Skillbox

Java Persistence API

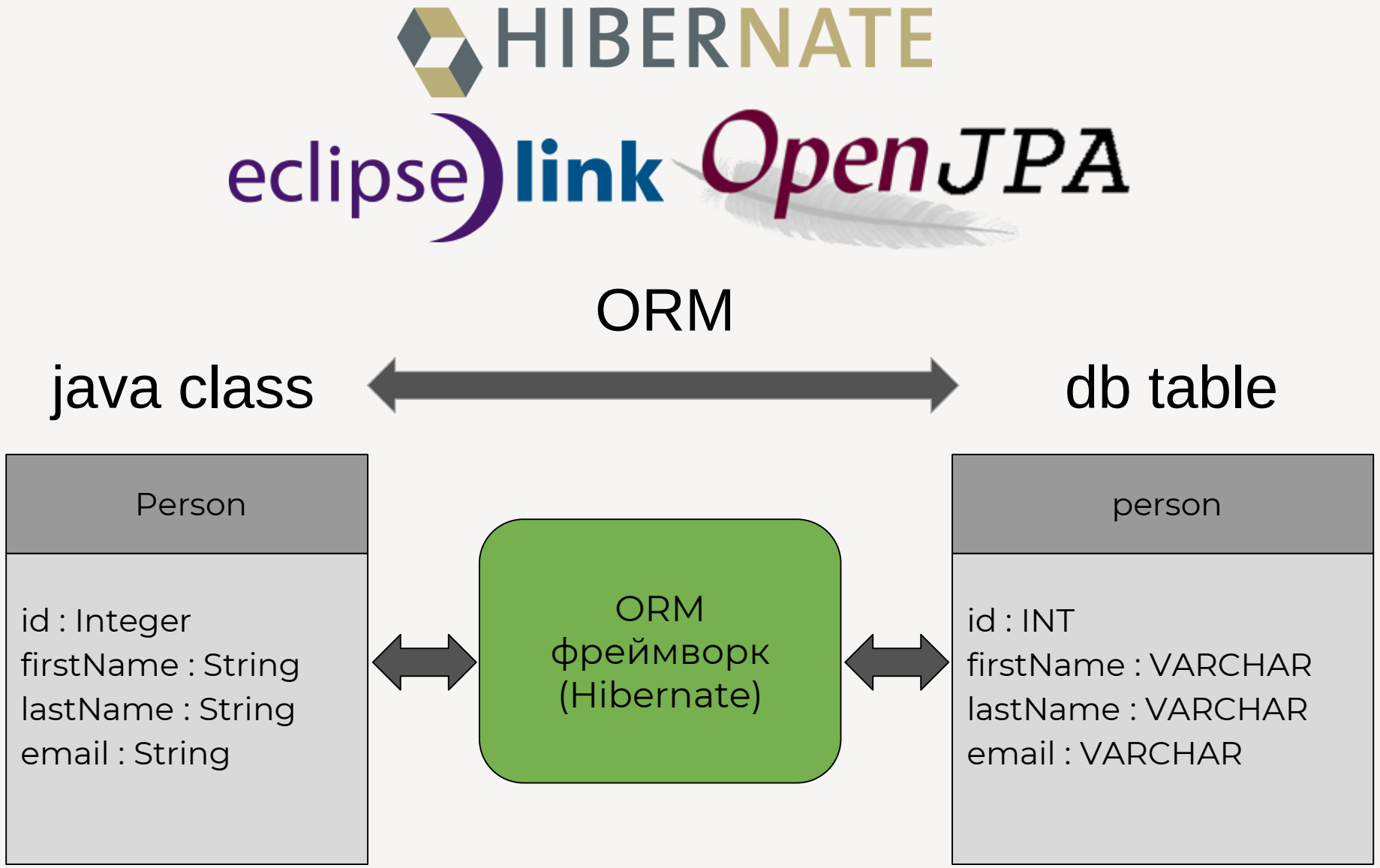
Кошаев К. А.

