База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется [системой управления базами данных (СУБД)](#WhatIsDBMS). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL).

**Типы баз данных**

Существует множество различных типов баз данных. Выбор наилучшей базы данных для конкретной компании зависит от того, как она намеревается использовать данные.

**Реляционные базы данных**

* [Реляционные базы данных](https://www.oracle.com/cis/database/what-is-a-relational-database/" \o "https://www.oracle.com/cis/database/what-is-a-relational-database/) стали преобладать в 1980-х годах. Данные в реляционной базе организованы в виде таблиц, состоящих из столбцов и строк. Реляционная СУБД обеспечивает быстрый и эффективный доступ к структурированной информации.

**Объектно-ориентированные базы данных**

* Информация в объектно-ориентированной базе данных представлена в форме объекта, как в объектно-ориентированном программировании.

**Распределенные базы данных**

* Распределенная база данных состоит из двух или более частей, расположенных на разных серверах. Такая база данных может храниться на нескольких компьютерах.

**Хранилища данных**

* Будучи централизованным репозиторием для данных, хранилище данных представляет собой тип базы данных, специально предназначенной для быстрого выполнения запросов и анализа.

**Базы данных NoSQL**

* База данных [NoSQL](https://www.oracle.com/cis/database/nosql-cloud.html" \o "https://www.oracle.com/cis/database/nosql-cloud.html), или нереляционная база данных, дает возможность хранить и обрабатывать неструктурированные или слабоструктурированные данные (в отличие от реляционной базы данных, задающей структуру содержащихся в ней данных). Популярность баз данных NoSQL растет по мере распространения и усложнения веб-приложений.

**Графовые базы данных**

* Графовая база данных хранит данные в контексте сущностей и связей между сущностями.
* **Базы данных OLTP.** База данных OLTP — это база данных предназначенная для выполнения бизнес-транзакций, выполняемых множеством пользователей.

Это лишь некоторые из десятков типов баз данных, используемых в настоящее время. Другие, менее распространенные базы данных, предназначены для очень специфических научных, финансовых и иных задач. Помимо появления новых типов, базы данных развиваются в абсолютно новых направлениях — изменяются подходы к разработке технологий, происходят значительные сдвиги, такие как внедрение облачных технологий и автоматизации. В частности, в последнее время появились следующие базы данных.

**Базы данных с открытым исходным кодом**

* Такие базы данных имеют открытый исходный код и могут управляться средствами как SQL, так и NoSQL.

**Облачные базы данных**

* [Облачная база данных](https://www.oracle.com/cis/database/what-is-a-cloud-database/" \o "https://www.oracle.com/cis/database/what-is-a-cloud-database/) представляет собой набор структурированных или неструктурированных данных, размещенный на частной, публичной или гибридной платформе облачных вычислений. Существует два типа моделей облачных баз данных: традиционная база данных и база данных как услуга (DBaaS). В модели DBaaS административные задачи и обслуживание выполняются поставщиком облачных услуг.

**Многомодельные базы данных**

* Многомодельная база данных объединяет разные типы моделей баз данных в единую интегрированную серверную СУБД. Это означает, что она может содержать различные типы данных.

**Базы данных документов/JSON**

* [Базы данных документов](https://www.oracle.com/cis/autonomous-database/autonomous-json-database/" \o "https://www.oracle.com/cis/autonomous-database/autonomous-json-database/) предназначены для хранения, извлечения и обработки документоориентированной информации и предоставляют современный способ хранения данных в формате JSON, а не в виде строк и столбцов.

