

## MODULE 15

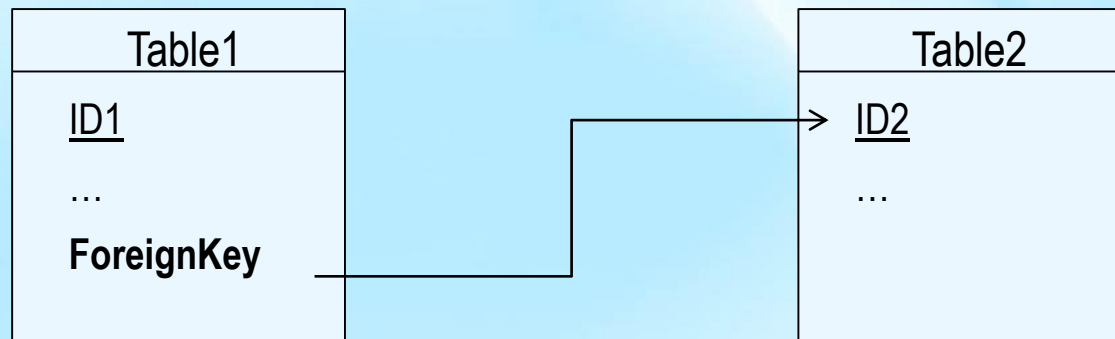
# DATABASE RELATIONSHIPS

# TABLE OF CONTENT

- One to Many Relation in Database
- One to Many Relation in Hibernate
- Fetch Type : Lazy >< Eager

# One to Many Relation in Database

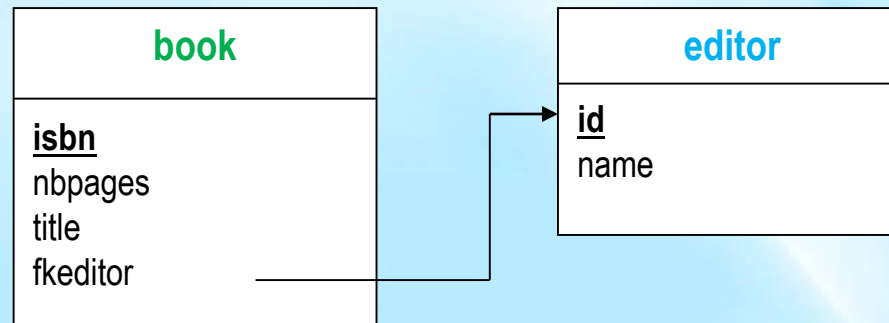
- ▶ In relational database, one to many relations are implemented through foreign keys



- Where
  - ID1 and ID2 are Identifiers
  - **ForeignKey** is the foreign key column referencing the Identifier of the related table
- Means that
  - Each row of the Table1 is linked to 0 or 1 row of the Table2
  - Each row of the Table2 is linked to 0, 1 or many rows of the Table1

# One to Many Relation in Database

► E.g,



- **MANY** **book** objects  $\Rightarrow$  **TO**  $\Rightarrow$  **ONE** **editor** object
  - i.e.,
    - A book has at most one editor
    - From a book object, a **single** *editor* object can be retrieved
- **ONE** **editor** object  $\Rightarrow$  **TO**  $\Rightarrow$  **MANY** **book** objects
  - i.e.,
    - An editor can edit several books
    - From an *editor* object, a **collection** of *book* objects can be retrieved