Java-разработчик

Skillbox

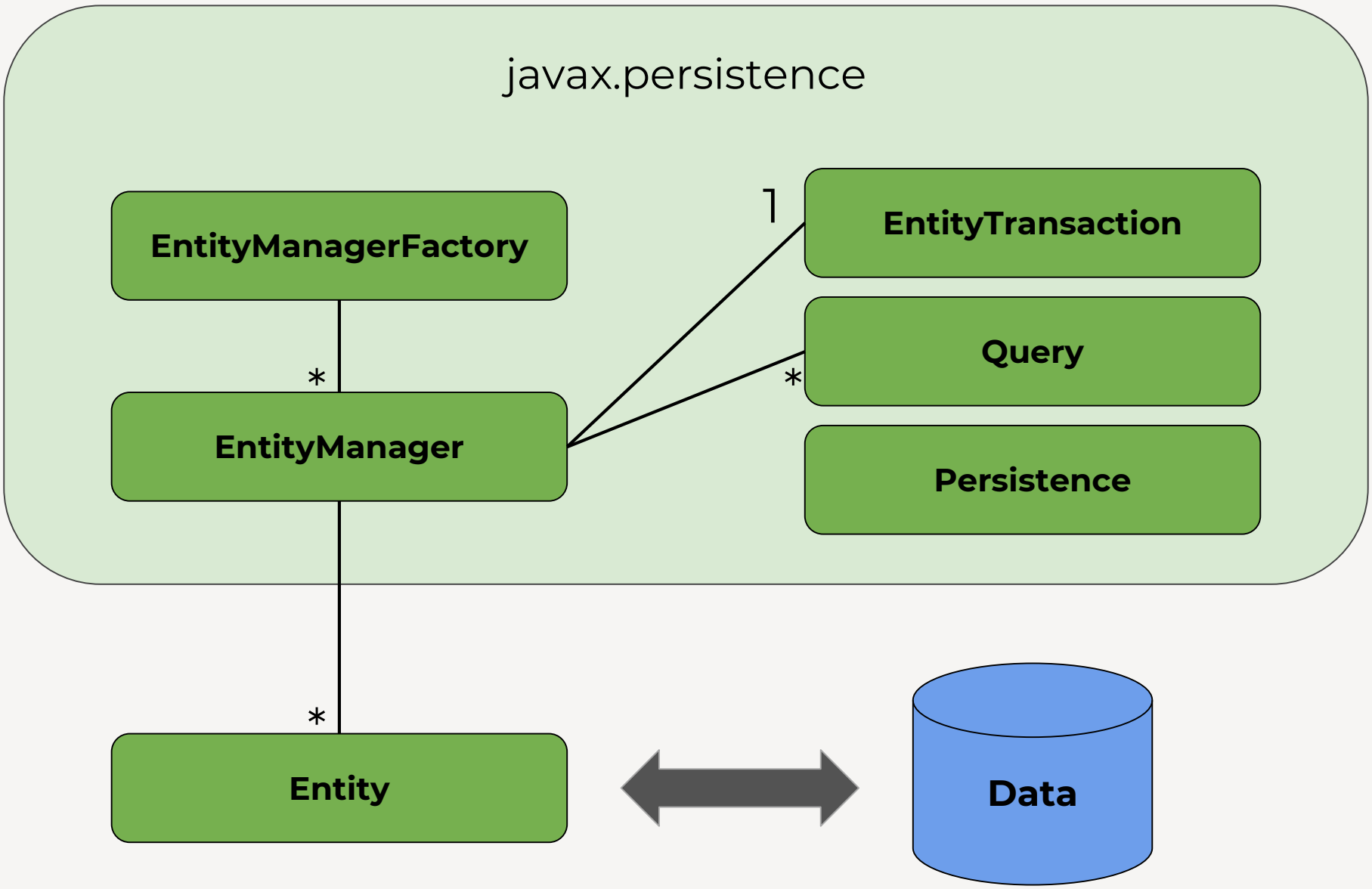
Java Persistence API

Java Persistence API (JPA) — это стандарт Java для отображения объектов Java в реляционную базу данных. Процесс маппинга объектов Java в таблицы базы данных и наоборот называется объектно-реляционным отображением (ORM)