**Автономные базы данных**

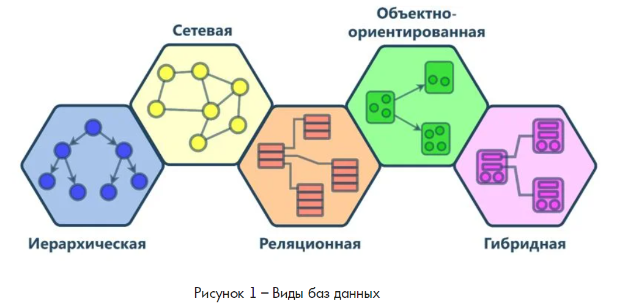
* Самоуправляемые базы данных (также называемые автономными) — это новейшие и самые революционные облачные базы данных, которые используют машинное обучение для автоматизации настройки, защиты, резервного копирования, обновления и других стандартных задач обслуживания, обычно выполняемых администраторами баз данных.

Изображение выглядит как стол

Автоматически созданное описание

**Виды баз данных**

По способу хранения данных и организации связей между ними, базы данных делятся на несколько видов: - иерархическая; - сетевая; - реляционная; - объектно-ориентированная; - гибридная.



Такое разнообразие обусловлено эволюционным характером развития систем управления базами данных, которое неразрывно связано с развитием технических средств ЭВМ. До 60-х годов прошлого века основным носителем информации была магнитная лента, чтение с которой можно было осуществлять только в последовательном режиме. С появлением магнитных дисков стал возможен прямой доступ к любой позиции файла базы данных и зародилась концепция навигационных хранилищ, к которым относятся иерархическая СУБД и сетевая СУБД. В навигационных хранилищах к элементам данных добавляются ссылки на другие данные, а путь к каждому элементу может быть представлен последовательностью переходов по ссылкам.   В 1970 году британский ученый, сотрудник IBM, Эдгар Франк Кодд предложил модель реляционной баз данных, в которой данные ищутся не по ссылкам, а по содержанию. Вычислительные мощности компьютеров того времени не позволяли в полной мере реализовать основные возможности этой СУБД, но уже в 1990-х годах реляционные системы заняли доминирующую позицию и удерживают ее до сих пор. В наши дни развиваются так называемые «постреляционные» модели баз данных, к которым относятся объектно-ориентированная и гибридная.

**Иерархическая база данных**

В иерархической модели данные объединены в древовидную структуру, в корневом элементе которой есть ссылки на узлы, расположенные на втором уровне иерархии, а в узлах второго – на узлы третьего, и так далее. Такая конструкция называется ориентированный граф.

Достоинство модели – простота доступа. Например, если корневой узел – это дом, узлы второго уровня – квартиры, третьего – жильцы, а программа должна найти человека, чтобы доставить письмо, то можно просто последовательно перемещаться по графу, читая адрес на конверте.   Недостатком является наличие у каждого элемента только одной связи с элементом более высокого уровня. Допустим, что связь между квартирой и жильцом появляется, если у жильца есть ключ. В такой схеме жилец дома не сможет уехать в отпуск и попросить соседа полить цветы, отдав ему ключ, так как в этом случае у соседа образуется вторая связь с более высоким уровнем, а это в иерархической модели невозможно.

**Сетевая база данных**

Сетевая модель по способу записи элементов похожа на иерархическую, но связи можно строить между любыми узлами графа, а не только «сверху-вниз».

Достоинством данной модели является возможность построения взаимосвязей между элементами, не ограничиваясь иерархической или иной строгой топологией. Недостатки: - сложность реализации; - невозможность изменять структуру связей без изменения всех использующих базу программ.

**Реляционная база данных**

Слово «реляционный» в названии этой модели произошло от латинского «relativus» – относительный. Структура взаимосвязей между элементами в реляционных базах данных представляет собой набор отношений, оформленных в табличной форме.

