# Ключевые особенности обеих технологий и их главные отличия

## SQL и реляционные базы данных

Изобретена в 1970 году инженером IBM Эдгаром Коддом.   
В реляционной базе данных информация представлена в виде таблиц содержащих столбцы и строки. Каждая таблица, имеет несколько колонок или **атрибутов** в которых содержатся категории данных и каждая строка или **запись (record)** содержит в себе значения по каждой категории (столбцу). В каждой таблице имеется первичный ключ - столбец уникальных значений однозначно определяющий отдельные записи Отношения между таблицами –ключевая особенность реляционных баз данных, давшая им имя, реализована с помощью **внешних** **ключей (foreign keys)** – полей в таблице привязанных к столбцам в других таблицах. Два ограничения накладываются на первичные и внешние ключи:

Entity integrity: значения внешних и первичных ключей в таблице должны быть уникальные и ненулевые.  
Referential integrity: требование чтобы каждое значение внешнего ключа также присутствовало в таблице, к которой он относится, в виде первичного ключа

### Типы баз данных

Flat file (плоский файл) – представляет собой одну таблицу данных, обычно в текстовом формате (CSV), с возможностью для пользователя задать параметры данных, такие как названия столбцов и типов данных.  
Объектно-реляционные БД составлены из реляционных и объектно-ориентированных СУБД и совмещают оба подхода к обращению с данными, структура данных представлена как в обычных реляционных СУБД, запросы непосредственно к БД делаются на языке запросов, таких как SQL (Structured Query Language). При этом обращение с данными на верхнем уровне происходит на объектно-ориентированном языке программирования.

### Преимущества реляционных БД

Главное преимущество реляционных БД это возможность для пользователя легко классифицировать и структурировать данные, которые затем могут быть получены с помощью стандартизированного языка запросов. Другие преимущества:

**Точность:** данные не повторяются  
**Гибкость:** сложные запросы выполняются просто с пользовательской точки зрения  
**Кооперация:** несколько пользователей могут одновременно работать с одной БД  
**Надежность:** реляционные БД существуют уже давно и имеют огромную пользовательскую и теоретическую базу.  
**Безопасность:** доступ к данным в реляционных СУБД может быть ограничена по пользователям

## Масштабирование в реляционных БД

Реляционные БД проще масштабировать вверх, и обычно самые распространенные СУБД могут обрабатывать огромное количество таблиц с данными без необходимости горизонтального масштабирования. Все что требуется это докидывать больше и больше вычислительной мощности. Горизонтальное масштабирование все еще возможно но требует немалых усилий (partitioning, clastering, sharding etc)

## Нормализация реляционных БД

Целью нормализации реляционной базы данных является устранение недостатков структуры базы данных, приводящих к избыточности, которая, в свою очередь, потенциально приводит к различным аномалиям и нарушениям целостности данных.

общее назначение процесса нормализации заключается в следующем:

* исключение некоторых типов избыточности;
* устранение некоторых аномалий обновления;
* разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;
* упрощение процедуры применения необходимых ограничений целостности.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

## ACID принципы

В информатике акроним **ACID** описывает требования к транзакционной системе (например, к СУБД), обеспечивающие наиболее надёжную и предсказуемую её работу. Требования ACID были в основном сформулированы в конце 70-х годов Джимом Греем.

**Атомарность (Atomicity)** гарантирует, что никакая [транзакция](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) не будет зафиксирована в системе частично. Будут либо выполнены все её подоперации, либо не выполнено ни одной. Поскольку на практике невозможно одновременно и атомарно выполнить всю последовательность операций внутри транзакции, вводится понятие «отката» ([rollback](https://ru.wikipedia.org/wiki/Rollback" \o "Rollback)): если транзакцию не удаётся полностью завершить, результаты всех её до сих пор произведённых действий будут отменены и система вернётся во «внешне исходное» состояние — со стороны будет казаться, что транзакции и не было.

**Согласованность (Consistency) -** транзакция, достигающая своего нормального завершения (EOT — end of transaction, завершение транзакции) и, тем самым, фиксирующая свои результаты, сохраняет согласованность базы данных. Другими словами, каждая успешная транзакция по определению фиксирует только допустимые результаты. Это условие является необходимым для поддержки четвёртого свойства.

**Изолированность (Isolation) -** во время выполнения транзакции параллельные транзакции не должны оказывать влияния на её результат.

