

 免費電子書

學習

jpa

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#jpa

.....	1
<b>1: jpa</b> .....	<b>2</b>
.....	2
.....	2
.....	2
.....	2
.....	2
<b>Examples</b> .....	<b>2</b>
.....	2
.....	2
EclipseLink .....	2
.....	2
DataNucleus .....	3
.....	3
persistence.xml .....	3
HibernateH2 DB .....	3
EclipselinkH2 DB .....	4
DataNucleusH2 DB .....	4
.....	5
.....	5
.....	5
.....	6
<b>DAO</b> .....	<b>7</b>
.....	8
<b>2:</b> .....	<b>10</b>
.....	10
<b>Examples</b> .....	<b>10</b>
.....	10
<b>3:</b> .....	<b>12</b>
.....	12
<b>Examples</b> .....	<b>12</b>
.....	12

<b>4:</b>	<b>14</b>
.....	14
Examples.....	14
.....	14
<b>5:</b>	<b>17</b>
.....	17
.....	17
Examples.....	17
.....	17
<b>6:</b>	<b>21</b>
.....	21
.....	21
Examples.....	21
.....	21
.....	21
.....	22
<b>Java 8</b>	<b>22</b>
<b>Java 8</b>	<b>23</b>
Id.....	23
<b>7:</b>	<b>24</b>
.....	24
Examples.....	24
ManyToOne.....	24
<b>8:</b>	<b>26</b>
.....	26
.....	26
.....	26
Examples.....	26
.....	26
Embeddable.....	29
<b>9:</b>	<b>32</b>
.....	



---

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [jpa](#)

It is an unofficial and free jpa ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jpa.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

# 1: jpa

JPA Java Persistence API Java。 ORM。 [JDBC](#)。 Java。

JPA。 。 JPA 2.1 [EclipseLink](#) JPA 2.1""; [HibernateDataNucleus](#) 。

Java。 JPA。 JPAJava。

- 
- 
- 
- 
- 

1.0	<a href="#">JSR-220</a>	2006-11-06
2.0	<a href="#">JSR-317</a>	2009-12-10
2.1	<a href="#">JSR-338</a>	2013522

## Examples

### EclipseLink

EclipselinkJPA API。 Maven

```
<dependencies>
  <dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>eclipselink</artifactId>
    <version>2.6.3</version>
  </dependency>
  <dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>javax.persistence</artifactId>
    <version>2.1.1</version>
  </dependency>
  <!-- ... -->
</dependencies>
```

Hibernate-core。 Maven

```
<dependencies>
  <dependency>
    <!-- requires Java8! -->
    <!-- as of 5.2, hibernate-entitymanager is merged into hibernate-core -->
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.1.Final</version>
  </dependency>
</dependencies>
```

```

</dependency>
<dependency>
  <groupId>org.hibernate.javax.persistence</groupId>
  <artifactId>hibernate-jpa-2.1-api</artifactId>
  <version>1.0.0</version>
</dependency>
<!-- ... -->
</dependencies>

```

## DataNucleus

datanucleus-core datanucleus-api-jpa datanucleus-rdbms RDBMS。 Maven

```

<dependencies>
  <dependency>
    <groupId>org.datanucleus</groupId>
    <artifactId>datanucleus-core</artifactId>
    <version>5.0.0-release</version>
  </dependency>
  <dependency>
    <groupId>org.datanucleus</groupId>
    <artifactId>datanucleus-api-jpa</artifactId>
    <version>5.0.0-release</version>
  </dependency>
  <dependency>
    <groupId>org.datanucleus</groupId>
    <artifactId>datanucleus-rdbms</artifactId>
    <version>5.0.0-release</version>
  </dependency>
  <dependency>
    <groupId>org.datanucleus</groupId>
    <artifactId>javax.persistence</artifactId>
    <version>2.1.2</version>
  </dependency>
  <!-- ... -->
</dependencies>

```

JPA *persistence.xml* CLASSPATH META-INF。 JPA。

JPA *orm.xml* META-INF。 JPA Java/。

**persistence.xml**

## Hibernate H2 DB

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">

  <persistence-unit name="persistenceUnit">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>

```