В этой модели данные организуются в табличной форме, где каждая запись – это отдельная строка, имеющая уникальный признак, называемый ключом. Строки принято называть записями, а столбцы – полями. Каждая таблица такой базы данных описывает некоторую сущность, поля таблицы – атрибуты этой сущности, а записи отвечают за конкретные экземпляры. Например, в таблице «квартира» полями могут быть: «номер», «этаж», «площадь», «количество комнат», а каждая запись – это конкретная квартира с конкретным номером, этажом и площадью. Ключом в данном случае может служить номер квартиры. Другой сущностью может быть человек, у которого будут свои атрибуты – «имя», «пол», «дата рождения». Ключом будет, например, имя. Для того чтобы «поселить» конкретного человека в конкретную квартиру, нужно связать между собой эти две сущности, для этого в специальной таблице связей делается запись о том, что одно из полей таблицы «человек» будет содержать ключ таблицы «квартира». Взаимодействие с базой данных осуществляется с помощью языка структурированных запросов SQL, который имеет схожий синтаксис во всех реляционных СУБД.   Достоинства реляционной модели данных: – независимость данных от программ их обработки; – простота понимания структуры таблиц; – наличие универсального языка запросов.      Недостатки: – не всегда данные удобно представлять в табличном виде; – дополнительный расход памяти; – низкая производительность в некоторых задачах.

**Объектно-ориентированная база данных**

В объектно-ориентированной модели данных сущности хранятся в виде совокупности объектов, с тем определением термина «объект», которое принято в объектно-ориентированном программировании: – объекты имеют сложную структуру, гораздо более сложную, чем просто строки в таблице; – большая часть данных внутри объекта недоступна извне напрямую, а только через специализированный интерфейс доступа; – объекты могут создаваться и уничтожаться; – поддерживается механизм наследования, когда объект-потомок может получать свойства объекта-предка.

Одной из главных причин появления этого типа баз данных стала следующая ситуация: самая распространенная парадигма программирования на данный момент – объектная, а самая распространенная модель данных – реляционная. Программистам приходится делать двойную работу: описывать взаимодействие объектов в программе на одном языке программирования, а потом выгружать эти объекты в таблицы, используя SQL. Объектно-ориентированная база данных – это попытка совместить два подхода. Главное достоинство такой СУБД – «бесшовная» интеграция с современными подходами к написанию программ. Главный недостаток – низкая производительность.

**Гибридная база данных**

Гибридная СУБД – это попытка объединить реляционную модель и объектно-ориентированный подход.

Внутри такая база содержит обычные файлы реляционной базы данных, обернутые в специальную оболочку, предоставляющую объектный интерфейс. Достоинством гибридных СУБД является сохранение привычной схемы проектирования базы на основе таблиц, без необходимости обращаться к ним напрямую. Недостаток заключается в непрозрачности механизма конвертирования запросов. Программисты, не знакомые с особенностями работы реляционных баз и не имеющие опыта создания SQL запросов, могут столкнуться с низким быстродействием или ошибками создания баз данных.

SQLite - это встроенная файловая СУБД, которая не требует установки или настройки. В свою очередь это означает, что приложение не запускается под отдельным серверным процессом, который необходимо запустить, остановить или настроить. Эта безсерверная архитектура обеспечивает совместимость базы данных на разных платформах.

Полная база данных SQL содержится в одном файле на диске, и все операции чтения и записи выполняются непосредственно в этом файле на диске. Поскольку данные записываются непосредственно обратно в файл диска, SQLite придерживается атрибута ACID для защиты транзакций от сбоев выделения памяти и ошибок ввода-вывода диска, вызванных непредвиденными сбоями системы или сбоями питания.

**Преимущества и варианты использования**

Библиотека SQLite является одной из самых компактных библиотек в этом списке, где размер библиотеки может быть меньше 600 КБ. Из-за своего небольшого размера и характеристик СУБД, он очень подходит для Интернета вещей и встроенных устройств.

Некоторые другие хорошие примеры использования включают веб-сайты с низким и средним трафиком (около 100 тыс. Запросов в день), тестирование и внутреннюю разработку, использование Tcl или Python для анализа данных и в образовательных целях (это легко настроить и может быть использовано для обучения SQL концепциям учащихся).

Основным преимуществом SQLite является то, как его можно использовать в качестве дополнительного решения для клиент-серверных корпоративных СУБД. Например, он может локально кэшировать данные из клиент-серверной СУБД, тем самым уменьшая задержку запроса и поддерживая конечное приложение активным, когда корпоративная СУБД прерывается.