**Долговечность (Durability)** – независимо от проблем на нижних уровнях (к примеру, обесточивание системы или сбои в оборудовании) изменения, сделанные успешно завершённой транзакцией, должны остаться сохранёнными после возвращения системы в работу. Другими словами, если пользователь получил подтверждение от системы, что транзакция выполнена, он может быть уверен, что сделанные им изменения не будут отменены из-за какого-либо сбоя.

## 12 правил Кодда

**Правило 0: Основное правило** (*Foundation Rule*):

Система, которая рекламируется или позиционируется как реляционная система управления базами данных, должна быть способна управлять базами данных, используя исключительно свои реляционные возможности.

**Правило 1: Информационное правило** (*The Information Rule*):

Вся информация в реляционной базе данных на логическом уровне должна быть явно представлена единственным способом: значениями в таблицах.

**Правило 2: Гарантированный доступ к данным** (*Guaranteed Access Rule*):

В реляционной базе данных каждое отдельное (атомарное) значение данных должно быть логически доступно с помощью комбинации имени таблицы, значения первичного ключа и имени столбца.

**Правило 3: Систематическая поддержка отсутствующих значений** (*Systematic Treatment of Null Values*):

Неизвестные, или отсутствующие значения NULL, отличные от любого известного значения, должны поддерживаться для всех типов данных при выполнении любых операций. Например, для числовых данных неизвестные значения не должны рассматриваться как нули, а для символьных данных — как пустые строки.

**Правило 4: Доступ к словарю данных в терминах реляционной модели** (*Active On-Line Catalog Based on the Relational Model*):

Словарь данных должен сохраняться в форме реляционных таблиц, и СУБД должна поддерживать доступ к нему при помощи стандартных языковых средств, тех же самых, которые используются для работы с реляционными таблицами, содержащими пользовательские данные.

**Правило 5: Полнота подмножества языка** (*Comprehensive Data Sublanguage Rule*):

Система управления реляционными базами данных должна поддерживать хотя бы один реляционный язык, который

(а) имеет линейный синтаксис,

(б) может использоваться как интерактивно, так и в прикладных программах,

(в) поддерживает операции определения данных, определения представлений, манипулирования данными (интерактивные и программные), ограничители целостности, управления доступом и операции управления транзакциями (begin, commit и rollback).

**Правило 6: Возможность изменения представлений** (*[View](https://ru.wikipedia.org/wiki/View" \o "View) Updating Rule*):

Каждое представление должно поддерживать все операции манипулирования данными, которые поддерживают реляционные таблицы: операции выборки, вставки, изменения и удаления данных.

**Правило 7: Наличие высокоуровневых операций управления данными** (*High-Level Insert, Update, and Delete*):

Операции вставки, изменения и удаления данных должны поддерживаться не только по отношению к одной строке реляционной таблицы, но и по отношению к любому множеству строк.

**Правило 8: Физическая независимость данных** (*Physical Data Independence*):

Приложения не должны зависеть от используемых способов хранения данных на носителях, от аппаратного обеспечения компьютеров, на которых находится реляционная база данных.

**Правило 9: Логическая независимость данных** (*Logical Data Independence*):

Представление данных в приложении не должно зависеть от структуры реляционных таблиц. Если в процессе нормализации одна реляционная таблица разделяется на две, представление должно обеспечить объединение этих данных, чтобы изменение структуры реляционных таблиц не сказывалось на работе приложений.

**Правило 10: Независимость контроля целостности** (*Integrity Independence*):

Вся информация, необходимая для поддержания целостности, должна находиться в словаре данных. Язык для работы с данными должен выполнять проверку входных данных и автоматически поддерживать целостность данных.

**Правило 11: Независимость от расположения** (*Distribution Independence*):

База данных может быть распределённой, может находиться на нескольких компьютерах, и это не должно оказывать влияния на приложения. Перенос базы данных на другой компьютер не должен оказывать влияния на приложения.

**Правило 12: Согласование языковых уровней** (*The Nonsubversion Rule*):

Если используется низкоуровневый язык доступа к данным, он не должен игнорировать правила безопасности и правила целостности, которые поддерживаются языком более высокого уровня.

## Когда выбирать SQL и NoSQL

Если все аспекты, связанные с данными хорошо задокументированы и определены то SQL лучший выбор. Также если предполагается работа с чувствительными к изменениям данными и финансовыми в частности, также рекомендуется использовать реляционные БД и транзакции.  
NoSQL хороший выбор для быстрого старта разработки приложения на этапе когда еще не совсем определено какие поля у каких сущностей будут, также NoSQL базы данных легче масштабировать горизонтально. Еще один минус NoSQL баз данных это отсутствие стандартизации и вкатываться в новую NoSQL СУБД придется с нуля