Архтектура JPA

Представляет собой набор классов и интерфейсов, объединенных в пакете javax.persistence



JPA Entity

В этой схеме java объекты, которые являются persistence-объектами, определяются при помощи так называемых entity классов, т.е. сущностей. Сущность представляет собой отражение таблицы в реляционной базе данных

```
@Entity
public class User {
    // Persistent Fields:
    private String name;
    private String email;

    // Constructor:
    User (String name, String email) {
        this.name = name;
        this.email = email;
    }

    // getter/setter:
    public int getName() { return this.name; }
    public int getEmail() { return this.email; }

}
```

Аннотация @Entity указывает на то, что экземпляры этого класса являются persistence-объектами

Persistence - поля entity-класса User

Конструктор с аргументами

Методы доступа к persistence-полям

JPA Entity

- Класс должен быть помечен аннотацией @Entity
- Класс должен иметь public или protected no-args конструктор
- Класс нельзя объявлять как final
- Persistence-поля не должны быть объявлены как final
- Если экземпляр сущности передается по значению как отдельный объект, то класс такой сущности должен имплементировать интерфейс Serializable
- Entity классы могут наследовать и быть унаследованными другими сущностями и обычными классами
- Persistence-поля entity-класса должны иметь private или package-private модификатор доступа
- Доступ к persistence-полям должен осуществляться только с помощью getter и setter методов

JPA Entity

```
@Entity
@Table(name = "users")
public class User {
    // Persistent Fields:
        @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

        @Transient
    private String name;

    @Column(name = "mail")
    private String email;

    // Constructor:
    User (String name, String email) {
        this.name = name;
        this.email = email;
    }
        ...
}
```