**Недостатки**

Основным недостатком системы SQLite является отсутствие многопользовательских функций, которые можно найти в зрелых системах СУБД, таких как MySQL и PostgreSQL. Это приводит к отсутствию детального контроля доступа, удобной системы управления пользователями и функций безопасности, помимо самого зашифрованного файла базы данных. Это является основным недостатком при разработке многопользовательских приложений, таких как приложения CRM и SaaS, и обычно не пользуется популярностью при создании многопользовательских или многопользовательских приложений.

Другим большим недостатком SQLite является то, что он обрабатывает сериализованные операции записи. Это может быть основным узким местом для приложений, требующих параллелизма. Поскольку SQLite является файловой СУБД, это может вызвать проблемы с производительностью для больших наборов данных из-за ограничений файловой системы. В этих случаях целесообразно выбирать клиент-серверные базы данных (такие как MySQL и PostgreSQL), особенно при работе с большими наборами данных, такими как большие данные.

Стоит также отметить, что у любого крупного облачного провайдера отсутствуют продукты для баз данных как услуга (DBaaS). С появлением общедоступного облака разработчики и команды DevOps, использующие службы PaaS (такие как DBaaS), стали обычным явлением. Отсутствие управляемых услуг у ведущих поставщиков общедоступного облака означает, что администратор БД должен выполнять общие задачи управления, что увеличивает затраты на OPEX.

**Владение, поддержка и основные клиенты**

SQLite состоит из группы[разработчик](https://www.sqlite.org/crew.html" \o "https://www.sqlite.org/crew.html)База данных с открытым исходным кодом поддерживается. Для поддержки сообщества это обеспечивает[Публичная рассылка](http://sqlite.org:8080/cgi-bin/mailman/listinfo/sqlite-users)И предлагаем оплату[Профессиональная поддержка](https://www.sqlite.org/prosupport.html), Некоторые из основных клиентов SQLite - это Facebook, Google и Apple.

**MySQL**

**Обзор и особенности**

MySQL это[Самый популярный](https://db-engines.com/en/ranking)Одна из открытых и масштабных систем СУБД. В отличие от SQLite, он использует архитектуру сервер / клиент, состоящую из многопоточного сервера SQL. Эта многопоточная функция MySQL обеспечивает более высокую производительность, поскольку потоки ядра могут легко использовать несколько процессоров. База данных написана на C и C ++ и поддерживает[Различные платформы,](https://www.mysql.com/support/supportedplatforms/database.html)Такие как операционная система Windows Server и дистрибутивы Linux, такие как RHEL 7 и Ubuntu. Он также следует системе ACID для обеспечения согласованности транзакций и предоставляет различные коннекторы и API, такие как C, C ++, Java, PHP и т. Д.

Масштабируемость, безопасность и репликация - вот некоторые ключевые функции, которые делают MySQL одним из самых популярных решений в корпоративных приложениях:

* Функции безопасности включают MySQL Access Privilege System, которая обеспечивает аутентификацию пользователя,[Система управления учетными записями пользователей](https://dev.mysql.com/doc/refman/8.0/en/user-account-management.html)И зашифрованное соединение с использованием SSL.
* MySQL обеспечивает репликацию с главного сервера на подчиненный сервер и с главного сервера на главный сервер. Это очень полезно при расширении операций чтения. Его можно использовать в качестве решения для резервного копирования и даже при сбое в случае сбоя. MySQL также имеет больше функций[Коммерческие продукты](https://www.mysql.com/products/)。
* Например, MySQL Enterprise Edition имеет дополнительные функции, такие как MySQL Transparent Data Encryption (TDE), MySQL Enterprise Backup и MySQL Document Storage.
* MySQL также предоставляет встроенную многопоточную библиотеку, чтобы обеспечить меньшую площадь для встраиваемых и IoT-систем.

