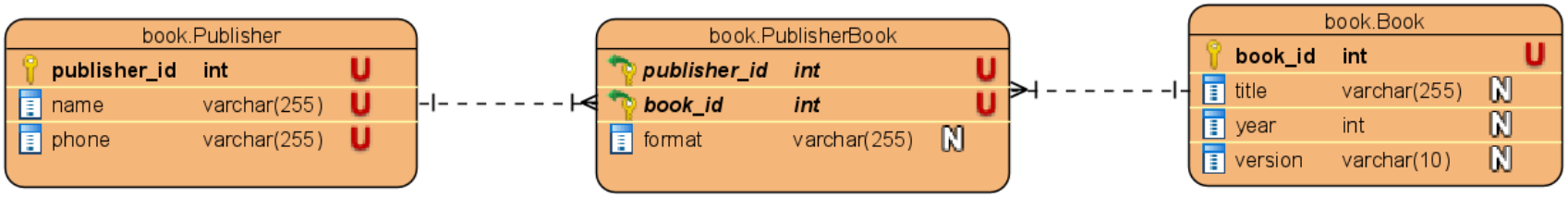
Hibernate Mapping

Dessign by: DieuNT1

Many-to-Many Mapping Annotation

@EmbeddedId and @MapsId

- Map a **many-to-many** association with extra columns.



- **Many-to-Many Using a Composite Key**
- Note, that there're some **key requirements, which a composite key class has to fulfill**:
  - ✓ We have to mark it with *@Embeddable*
  - ✓ It has to implement *java.io.Serializable*
  - ✓ We need to provide an implementation of the *hashcode()* and *equals()* methods
  - ✓ None of the fields can be an entity themselves.

Many-to-Many Mapping Annotation

@EmbeddedId and @MapsId

- Create a class that contains the id of Publisher_Book table.
- Implement the *Serializable* interface and the attributes *bookId* and *publisherId*.

```
@Embeddable
public class PublisherBookId implements Serializable {

    private static final long serialVersionUID = 1L;

    private int publisherId;
    private int bookId;

    // getter and setter method
    //
    @Override
    public int hashCode(){}

    @Override
    public boolean equals(Object obj) {}
}
```

Many-to-Many Mapping Annotation

@EmbeddedId and @MapsId

- You need to model the *Publisher_Book* table as an entity with 2 many-to-one relationships to the Publisher and Book entities.

```
@Entity
@Table(name = "Publisher_Book", schema = "book")
public class PublisherBook {

    @EmbeddedId
    private PublisherBookId id;

    @ManyToOne
    @MapsId(value = "publisherId")
    private Publisher publisher;

    @ManyToOne
    @MapsId(value = "bookId")
    private Book book;

    private String format;
    // getter and setter methods

}
```

Many-to-Many Mapping Annotation
@EmbeddedId and @MapsId

- And if you want to map them as bidirectional associations, you need to model the referencing side of the association on the *Book* and *Publisher* entity.

```
@Entity
@Table(name = "Publisher", schema = "book")
public class Publisher {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "publisher_id")
    private int publisherId;

    @Column(unique = true)
    private String name;
    @Column(unique = true)
    private String phone;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "publisher")
    private Set<PublisherBook> publisherBook;

    // getter and setter methods
}
```

Many-to-Many Mapping Annotation
@EmbeddedId and @MapsId

- And if you want to map them as bidirectional associations, you need to model the referencing side of the association on the *Book* and *Publisher* entity.

```
Entity
@Table(name = "Book", schema = "book")
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "book_id")
    private int bookId;

    @Column(unique = true)
    private String title;

    private int year;

    @Column(length = 10)
    private String version;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "book")
    private Set<PublisherBook> publisherBook;
    // getter and setter methods
}
```