Аннотация **@Table** указывает, что класс User отображает таблицу users

Аннотация **@Id** указывает, что поле **id** хранит первичный ключ таблицы user

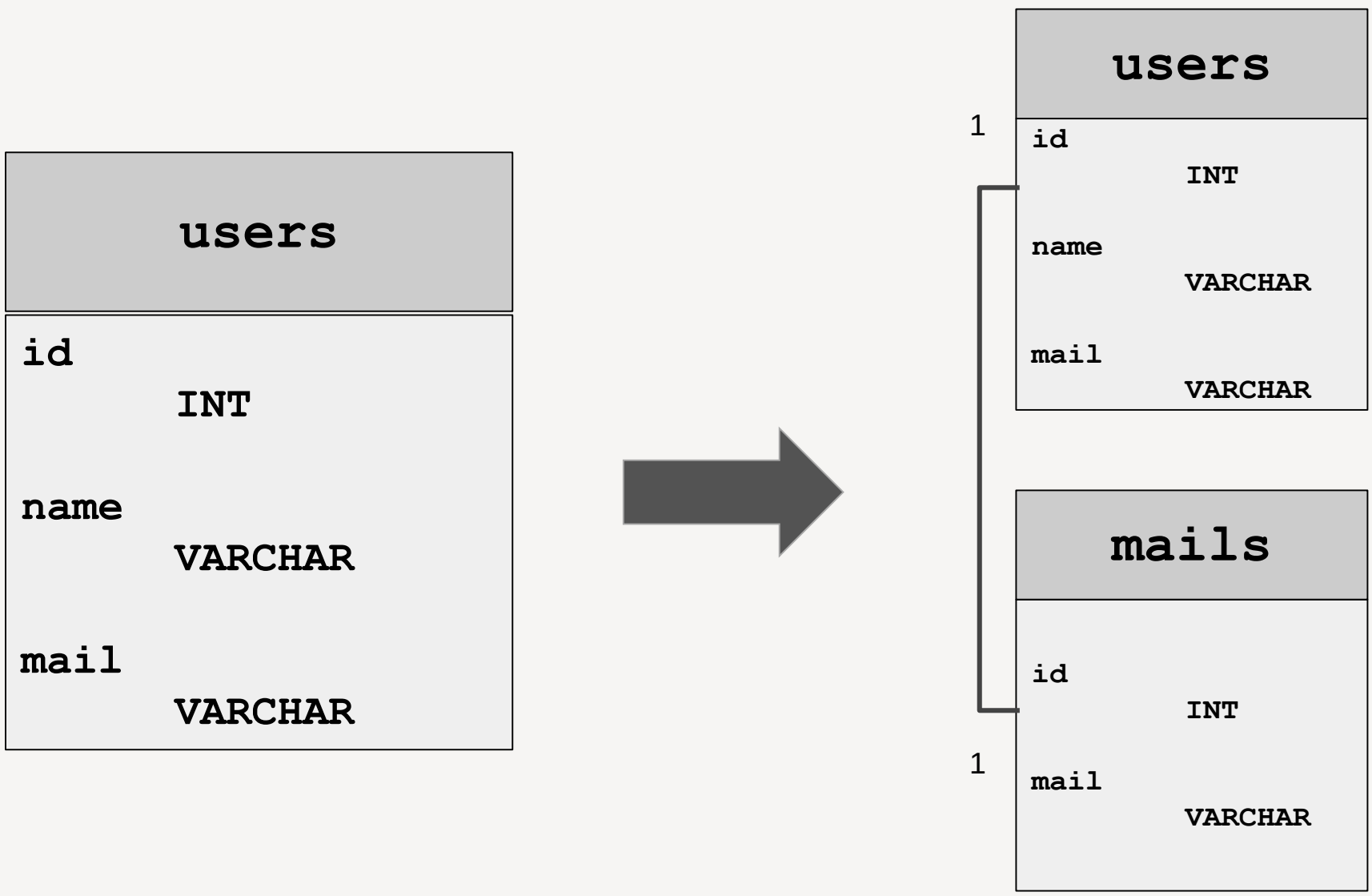
Значение первичного ключа будет генерироваться автоматически.

Поле **name** не участвует в persistence процессе из-за аннотации @Transient

@Column указывает, что поле **email** хранит данные столбца **mail** в таблице user

@OneToOne

Связь один к одному появляется, когда ключевой столбец (идентификатор) присутствует в другой таблице, в которой тоже является ключом, либо свойствами столбца задана его уникальность



@OneToOne

```
@Entity
@Table(name = "users")
public class User {
    // Persistent Fields:
        @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "mail_id", referencedColumnName = "id")
    private Email email;

    //getters setters constructors...
}

@Entity
@Table(name = "mails")
public class Email {
    // Persistent Fields:
        @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

        private String mail;

    @OneToOne(mappedBy = "email")
    private User user;

    //getters setters constructors...
}
```