**Преимущества и варианты использования**

В дополнение к наличию нескольких корпоративных функций, еще одним важным отличием MySQL от SQLite является поддержка многопользовательскими функциями MySQL. Вместе с функциональностью и масштабируемостью предприятия это делает его идеальным выбором для распределенных приложений.

С точки зрения пропускной способности и производительности, MySQL имеет преимущества перед PostgreSQL и может использоваться для простых операций чтения. По сравнению с PostgreSQL, он также проще в установке и использовании и имеет более широкое сообщество.

**Недостатки**

Когда MySQL перемещает старые данные в отдельную область, называемую сегментом отката, массовая INSERT отрицательно влияет на производительность. Это основной момент PostgreSQL. Он также не очень хорошо работает для длительных SELECT и наиболее подходит для небольших SELECT, особенно тех, которые охватывают кластерные индексы. Некоторые другие недостатки включают в себя отсутствие полнотекстового поиска и медленное одновременное чтение и запись.

**Владение, поддержка и основные клиенты**

MySQL принадлежит и поддерживается Oracle. Поддерживать поддержку сообщества через форумы и через[покупка](https://www.mysql.com/products/)Коммерческие продукты получили важную поддержку. Некоторые из основных клиентов MySQL - это Facebook, GitHub и YouTube.

**PostgreSQL**

**Обзор и особенности**

PostgreSQL - это СУБД с открытым исходным кодом, в которой особое внимание уделяется масштабируемости и совместимости стандартов. Как и MySQL, PostgreSQL использует модель базы данных клиент / сервер и серверный процесс, который обрабатывает взаимодействие с клиентом и управляет файлами базы данных и операциями, которые называются процессами.

PostgreSQL обрабатывает параллельные клиентские сессии, создавая («разветвляя») новые процессы для каждого соединения. Этот процесс отделен от основного процесса и создается и уничтожается в течение жизненного цикла клиентского соединения. Postgres написан на C и соответствует стандарту ACID, поддерживает функции и хранимые процедуры. В отличие от MySQL, PostgreSQL поддерживает материализованные представления (кэшированные представления) для частого и быстрого доступа к большим и активным таблицам.

Как и MySQL, PostgreSQL также обладает некоторыми расширенными функциями, такими как безопасность и репликация. PostgreSQL использует синхронную репликацию между первичной и вторичной базами данных. В дополнение к обеспечению контроля доступа пользователя, контроля доступа на основе хоста и аутентификации пользователя, сам PostgreSQL также обеспечивает функцию шифрования связи клиент-сервер с использованием SSL. Полное соответствие ACID присуще PostgreSQL и существует в программах NDB Cluster InnoDB и MySQL.

PostgreSQL использует технику, называемую многоверсионным управлением параллелизмом или MVCC, для обеспечения согласованности данных при одновременном доступе к данным. Этот метод лучше, чем использование только блокировок для параллелизма, поскольку он минимизирует конфликт блокировок в многопользовательской среде, что значительно повышает производительность. Для обратной совместимости или приложений, требующих классической технологии блокировки, PostgreSQL также позволяет использовать технологию блокировки таблиц и строк для обеспечения параллелизма. Напротив, MySQL поддерживает MVCC только в экземплярах InnoDB.

**Преимущества и варианты использования**

**Недостатки**

Во время частых обновлений вы можете увидеть еще один большой недостаток: поскольку кластерные индексы не поддерживаются, PostgreSQL окажет огромное негативное влияние на производительность по сравнению с базами данных MySQL.

**Владение, поддержка и основные клиенты**

PostgreSQL является открытым исходным кодом, и исходный код публикуется и поддерживается глобальной командой разработчиков PostgreSQL. PostgreSQL имеет два варианта поддержки сообщества и коммерции. Сообщество поддерживает принятие[список рассылки](https://www.postgresql.org/list/)Форма, вы также можете найти список поставщиков коммерческой поддержки. Хотя PostgreSQL имеет меньшую долю рынка по сравнению с MySQL, у него впечатляющий список клиентов, таких как AWS RedShift, Instagram, ViaSat и Cloudera.

