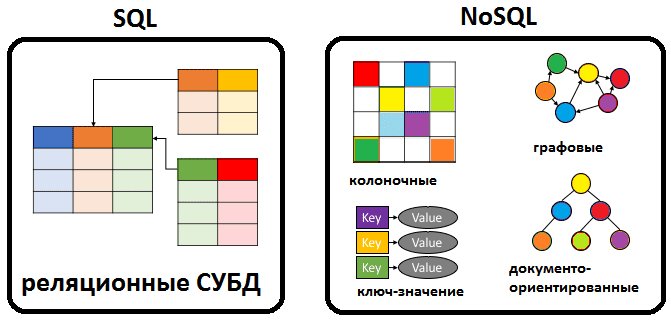
**База данных** - это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. Управляются с помощью СУБД (MySQL, PostgreSQL…)



**Primary key** — особенное поле в SQL-таблице, которое позволяет однозначно идентифицировать каждую запись в ней.

**Foreign key** - столбец (или несколько столбцов), ссылающийся на первичный ключ другой таблицы.

**Виды связей**

* Один к одному
* Один ко многим
* Многие ко многим

У первых двух могут быть обязательная связь (дочерняя таблица существует только с родительской) и необязательная.

**JPA (Java Persistence API)** - это спецификация Java EE и Java SE, описывающая систему управления сохранением java объектов в таблицы реляционных баз данных в удобном виде. Сама Java не содержит реализации JPA, однако есть существует много реализаций данной спецификации от разных компаний (открытых и нет). Это не единственный способ сохранения java объектов в базы данных (ORM систем), но один из самых популярных в Java мире.

**Hibernate** - одна из самых популярных открытых реализаций последней версии спецификации (JPA 2.1). Даже скорее самая популярная, почти стандарт де-факто. То есть JPA только описывает правила и API, а Hibernate реализует эти описания, впрочем у Hibernate (как и у многих других реализаций JPA) есть дополнительные возможности, не описанные в JPA (и не переносимые на другие реализации JPA).

[**Object-relational mapping**](https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping) (ORM) - это способ (он же шаблон проектирования) доступа к реляционной базе данных с помощью объектно-ориентированного языка (например, Java).

**Spring Data** — дополнительный удобный механизм для взаимодействия с сущностями базы данных, организации их в репозитории, извлечение данных, изменение, в каких то случаях для этого будет достаточно объявить интерфейс и метод в нем, без имплементации.

Основное понятие в Spring Data — это репозиторий. Это несколько интерфейсов которые используют JPA Entity для взаимодействия с ней. Так например интерфейс  
public interface CrudRepository<T, ID extends Serializable> extends Repository<T, ID>  
обеспечивает основные операции по поиску, сохранения, удалению данных (CRUD операции)

**application.properties** и **application.yml** используются для конфигурации приложений. Они предоставляют способы определения настроек, таких как параметры подключения к базе данных, порты, уровень логирования и другие параметры, необходимые для работы приложения.

Отличия в формате. properties – ключ значение, yml – язык разметки YAML:

server:

port: 8080

spring:

datasource:

url: jdbc:mysql://localhost:3306/mydatabase

logging:

level:

root: info

**Аннотации Spring**

* @Entity — Указывает, что данный бин (класс) является сущностью
* @Table — указывает на имя таблицы, которая будет отображаться в этой сущности
* @Column — указывает на имя колонки, которая отображается в свойство сущности
* @OneToMany — указывает на связь один ко многим. Применяется с другой стороны от сущности с @ManyToOne
* @ManyToMany — связь многие ко многим.
* @JoinColumn  — применяется когда внешний ключ находится в одной из сущностей. Может применяться с обеих сторон взаимосвязи. Но рекомендуется применять в сущности, которая является владельцем физической информации (**обычно** **сторона @ManyToOne**). ManyToOne (часто) является стороной-владельцем в двунаправленных связях и таким образом противоположная сторона использует @OneToMany(mappedBy=..).

**Каскадирование –** процесс**, когда мы выполняем какое-либо действие над целевым объектом, то же самое действие будет применено к связанному объекту.**

* *ALL*
* *PERSIST*
* *MERGE*
* *REMOVE*
* *REFRESH*
* *DETACH*

Если **параметр fetch равен EAGER**, то при загрузке родительской сущности будут загружены и все ее дочерние сущности. Кроме того, Hibernate постарается сделать это одним SQL-запросом, сгенерировав здоровенный запрос и сразу получив все данные.