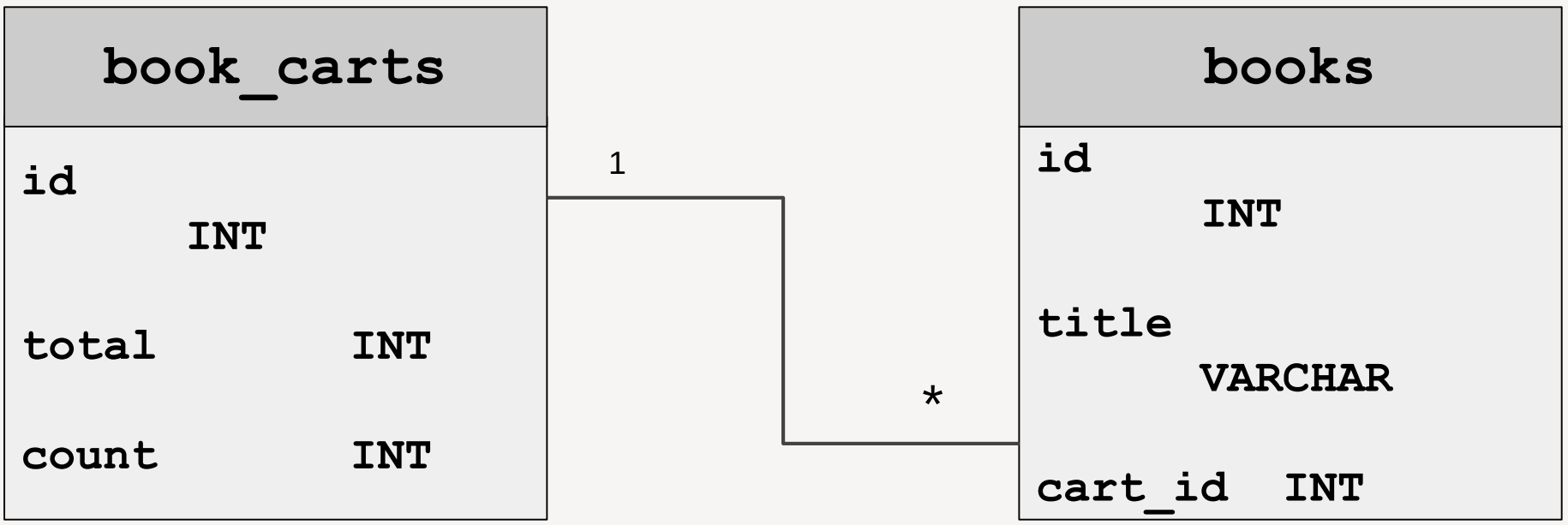
Связь “один к одному” с сущностью Email

Связь “один к одному” с сущностью Email

Skillbox

@OneToMany
@ManyToOne

Тип связи один ко многим — наиболее часто встречаемый на практике тип отношений между таблицами. “Один ко многим” предполагает, что несколько строк из дочерней таблицы зависят от одной строки в родительской таблице



Skillbox

@OneToMany
@ManyToOne

```
@Entity
@Table(name = "book_carts")
public class BookCart {
        @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private long total;

    private int count;

        @OneToMany(mappedBy = "bookCart")
    private Set<Book> bookSet;

    //getters setters constructors...
}
```