**Microsoft SQL Server** — [система управления реляционными базами данных (РСУБД)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Основной используемый язык запросов — [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL), создан совместно Microsoft и [Sybase](https://ru.wikipedia.org/wiki/Sybase). Transact-SQL является реализацией стандарта [ANSI](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2)/[ISO](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8) по структурированному языку запросов ([SQL](https://ru.wikipedia.org/wiki/SQL)) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

**Плюсы SQL Server**

* СУБД масштабируется, поэтому работать с ней можно на портативных ПК или мощной мультипроцессорной технике. Процессор может одновременно обрабатывать большой объем запросов.
* Размер страниц – до 8 кб, поэтому данные извлекаются быстро, подробную и сложную информацию хранить удобнее. Система позволяет обрабатывать транзакции в интерактивном режиме, есть динамическая блокировка.
* Рутинные административные задачи автоматизированы: это управление блокировками, памятью, редактура размеров файлов. У системы продуманы настройки, можно создать профили пользователей.
* Реализован поиск по фразам, тексту, словам, можно создавать ключевые индексы.
* В SQL Server есть репликации через интернет, предусмотрена синхронизация. Есть полноценный веб-ассистент для форматирования страниц.
* В систему интегрирован сервер интерактивного анализа для принятия решений, создания корпоративных отчетов. Есть службы преобразования информации.
* Запросы можно формулировать на английском языке, без программирования.
* СУБД поддерживает работу с другими продуктами Microsoft: Access, MS Excel.

**Недостатки**

* Зависимость от операционной среды: СУБД работает только с системой Windows.
* Высокая цена программы.

**SQL** - простой язык программирования, который имеет немного команд и которой может научиться любой желающий. Расшифровывается как Structured Query Language - язык структурированных запросов, который был разработан для работы с БД, а именно, чтобы получать /добавлять /изменять данные, иметь возможность обрабатывать большие массивы информации и быстро получать структурированную и сгруппированную информацию. Есть много вариантов языка SQL, но у них всех основные команды почти одинаковы. Также существует и много СУБД, но основными из них являются: Microsoft Access, Microsoft SQL Server, MySQL, Oracle SQL, IBM DB2 SQL, PostgreSQL та Sybase Adaptive Server SQL. Чтобы работать с SQL кодом, нам понадобится одна из вышеперечисленных СУБД.

SQL как и другие языки программирования имеет свои команды (операторы), с помощью которых отдаются инструкции для выборки данных.

CRUD — акроним, обозначающий четыре базовые функции, используемые при работе с базами данных[1]: создание (англ. create), чтение (read), модификация (update), удаление (delete). Введён Джеймсом Мартином (англ. James Martin) в 1983 году[2] как стандартная классификация функций по манипуляции данными.

В SQL этим функциям, операциям соответствуют операторы

**Insert** (создание записей)- insert into <название таблицы> ([<Имя столбца>, ... ]) values (<Значение>,...)

-**Select** (чтение записей)

SELECT (от англ. select — «выбрать») — оператор запроса (DML/DQL) в языке SQL, возвращающий набор данных (выборку) из базы данных.

Оператор возвращает ноль или более строк. Список возвращаемых столбцов задается в части оператора, называемой предложением SELECT. Поскольку SQL является декларативным языком, запрос SELECT определяет лишь требования к возвращаемому набору данных, но не является точной инструкцией по их вычислению. СУБД транслирует запрос SELECT во внутренний план исполнения («query plan»), который может различаться даже для синтаксически одинаковых запросов и от конкретной СУБД.

