баз данных SQL. Типичные ошибки и их устранение. — Рид
Групп, 2011. — 336 с.
14. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование,
поддержка. — СПб.: Питер, 2018. — 640 с.: ил.
15. Редмонд Эрик , Уилсон Джим Р. Семь баз данных за семь недель. Введение в современные
базы данных и идеологию NoSQL. — М: ДМК Пресс — 384с.
16. Фаулер, Мартин, Садаладж, Прамодкумар Дж. NoSQL: новая методология разработки
нереляционных баз данных. — Пер. с англ. — М.: ООО "И.Д. Вильямс", 2013. — 192 с.
17. Робинсон Ян, Вебер Джим, Эифрем Эмиль. Графовые базы данных: новые возможности для
работы со связанными данными. — 2-е изд. — М.: ДМК Пресс, 2016. — 256 с.
18. Карпентер Д., Хьюитт Э. Cassandra. Полное руководство. 2-е изд. — М.: ДМК Пресс, 2017. —
400 с.
19. Бэнкер Кайл. MongoDB в действии. — М.: ДМК Пресс, 2017. — 394с.
20. https://redis.io/documentation.