Если параметр **fetch принимает значение LAZY**, то при загрузке родительской сущности, дочерняя сущность загружена не будет. Вместо нее будет создан proxy-объект. С помощью этого proxy-объекта Hibernate будет отслеживать обращение к этой дочерней сущности и при первом обращении загрузит ее в память.

1. "Open Session in View" (OSIV) - это концепция, которая была впервые представлена в мире Java-фреймворков, включая Spring, и она часто связана с использованием Hibernate в контексте работы с базами данных.
2. Сеанс (Session): В контексте Hibernate (фреймворк для работы с базами данных), сеанс представляет собой временное соединение между приложением и базой данных. Он обеспечивает механизм для выполнения операций базы данных, таких как сохранение, обновление, удаление и запросы.
3. Open Session in View (OSIV): Эта концепция предлагает держать сеанс (Hibernate Session) открытым в течение всего жизненного цикла запроса пользователя, начиная с момента обработки запроса в контроллере и заканчивая отображением данных на веб-странице. Суть в том, чтобы держать сеанс открытым в течение всего времени обработки запроса и даже при рендеринге представления.
4. Проблема, которую решает OSIV: Когда вы работаете с Lazy Loading в Hibernate (отложенной загрузкой), данные могут быть запрошены из базы данных только тогда, когда они действительно нужны. Если сеанс закрывается раньше, чем данные требуется отобразить на веб-странице (например, в JSP или Thymeleaf), это может привести к исключениям LazyInitializationException. OSIV предотвращает эти проблемы, держа сеанс открытым до завершения цикла запроса-ответа
5. Как реализовать OSIV в Spring Boot: Для включения Open Session in View в Spring Boot, вы можете добавить свойство spring.jpa.open-in-view в файл application.properties или application.yml:

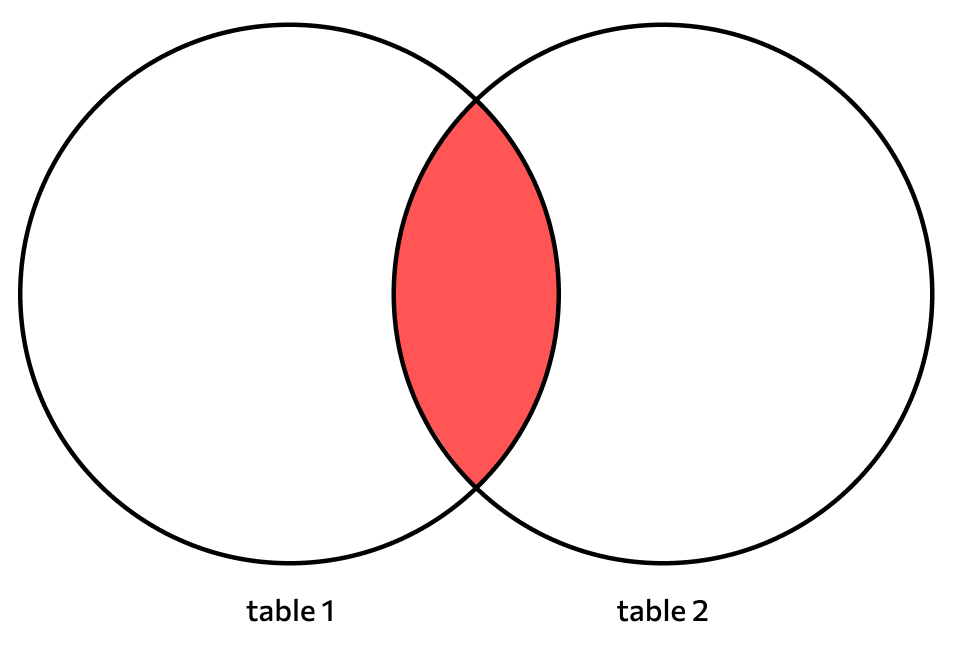
Для чего нужен оператор JOIN

* Для «умного» поиска по таблицам в рамках одной базы данных. Использование разных режимов работы оператора помогает отфильтровать сведения по определенным критериям.
* Для быстрого отсечения информации, которая не нужна в рамках конкретного запроса.
* Для соединения двух таблиц в одну, например, чтобы отправить полученную информацию единым блоком.