Оператор SELECT состоит из нескольких предложений (разделов):

    SELECT определяет список возвращаемых столбцов (как существующих, так и вычисляемых), их имена, ограничения на уникальность строк в возвращаемом наборе, ограничения на количество строк в возвращаемом наборе;

    FROM задаёт табличное выражение, которое определяет базовый набор данных для применения операций, определяемых в других предложениях оператора;

    WHERE задает ограничение на строки табличного выражения из предложения FROM;

    GROUP BY объединяет ряды, имеющие одинаковое свойство с применением агрегатных функций

    HAVING выбирает среди групп, определённых параметром GROUP BY

    ORDER BY задает критерии сортировки строк; отсортированные строки передаются в точку вызова.

Оператор SELECT имеет следующую структуру:

SELECT

  [DISTINCT | DISTINCTROW | ALL]

  select\_expression,...

FROM table\_references

[WHERE where\_definition]

[GROUP BY {unsigned\_integer | col\_name | formula}]

[HAVING where\_definition]

[ORDER BY {unsigned\_integer | col\_name | formula} [ASC | DESC], ...]

Предложения оператора

SELECT

Предложение SELECT оператора SELECT предназначено для определения результирующего набора столбцов, получаемого после вычисления табличного выражения в предложении FROM и группировки в результате GROUP BY (при наличии). Предложение SELECT реализует операцию проекции, то есть указание подмножества столбцов из таблиц табличного выражения, а также операцию переименования столбцов и операцию добавления новых вычислимых столбцов.

FROM

Предложение FROM используется для вычисления базового табличного выражения, которое затем используется остальными предложениями оператора SELECT.

WHERE

Предложение [[WHERE (SQL)|WHERE]] используется для определения, какие строки должны быть выбраны из табличного выражения в предложении FROM.

GROUP BY

[[GROUP BY (SQL)|GROUP BY]] — необязательное предложение оператора SELECT, для группировки строк по результатам агрегатных функций (MAX, SUM, AVG, …).

Необходимо, чтобы в предложении SELECT были заданы только требуемые в выходном потоке столбцы, перечисленные в GROUP BY и/или агрегированные значения. Распространённая ошибка — указание в предложении SELECT столбца, пропущенного в GROUP BY.

HAVING

HAVING — необязательное предложение оператора SELECT для отбора групп, получающихся в результате GROUP BY.

При указании HAVING <условия> можно указывать условия на столбцах, указанных в GROUP BY, и значениях агрегатных функций, вычисленных для каждой группы, образованной GROUP BY.

ORDER BY

ORDER BY — необязательное предложение операторов SELECT и UNION, который означает что операторы SELECT, UNION возвращают набор строк, отсортированных по значениям одного или более столбцов. Его можно применять как к числовым столбцам, так и к строковым. В последнем случае, сортировка будет происходить по алфавиту.

Использование предложения ORDER BY является единственным способом отсортировать результирующий набор строк. Без этого предложения СУБД может вернуть строки в любом порядке. Если упорядочение необходимо, ORDER BY должен присутствовать в SELECT, UNION.

Сортировка может производиться как по возрастанию, так и по убыванию значений.

    Параметр ASC (по умолчанию) устанавливает порядок сортировки по возрастанию, от меньших значений к большим.

    Параметр DESC устанавливает порядок сортировки по убыванию, от больших значений к меньшим.

**-Update** (редактирование записей) –

UPDATE — оператор языка SQL, позволяющий обновить значения в заданных столбцах таблицы.

Синтаксис

UPDATE [top(x)] <объект>

SET <присваивание1 [, присваивание2, ...]>

[WHERE <условие>];

    top(x) — команда выполнится только х раз

    <объект> — объект, над которым выполняется действие (таблица или представление)

    <присваивание> — присваивание, которое будет выполняться при каждом выполнении условия <условие>, или для каждой записи, если отсутствует раздел where

    <условие> — условие выполнения команды

    SET — после ключевого слова должен идти список полей таблицы, которые будут обновлены и непосредственно сами новые значения в виде

    имя поля="значение"

**-Delete** (удаление записей).

DELETE — в языках, подобных MDl, SQL-операция удаления записей из таблицы. Критерий отбора записей для удаления определяется выражением where. В случае, если критерий отбора не определён, выполняется удаление всех записей.