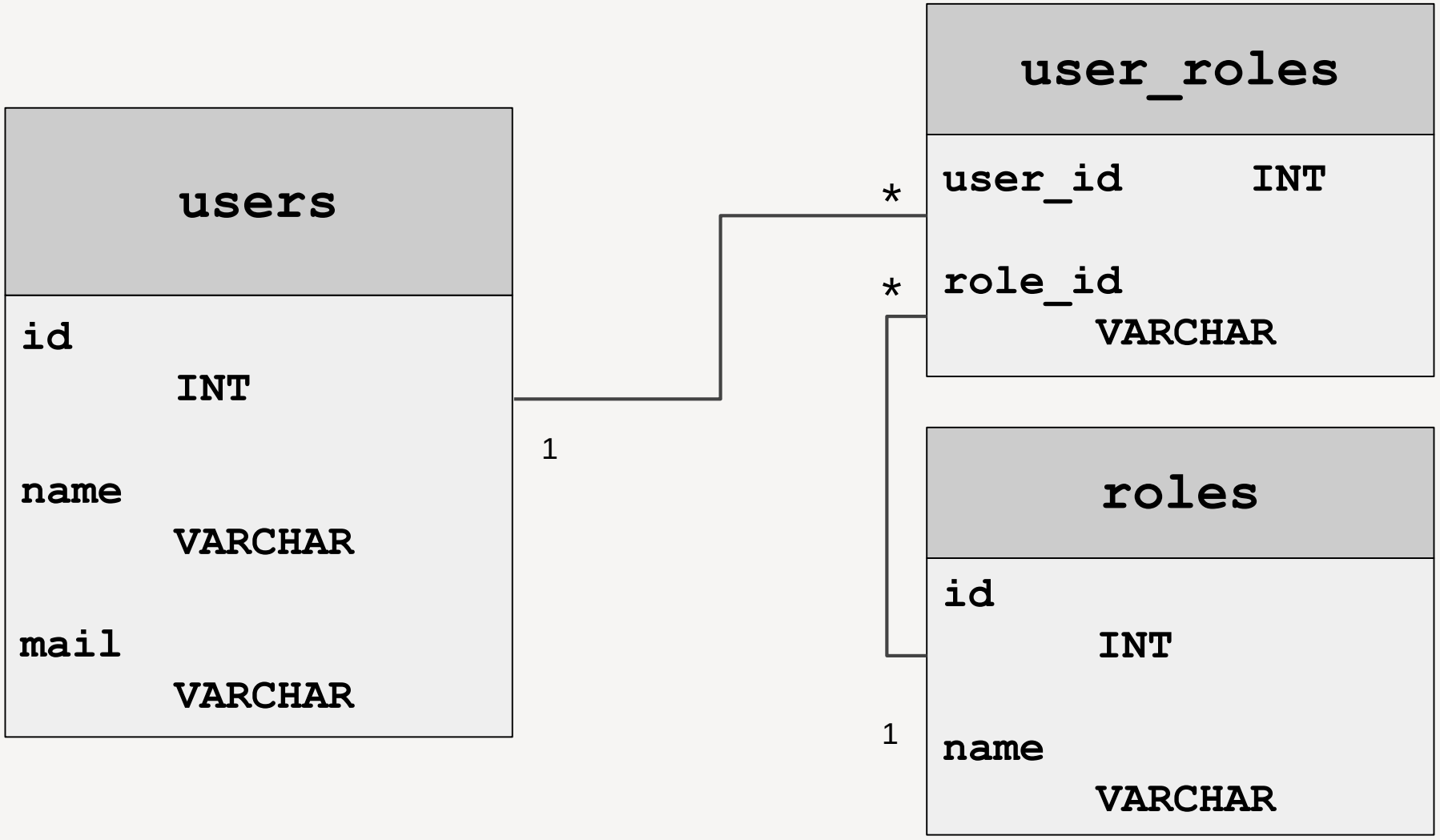
Связь “один ко многим” с сущностью Book

```
@Entity
@Table(name = "books")
public class Book {
        @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

        private String title;
        private int pagesCount;

    @ManyToOne
    @JoinColumn(name = "cart_id", nullable = false)
        private BookCart bookCart;

    //getters setters constructors...
}
```

Связь “многие к одному” с сущностью User

@ManyToMany

Определяет тип связей, при котором одна строка из таблицы X может быть связана с множеством строк из таблицы Y. В свою очередь, одна строка из таблицы Y может быть связана с множеством строк из таблицы X



@ManyToMany

```
@Entity
@Table(name = "users")
public class User {
        @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    private String mail;

        @ManyToMany
            @JoinTable(name = "user_role",
                    joinColumns = @JoinColumn(name = "user_id"),
                    inverseJoinColumns = @JoinColumn(name = "role_id")
                            )
    private Set<Role> roleSet;

    //getters setters constructors...
}
@Entity
@Table(name = "roles")
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

        private String name;

    @ManyToMany(mappedBy = "roleSet")
    private Set<User> userSet;

    //getters setters constructors...
}
```

Связь “многие ко многим” с сущностью Role

Атрибуты для сущности-владельца отношения “многие ко многим”

Связь “многие ко многим” с сущностью User с атрибутом mappedBy для указания на то, что сущность является дочерней

Cascade

Отношения одних сущностей часто зависят от существования других — например, как в случае отношения "корзина для книг — покупаемая книга". Без сущности, отражающей корзину для покупок, сущность покупаемая книга не будет иметь логического смысла, как самостоятельная единица

Типы cascade:

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

Skillbox

Спасибо за внимание!