**Metoda 1**

Fie User si Role table, si deci Users\_Roles table pentru many to many.

Tabelul Users\_Roles mai are o coloana, numita Active

@Entity  
@Table(name = "users")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class User {  
 @Id  
 @Column(name = "id")  
 private int id;  
 @Column(name = "username")  
 private String username;  
 @Column(name = "password")  
 private String password;  
  
  
}

@Entity  
@Table(name = "role")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class Role {  
 @Id  
 @Column(name = "id")  
 private int id;  
 @Column(name = "role")  
 private String role;  
  
}

Id va fi compus din 2 entities

@Embeddable  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
@EqualsAndHashCode  
public class UserRoleId implements Serializable {  
 @JoinColumn(name = "user\_id")  
 @ManyToOne  
 private User user;  
 @JoinColumn(name = "role\_id")  
 @ManyToOne  
 private Role role;  
}

@Entity  
@Table(name = "users\_roles")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class UserRole {  
 @EmbeddedId  
 private UserRoleId userRoleId;  
  
 @Column(name = "active")  
 private String active;  
  
**}**

**V II**

Stergem UserRoleId class, si folosim direct @Id la mai multe entity

@Entity  
@Table(name = "users\_roles")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class UserRole {  
 @Id  
 @JoinColumn(name = "user\_id")  
 @ManyToOne  
 private User user;

@Id  
 @JoinColumn(name = "role\_id")  
 @ManyToOne  
 private Role role;  
  
 @Column(name = "active")  
 private String active;  
  
}

**V III**

**@MapsId("fieldName") si EmbeddedId si Embeddable**

Cream un @Embeddable ce sa contina cele 2 entity ca id

@Embeddable  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
@EqualsAndHashCode  
public class UserRoleId implements Serializable {  
 @JoinColumn(name = "user\_id")  
 @ManyToOne  
 private User user;  
 @JoinColumn(name = "role\_id")  
 @ManyToOne  
 private Role role;  
}

Si cream entity pentru join table, si punem @EmbeddedId si atentie, @MapsId() contine in el numele la field ce corespunde obiectului dat in clasa @Embeddable, nu numele la coloana!

@Entity  
@Table(name = "users\_roles")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class UserRole {  
 @EmbeddedId  
 private UserRoleId userRoleId;  
 @MapsId("userId")  
 @ManyToOne  
 private User user;  
 @MapsId("roleId")  
 @ManyToOne  
 private Role role;  
  
 @Column(name = "active")  
 private String active;  
  
}

@Entity  
@Table(name = "users")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class User {  
 @Id  
 @Column(name = "id")  
 private int id;  
 @Column(name = "username")  
 private String username;  
 @Column(name = "password")  
 private String password;  
  
}

@Entity  
@Table(name = "role")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class Role {  
 @Id  
 @Column(name = "id")  
 private int id;  
 @Column(name = "role")  
 private String role;  
  
  
}

***Cea mai buna metoda***

@Entity  
@Table(name = "users\_roles")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class UserRole {  
 @Id  
 @ManyToOne  
 @JoinColumn(name = "user\_id")  
 private User user;  
 @Id  
 @ManyToOne  
 @JoinColumn(name = "role\_id")  
 private Role role;  
  
 @Column(name = "active")  
 private String active;  
  
}

@Entity  
@Table(name = "users")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class User {  
 @Id  
 @Column(name = "id")  
 private int id;  
 @Column(name = "username")  
 private String username;  
 @Column(name = "password")  
 private String password;  
  
 @OneToMany  
 @JoinTable(name = "users\_roles",  
 joinColumns = @JoinColumn(name = "user\_id"),  
 inverseJoinColumns = @JoinColumn(name = "role\_id"))  
 private List<Role> roles;  
}

@Entity  
@Table(name = "role")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class Role {  
 @Id  
 @Column(name = "id")  
 private int id;  
 @Column(name = "role")  
 private String role;  
  
}

**Simple Join Table**

@Entity  
@Table(name = "users\_roles")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class UserRole {  
 @Id  
 @Column(name = "user\_id")  
 private int userId;  
 @Id  
 @Column(name = "role\_id")  
 private int role\_id;  
  
 @Column(name = "active")  
 private String active;  
  
}

**Am folosit direct int aici, nu obiecte**

**si aici**

@Entity  
@Table(name = "users")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class User {  
 @Id  
 @Column(name = "id")  
 private int id;  
 @Column(name = "username")  
 private String username;  
 @Column(name = "password")  
 private String password;  
  
 @OneToMany  
 @JoinColumn(name = "user\_id")  
 private List<UserRole> userRoleList;  
}

@Entity  
@Table(name = "role")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class Role {  
 @Id  
 @Column(name = "id")  
 private int id;  
 @Column(name = "role")  
 private String role;  
  
 @OneToMany  
 @JoinColumn(name = "role\_id")  
 private List<UserRole> userRoleList;  
  
}

**Nu are cum exista ciclu!**

**Bidirection infinite loop**

Daca vom incerca sa facem o relatie bidirectionala, gen asa

@Entity  
@Table(name = "users")  
  
@NoArgsConstructor  
@Setter  
@Getter  
@ToString  
@AllArgsConstructor  
@Builder  
public class User {  
 @Id  
 @Column(name = "id")  
 private int id;  
 @Column(name = "username")  
 private String username;  
 @Column(name = "password")  
 private String password;  
  
 @OneToMany  
 @JoinColumn(name = "user\_id")  
 private List<UserRole> userRoleList;  
}

Cand vom extrage datele, vom avea bucla infinita, caci avem userRoleList si in el avem User si Role si in ele iar si iar userRoleList si asa la infinit

Trebuie sa rupem legatura bidirectionala, altceva nu prea avem ce face. Entity Role de ex nu are nevoie sa aiba legatura cu User

Dar, un asa approche nici nu ne ofera prea multe. Mai bine folosim approach cu @JoinTable

* Recursia apare chiar si cand avem tabele legate biderectional cu @OneToMnay, fara join table intre ele.

**@JsonIgnoreProperties({})**

Permite sa pecificam fieldurile care vrem sa nu fie mapate sau luate in considerare in obiectul deasupra la care se pune anotatia

@Entity  
@Table(name = "users")  
  
@Getter  
@Setter  
@ToString  
public class User {  
 @Id  
 private int id;  
 private String username;  
 private String password;  
 @OneToMany(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "user\_id")  
 @JsonIgnoreProperties({"user","userRoleId"})  
 private List<UserRole> users;  
}

Deci, din obiectul UserRole se vor evita fieldurile user, userRoleId

@Entity  
@Table  
  
@Getter  
@Setter  
@ToString  
public class Role {  
 @Id  
 private int id;  
 private String role;  
  
}

@Entity  
@Table(name = "users\_roles")  
  
@Getter  
@Setter  
@ToString  
public class UserRole {  
 @EmbeddedId  
 private UserRoleId userRoleId;  
 @MapsId("userId")  
 @ManyToOne(fetch = FetchType.*LAZY*)  
 private User user;  
 @MapsId("roleId")  
 @ManyToOne(fetch = FetchType.*LAZY*)  
 private Role role;  
  
 private String active;  
}