    В СУБД, поддерживающих триггеры, операция Delete может вызывать их срабатывание;

    При наличии на таблице внешних ключей все дочерние к удаляемым записи в подчинённых таблицах также должны быть удалены для обеспечения ссылочной целостности;

    В СУБД, поддерживающих транзакции, выполнение операции Delete должно быть подтверждено (COMMIT), либо опровергнуто (ROLLBACK) вызовом соответствующих операций.

Синтаксис

Общий синтаксис команды:

DELETE FROM <Имя Таблицы> WHERE <Условие отбора записей>

Последствием выполнения такой команды будет удаление тех строк из таблицы <Имя Таблицы>, которые соответствуют условию <Условие отбора записей>. При этом никакого результата команда не возвращает и, следовательно, не может быть использована в качестве параметра в команде SELECT.

**DB Testing**

Есть несколько причин, по которым проводится тестирование базы данных. Необходимо выполнить проверку целостности данных, проверку достоверности и целостности данных в базе данных, так как бэкэнд-система отвечает за хранение данных и доступ к ней имеет многоцелевое назначение.

Ниже приведены некоторые общие причины для тестирования базы данных –

    Чтобы облегчить сложность обращений к бэкэнду базы данных, разработчики расширяют использование представлений и хранимых процедур.

    Эти хранимые процедуры и представления содержат важные задачи, такие как вставка сведений о клиенте (имя, контактная информация и т. Д.) И данных о продажах. Эти задачи необходимо протестировать на нескольких уровнях.

    Тестирование черного ящика на внешнем интерфейсе важно, но затрудняет выявление проблемы. Тестирование в бэкэнд-системе повышает надежность данных. Вот почему тестирование базы данных выполняется на внутренней системе.

    В базе данных данные поступают из нескольких приложений, и существует вероятность того, что вредоносные или неправильные данные хранятся в базе данных. Поэтому необходимо регулярно проверять компоненты базы данных. Кроме того, целостность и согласованность данных должны регулярно проверяться.

Чтобы облегчить сложность обращений к бэкэнду базы данных, разработчики расширяют использование представлений и хранимых процедур.

Эти хранимые процедуры и представления содержат важные задачи, такие как вставка сведений о клиенте (имя, контактная информация и т. Д.) И данных о продажах. Эти задачи необходимо протестировать на нескольких уровнях.

Тестирование черного ящика на внешнем интерфейсе важно, но затрудняет выявление проблемы. Тестирование в бэкэнд-системе повышает надежность данных. Вот почему тестирование базы данных выполняется на внутренней системе.

В базе данных данные поступают из нескольких приложений, и существует вероятность того, что вредоносные или неправильные данные хранятся в базе данных. Поэтому необходимо регулярно проверять компоненты базы данных. Кроме того, целостность и согласованность данных должны регулярно проверяться.

Кроме этого при других видах тестирования БД поможет:

1. Создать тестовую среду
2. Создать тестовые данные
3. Выявить причину дефекта

Процесс тестирования базы данных аналогичен тестированию других приложений. Тестирование БД может быть описано с помощью ключевых процессов, приведенных ниже.

* Настройте среду
* Запустить тест
* Проверьте результат теста
* Подтвердить в соответствии с ожидаемыми результатами
* Сообщить о результатах соответствующим заинтересованным сторонам

Различные операторы SQL используются для разработки тестовых случаев. Наиболее распространенным оператором SQL, который используется для тестирования БД, является оператор **Select** . Помимо этого, могут использоваться различные операторы DDL, DML, DCL.

**Пример** – создание, вставка, выбор, обновление и т. Д.

## Этапы тестирования базы данных

Тестирование БД не является утомительным процессом и включает в себя различные этапы жизненного цикла тестирования базы данных в соответствии с процессами тестирования.

Ключевые этапы тестирования базы данных:

* Проверка исходного состояния
* Тест ран
* Подтверждение результата согласно ожидаемому результату
* Генерация результатов