```

<class>my.application.entities.MyEntity</class>

<properties>
  <property name="javax.persistence.jdbc.driver" value="org.h2.Driver" />
  <property name="javax.persistence.jdbc.url" value="jdbc:h2:data/myDB.db" />
  <property name="javax.persistence.jdbc.user" value="sa" />

  <!-- DDL change options -->
  <property name="javax.persistence.schema-generation.database.action" value="drop-and-
create"/>

  <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
  <property name="hibernate.flushMode" value="FLUSH_AUTO" />
</properties>
</persistence-unit>
</persistence>

```

## EclipselinkH2 DB

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">

  <persistence-unit name="persistenceUnit">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>

    <class>my.application.entities.MyEntity</class>

    <properties>
      <property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>
      <property name="javax.persistence.jdbc.url" value="jdbc:h2:data/myDB.db"/>
      <property name="javax.persistence.jdbc.user" value="sa"/>

      <!-- Schema generation : drop and create tables -->
      <property name="javax.persistence.schema-generation.database.action" value="drop-and-
create-tables" />
    </properties>
  </persistence-unit>

</persistence>

```

## DataNucleusH2 DB

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">

  <persistence-unit name="persistenceUnit">
    <provider>org.datanucleus.api.jpa.PersistenceProviderImpl</provider>

```



```

<class>my.application.entities.MyEntity</class>

<properties>
  <property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>
  <property name="javax.persistence.jdbc.url" value="jdbc:h2:data/myDB.db"/>
  <property name="javax.persistence.jdbc.user" value="sa"/>

  <!-- Schema generation : drop and create tables -->
  <property name="javax.persistence.schema-generation.database.action" value="drop-and-
create-tables" />
</properties>
</persistence-unit>

</persistence>

```

## Hallo World

1. JPA 2.1
2. persistence-unit
- 3.
4. DAO
- 5.

---

## maven

```

<dependencies>

  <!-- JPA is a spec, I'll use the implementation with HIBERNATE -->
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>5.2.6.Final</version>
  </dependency>

  <!-- JDBC Driver, use in memory DB -->
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>1.4.193</version>
  </dependency>

</dependencies>

```

---

## resources

```

<persistence-unit name="hello-jpa-pu" transaction-type="RESOURCE_LOCAL">
  <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

  <properties>
    <!-- ~ = relative to current user home directory -->
    <property name="javax.persistence.jdbc.url" value="jdbc:h2:./test.db"/>
    <property name="javax.persistence.jdbc.user" value=""/>
    <property name="javax.persistence.jdbc.password" value=""/>
  </properties>
</persistence-unit>

```

```
<property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>
<property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
<property name="hibernate.show_sql" value="true"/>

<!-- This create automatically the DDL of the database's table -->
<property name="hibernate.hbm2ddl.auto" value="create-drop"/>

</properties>
</persistence-unit>
```

---

## Biker

```
package it.hello.jpa.entities;

import javax.persistence.*;
import java.io.Serializable;
import java.util.Date;
import java.util.List;

@Entity
@Table(name = "BIKER")
public class Biker implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

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

    @Column(unique = true, updatable = false)
    private String battleName;

    private Boolean beard;

    @Temporal(TemporalType.DATE)
    private Date birthday;

    @Temporal(TemporalType.TIME)
    private Date registrationDate;

    @Transient // --> this annotation make the field transient only for JPA
    private String criminalRecord;

    public Long getId() {
        return this.id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```

}

public String getBattleName() {
    return battleName;
}

public void setBattleName(String battleName) {
    this.battleName = battleName;
}

public Boolean getBeard() {
    return this.beard;
}

public void setBeard(Boolean beard) {
    this.beard = beard;
}

public Date getBirthday() {
    return birthday;
}

public void setBirthday(Date birthday) {
    this.birthday = birthday;
}

public Date getRegistrationDate() {
    return registrationDate;
}

public void setRegistrationDate(Date registrationDate) {
    this.registrationDate = registrationDate;
}

public String getCriminalRecord() {
    return criminalRecord;
}

public void setCriminalRecord(String criminalRecord) {
    this.criminalRecord = criminalRecord;
}
}

```

---

## DAO