Many-to-Many Mapping Annotation
@EmbeddedId and @MapsId

- Create a Test Script to check.
- Assuming that you have implemented the BookDao, PublisherDao as the DAO classes mentioned above.

```
class PublisherBookDaoTest {
    static PublisherBookDao publisherBookDao;
    static BookDao bookDao;
    static PublisherDao publisherDao;

    @BeforeAll
    static void setUpBeforeClass() throws Exception {
        publisherBookDao = new PublisherBookDaoImpl();
        bookDao = new BookDaoImpl();
        publisherDao = new PublisherDaoImpl();
    }
```

Many-to-Many Mapping Annotation

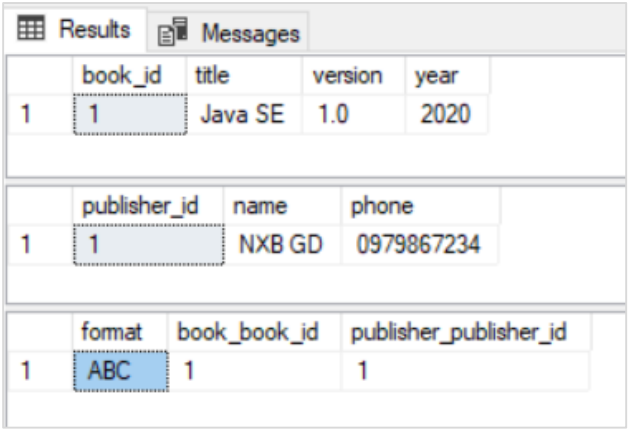
@EmbeddedId and @MapsId

```
    @Test
    void testSave() throws Exception {
        Book book = new Book(1, "Java SE", 2020, "1.0");
        assertTrue(bookDao.save(book));

        Publisher publisher = new Publisher(1, "NXB GD", "0979867234");
        assertTrue(publisherDao.save(publisher));

        PublisherBookId id = new PublisherBookId(1, 1);

        PublisherBook publisherBook = new PublisherBook();
        publisherBook.setId(id);
        publisherBook.setFormat("ABC");
        publisherBook.setBook(book);
        publisherBook.setPublisher(publisher);

        assertTrue(publisherBookDao.save(publisherBook));
    }
}
```

Many-to-Many Mapping Annotation

@EmbeddedId and @MapsId

- **Results**:



Results | Messages

	book_id	title	version	year
1	1	Java SE	1.0	2020

	publisher_id	name	phone
1	1	NXB GD	0979867234

	format	book_book_id	publisher_publisher_id
1	ABC	1	1

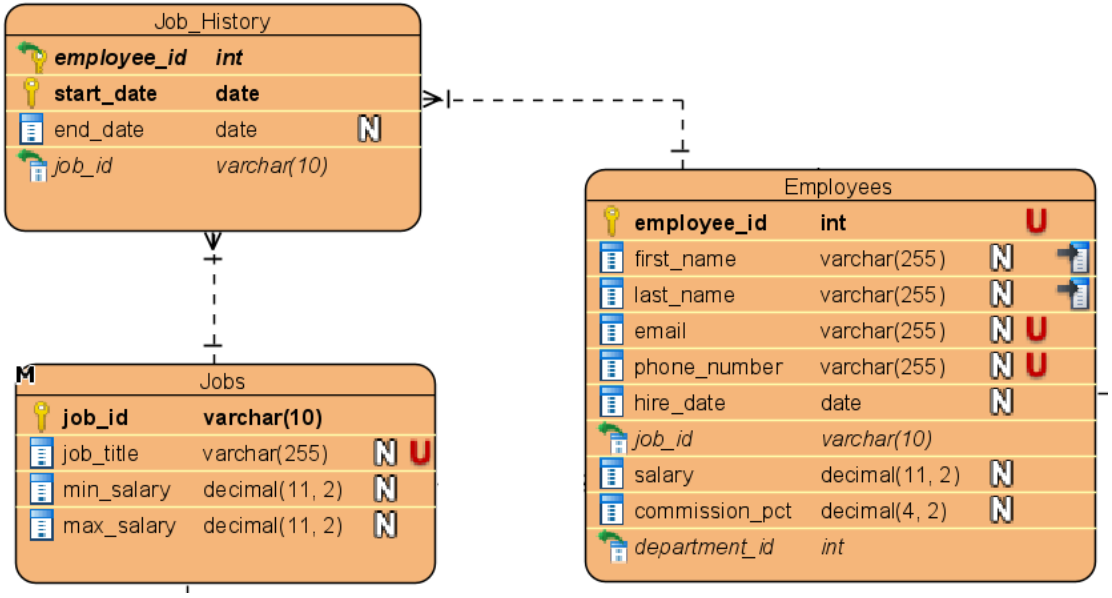
- **Console**:

```
Hibernate: create table book.Book (book_id int identity not null, title varchar(255), version varchar(10), year int no
Hibernate: create table book.Publisher (publisher_id int identity not null, name varchar(255), phone varchar(255), pri
Hibernate: create table book.Publisher_Book (format varchar(255), book_book_id int not null, publisher_publisher_id in
publisher_publisher_id))
Hibernate: alter table book.Book drop constraint UK_odppys65lq7q1xbx8o6p6fgxj
Hibernate: alter table book.Book add constraint UK_odppys65lq7q1xbx8o6p6fgxj unique (title)
Hibernate: alter table book.Publisher drop constraint UK_era79tsdasvick3e38j0e9b6v
Hibernate: alter table book.Publisher add constraint UK_era79tsdasvick3e38j0e9b6v unique (name)
Hibernate: alter table book.Publisher drop constraint UK_lfeio9fee753ckef2tac2vfku
Hibernate: alter table book.Publisher add constraint UK_lfeio9fee753ckef2tac2vfku unique (phone)
Hibernate: alter table book.Publisher_Book add constraint FKa0gwplfbh13yt8flgvc9yw5k6 foreign key (book_book_id) refer
Hibernate: alter table book.Publisher_Book add constraint FKkxs5ufy7sipi1me6hc2d7ksst foreign key (publisher_publisher
Oct 11, 2020 3:27:11 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatfo
Hibernate: insert into book.Book (title, version, year) values (?, ?, ?)
Hibernate: insert into book.Publisher (name, phone) values (?, ?)
Hibernate: insert into book.Publisher_Book (format, book_book_id, publisher_publisher_id) values (?, ?, ?)
```

Many-to-Many Mapping Annotation

@EmbeddedId and @AssociationOverride

- Map a **many-to-many** association with extra columns.



- **Many-to-Many Using a Composite Key**
- Note, that there're some **key requirements, which a composite key class has to fulfill**:
  - ✓ We have to mark it with *@Embeddable*
  - ✓ It has to implement *java.io.Serializable*
  - ✓ We need to provide an implementation of the *hashcode()* and *equals()* methods
  - ✓ None of the fields can be an entity themselves.

Many-to-Many Mapping Annotation

@EmbeddedId and @AssociationOverride

```java
@Embeddable
public class JobHistoryId implements Serializable {

    private static final long serialVersionUID = 1L;

    @ManyToOne(cascade = CascadeType.ALL)
    private Employees employee;

    @Column(name = "start_date")
    private LocalDate startDate;

    public Employees getEmployee() {
        return employee;
    }

    public void setEmployee(Employees employee) {
        this.employee = employee;
    }

    public LocalDate getStartDate() {
        return startDate;
    }

    public void setStartDate(LocalDate startDate) {
        this.startDate = startDate;
    }

    public int hashCode() {...

    public boolean equals(Object obj) {...

}
```

Many-to-Many Mapping Annotation

@EmbeddedId and @AssociationOverride

```
@Entity
@Table(name = "Job_History", schema = "dbo")
@AssociationOverride(name="id.employee", joinColumns=@JoinColumn(name="employee_id"))
public class JobHistory {

    @EmbeddedId
    private JobHistoryId id;

    // extra fields
    @Column(name = "end_date")
    private LocalDate endDate;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "job_id")
    private Jobs job;

    @Transient
    public Employees getEmployees() {
        return getId().getEmployee();
    }

    public void setEmployees(Employees employee) {
        getId().setEmployee(employee);
    }
    // getter and setter methods
}
```