# One to Many Relation in Database

- Create **editor** table

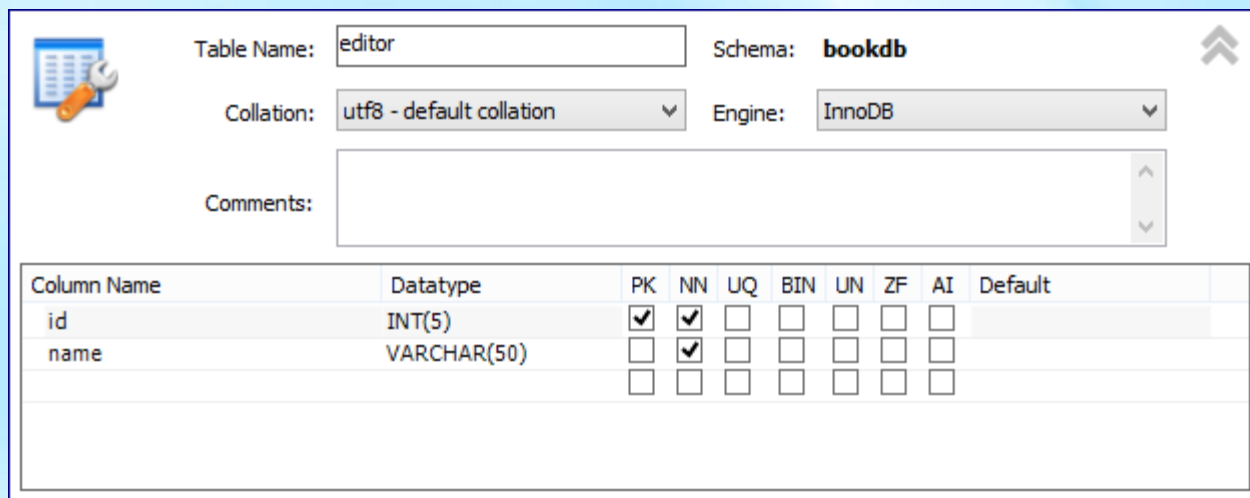


Table Name:  Schema: **bookdb**

Collation:  Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(5)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
name	VARCHAR(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	



```
CREATE TABLE `editor` (  
  `id` int(5) NOT NULL,  
  `name` varchar(50) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

# One to Many Relation in Database

- Add foreign key to **book** table

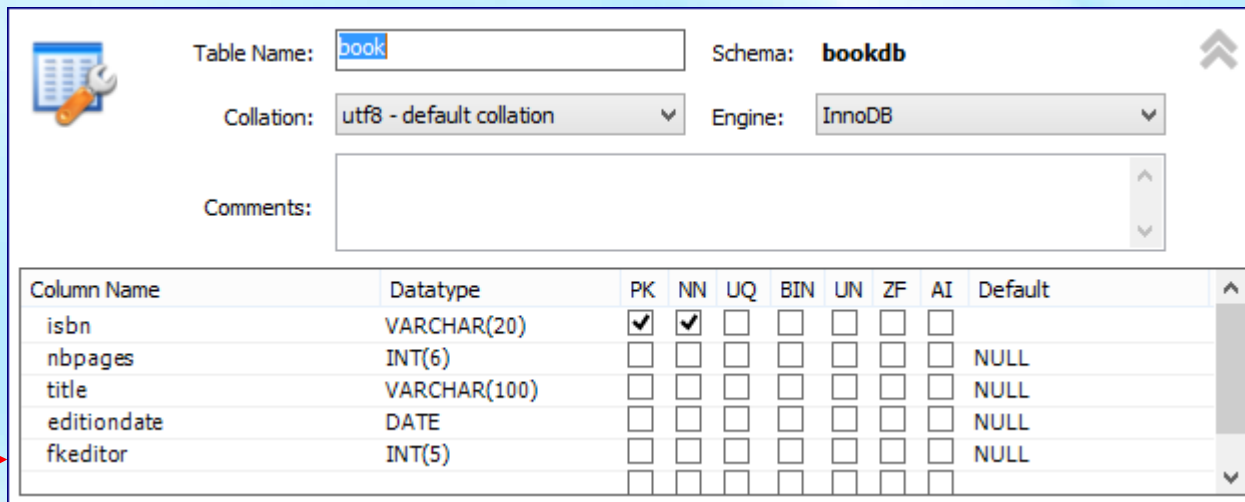


Table Name:  Schema: **bookdb**

Collation:  Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
isbn	VARCHAR(20)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
nbpages	INT(6)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
title	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
editiondate	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
fkeditor	INT(5)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

```
alter table book add  
foreign key (fkeditor)  
references editor(id);
```

# One to Many Relation in Hibernate

- ▶ To implement a one to many relation in entity beans
  - Use annotations **in both entity classes**
- ▶ For the foreign key property (e.g, in *BookEntity*)
  - **@JoinColumn**
    - **name** : name of the foreign key in the table
    - **referencedColumnName** : name of the Identifier column in the related table
  - **@ManyToOne**
- ▶ In the related table entity class : a **collection of objects** from the other table is created (e.g, *Collection <BookEntity>* in *EditorEntity*)
  - **@OneToMany**
    - **mappedBy** : name of the foreign key **property** in the other **entity class**

# One to Many Relation in Hibernate

## ► BookEntity class

```
@Entity
@Table(name="book")
public class BookEntity {

    @Id
    @Column(name="isbn")
    private String isbn;

    @Column(name="title")
    private String title;

    @Column(name="nbpages")
    private Integer nbPages;

    @Column(name="editiondate")
    private java.util.Date editionDate;

    @JoinColumn(name="fkeditor", referencedColumnName="id" )
    @ManyToOne
    private EditorEntity editor;
```

Name of the foreign key column in the table

Name of the ID column in the related table

Entity class of the related table



# One to Many Relation in Hibernate

## ► EditorEntity class

```
@Entity
@Table(name="editor")
public class EditorEntity {
    @Id
    @Column(name="id")
    private Integer id;

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

    @OneToOne(mappedBy="editor" )
    private Collection<BookEntity> books;
```

Name of the foreign key **property** in the **other entity class**

Collection of objects from the other entity class

# Fetch Type : Lazy >< Eager

## ► @OneToMany

- fetch = FetchType.**LAZY**
  - Loading of associated collections on-demand (as sub-queries)
  - Fetch when needed (when collections are first accessed)
- fetch = FetchType.**EAGER**
  - Loading of all associated collections at the creation of the objet
  - Fetch in one query (parent and child)
  - *But **resources consuming** ⇒ to avoid!*

# Fetch Type : Lazy >< Eager

► E.g,

```
import java.util.Collection;
import javax.persistence.*;

@Entity
@Table(name="editor")
public class EditorEntity {
    @Id
    @Column(name="id")
    private Integer id;

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

    @OneToMany(mappedBy="editor", fetch = FetchType.LAZY)
    private Collection<BookEntity> books;
```

# Named Query and Relationship

- ▶ Example of named query using **@ManyToOne** property

```
@Entity
@Table (name="book")
@NamedQueries ({
    @NamedQuery (
        name = "findBookByEditorName",
        query = "from BookEntity b where b.editor.name = :editorName"
    )
})
public class BookEntity {

    @Id
    @Column (name="isbn")
    private String isbn;

    @Column (name="title")
    private String title;

    @Column (name="nbpages")
    private Integer nbPages;

    @Column (name="editiondate")
    private java.util.Date editionDate;

    @JoinColumn (name="fkeditor", referencedColumnName="id" )
    @ManyToOne
    private EditorEntity editor;
```