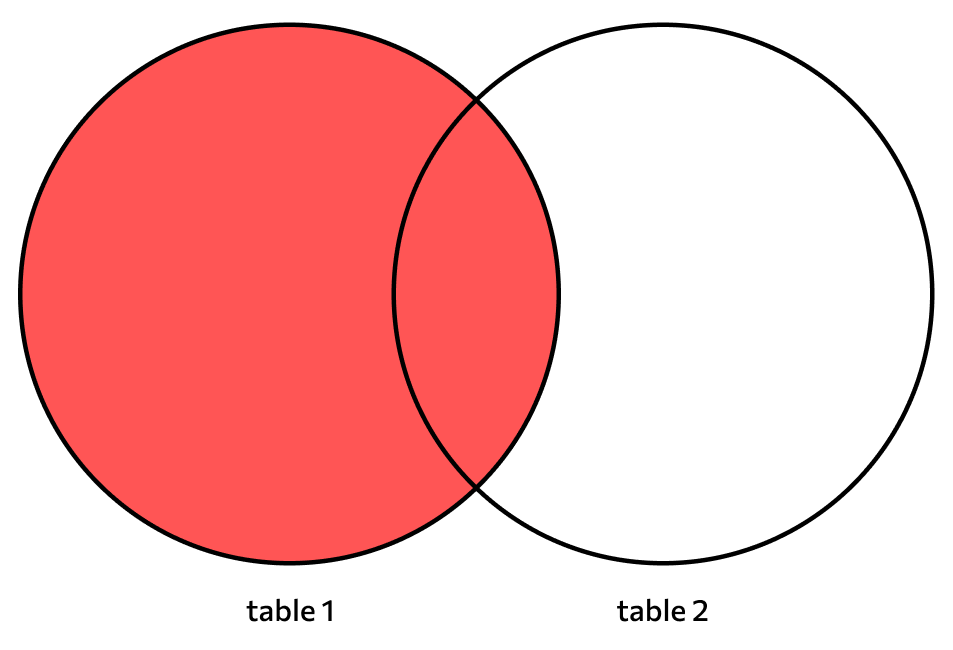
Оператору передаются таблицы, которые нужно объединить, и критерий для объединения — логическое выражение, которое называется ключом

Найденные результаты объединяются в одну таблицу. Две соответствующие друг другу разные строки преобразуются в одну — это важнейшее условие работы JOIN. Строки чаще всего из разных таблиц, но это не обязательно. Главная особенность JOIN — объединение двух объектов в один.

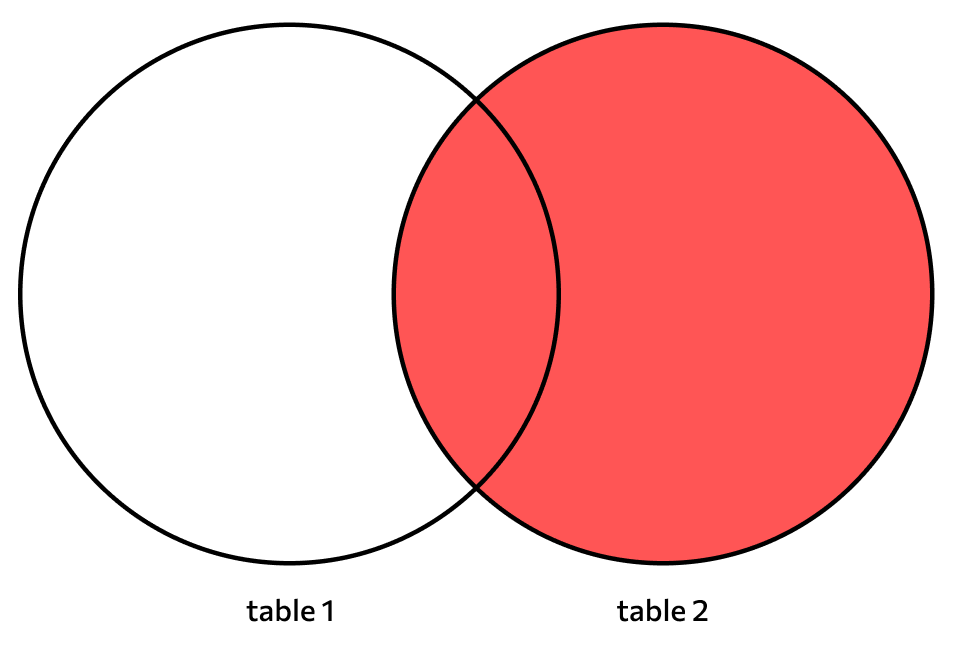
Inner Join



**Left Join**



**Right Join**



**Outer Full Join**