```

package it.hello.jpa.business;

import it.hello.jpa.entities.Biker;

import javax.persistence.EntityManager;
import java.util.List;

public class MotorcycleRally {

    public Biker saveBiker(Biker biker) {
        EntityManager em = EntityManagerUtil.getEntityManager();
        em.getTransaction().begin();
        em.persist(biker);
    }
}

```

```
        em.getTransaction().commit();
        return biker;
    }
}
```

#### EntityManagerUtil

```
package it.hello.jpa.utils;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class EntityManagerUtil {

    // USE THE SAME NAME IN persistence.xml!
    public static final String PERSISTENCE_UNIT_NAME = "hello-jpa-pu";

    private static EntityManager entityManager;

    private EntityManagerUtil() {
    }

    public static EntityManager getEntityManager() {
        if (entityManager == null) {
            // the same in persistence.xml
            EntityManagerFactory emFactory =
                Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);

            return emFactory.createEntityManager();
        }
        return entityManager;
    }
}
```

---

#### it.hello.jpa.test;

#### TestJpa {

```
@Test
public void insertBiker() {
    MotorcycleRally crud = new MotorcycleRally();

    Biker biker = new Biker();
    biker.setName("Manuel");
    biker.setBeard(false);

    biker = crud.saveBiker(biker);

    Assert.assertEquals(biker.getId(), Long.valueOf(1L));
}
```

```
}
```

it.hello.jpa.test.TestJpa Hibernate drop table BIKER Hibernate drop sequence if exists hibernate\_sequence Hibernate create table BIKER id bigint not null battleName varchar(255) bikerName varchar(255) registrationDate id  
Hibernate alter table BIKER add constraint UK\_a64ce28nywyk8wqrvfkkuapli unique battleName  
Hibernate BIKER battleName bikerName registrationDate id  
mar 01, 2017 11:00:02 PM org.hibernate.jpa.internal.util.LogHelper logPersistenceUnitInformation  
INFO: HHH000204 PersistenceUnitInfo [name: hello-jpa-pu ...]

1000

[jpa https://riptutorial.com/zh-TW/jpa/topic/2125/jpa](https://riptutorial.com/zh-TW/jpa/topic/2125/jpa)

## 2:

@TableGenerator	
@GeneratedValue	
@OneToOne	Employee
mappedBy	o

## Examples

o

### Employee.java

```
@Entity
public class Employee {

    @TableGenerator(name = "employee_gen", table = "id_gen", pkColumnName = "gen_name",
valueColumnName = "gen_val", allocationSize = 100)
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "employee_gen")
    private int idemployee;
    private String firstname;
    private String lastname;
    private String email;

    @OneToOne
    @JoinColumn(name = "iddesk")
    private Desk desk;

    // getters and setters
}
```

### Desk.java

```
@Entity
public class Desk {

    @TableGenerator(table = "id_gen", name = "desk_gen", pkColumnName = "gen_name",
valueColumnName = "gen_value", allocationSize = 1)
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "desk_gen")
    private int iddesk;
    private int number;
    private String location;
    @OneToOne(mappedBy = "desk")
    private Employee employee;

    // getters and setters
}
```

```
}
```

```
/* Create EntityManagerFactory */
EntityManagerFactory emf = Persistence
    .createEntityManagerFactory("JPAExamples");

/* Create EntityManager */
EntityManager em = emf.createEntityManager();