Many-to-Many Mapping Annotation

@EmbeddedId and @AssociationOverride

- Update Jobs class and add the following declaration:

```
@OneToMany(cascade = CascadeType.ALL, mappedBy = "job")
private Set<JobHistory> histories;
```

- Update Employees class add the following declaration:

```
@OneToMany(mappedBy = "id.employee", cascade = CascadeType.ALL)
private Set<JobHistory> jobHistory;
```

Many-to-Many Mapping Annotation

@EmbeddedId and @AssociationOverride

- Create JobHistoryDaoImpl class:

```
public class JobHistoryDaoImpl implements JobHistoryDao {

    @Override
    public boolean save(JobHistory jobHistory) throws Exception {
        Session session = null;
        Transaction transaction = null;

        try {
            session = HibernateUtils.getSessionFactory().openSession();
            transaction = session.beginTransaction();
            Serializable result = session.save(jobHistory);
            transaction.commit();

            return (result != null);

        } finally {
            if (session != null) {
                session.close();
            }
        }
    }

}
```

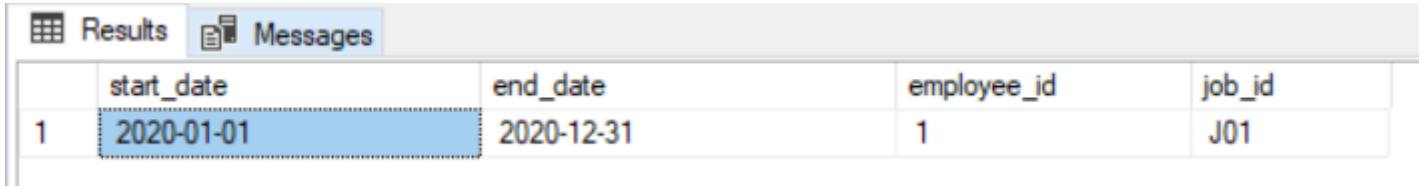
Many-to-Many Mapping Annotation
@EmbeddedId and @AssociationOverride

- Create a Test Script to test method of JobHistoryDaoImpl:

```
class JobHistoryDaoTest {
    static JobHistoryDao jhDao;

    @BeforeAll
    static void setUpBeforeClass() throws Exception {
        jhDao = new JobHistoryDaoImpl();
    }

    @Test
    void testSave() throws Exception {
        JobHistory jobHistory = new JobHistory();

        JobHistoryId id = new JobHistoryId();
        id.setEmployee(new Employees(1));
        id.setStartDate(LocalDate.of(2020, 1, 1));

        jobHistory.setId(id);
        jobHistory.setEndDate(LocalDate.of(2020, 12, 31));
        jobHistory.setJob(new Jobs("J01"));

        assertTrue(jhDao.save(jobHistory));
    }
}
```

Many-to-Many Mapping Annotation
@EmbeddedId and @AssociationOverride

- **Results**:


Results Messages

	start_date	end_date	employee_id	job_id
1	2020-01-01	2020-12-31	1	J01

- **Console**:

```
Hibernate: create table dbo.Job_History (start_date date not null, end_date date, employee_id int not null, job_id
varchar(10), primary key (employee_id, start_date))
Hibernate: alter table dbo.Job_History add constraint FKsrit9doy1c3hl0g01ju48x6tn foreign key (employee_id) references
dbo.Employees
Hibernate: alter table dbo.Job_History add constraint FK5fypfedbeoadd5lo3uo5yet26 foreign key (job_id) references dbo.Jobs
Oct 11, 2020 2:14:26 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:    select jobs_.job_id, jobs_.job_title as job_titl2_4_, jobs_.max_salary as max_sala3_4_,
              jobs_.min_salary as min_sala4_4_ from dbo.Jobs jobs_ where jobs_.job_id=?
Hibernate:    insert into dbo.Job_History (end_date, job_id, employee_id, start_date) values (?, ?, ?, ?)
```

Thank you!

Q&A