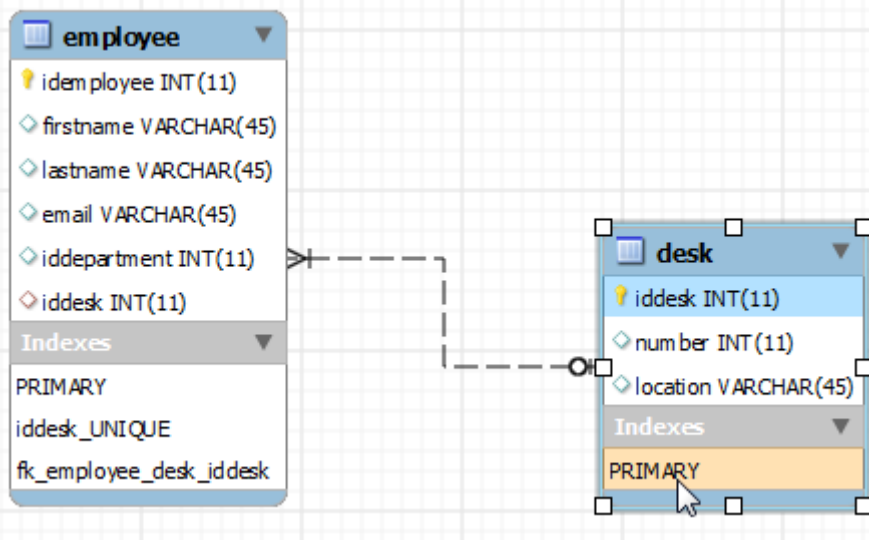
Employee employee;

employee = new Employee();
employee.setFirstname("pranil");
employee.setLastname("gilda");
employee.setEmail("sdfsdf");

Desk desk = em.find(Desk.class, 1); // retrieves desk from database
employee.setDesk(desk);

em.persist(employee);

desk = em.find(Desk.class, 1); // retrieves desk from database
desk.setEmployee(employee);
System.out.println(desk.getEmployee());
```



- **@JoinColumn**column. ◦ Employee**@JoinColumn**EmployeeDesk.
- **mappedBy@OneToOne** ◦ Desk.

<https://riptutorial.com/zh-TW/jpa/topic/6474/>

# 3:

@TableGenerator	
@GeneratedValue	
@ManyToOne	EmployeeDepartment
@OneToMany(mappedBy = "")	Employee@ManyToOneEmployeeDepartment

## Examples

◦ ◦

### Employee.java

```
@Entity
public class Employee {

    @TableGenerator(name = "employee_gen", table = "id_gen", pkColumnName = "gen_name",
valueColumnName = "gen_val", allocationSize = 100)
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "employee_gen")
    private int idemployee;
    private String firstname;
    private String lastname;
    private String email;

    @ManyToOne
    @JoinColumn(name = "iddepartment")
    private Department department;

    // getters and setters
}
```

### Department.java

```
@Entity
public class Department {

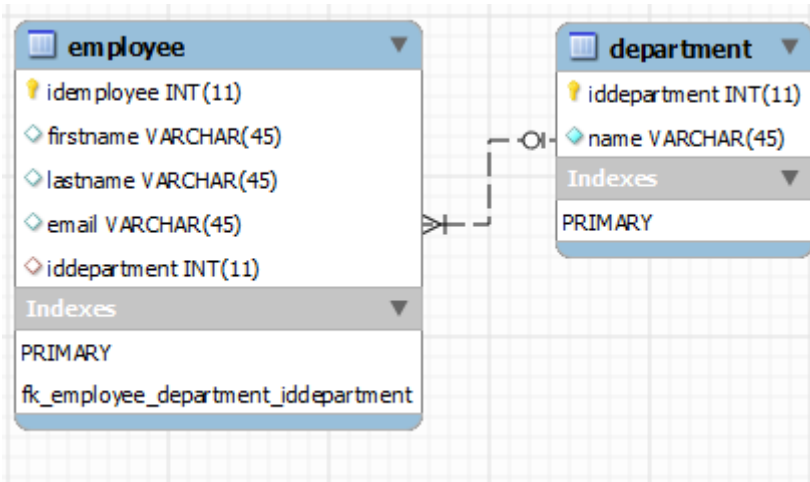
    @TableGenerator(table = "id_gen", pkColumnName = "gen_name", valueColumnName = "gen_val",
name = "department_gen", allocationSize = 1)
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "department_gen")
    private int iddepartment;
    private String name;

    @OneToMany(mappedBy = "department")
    private List<Employee> employees;

    // getters and setters
}
```



```
}
```



jpa

- ◦ ◦
- `mappedBy`

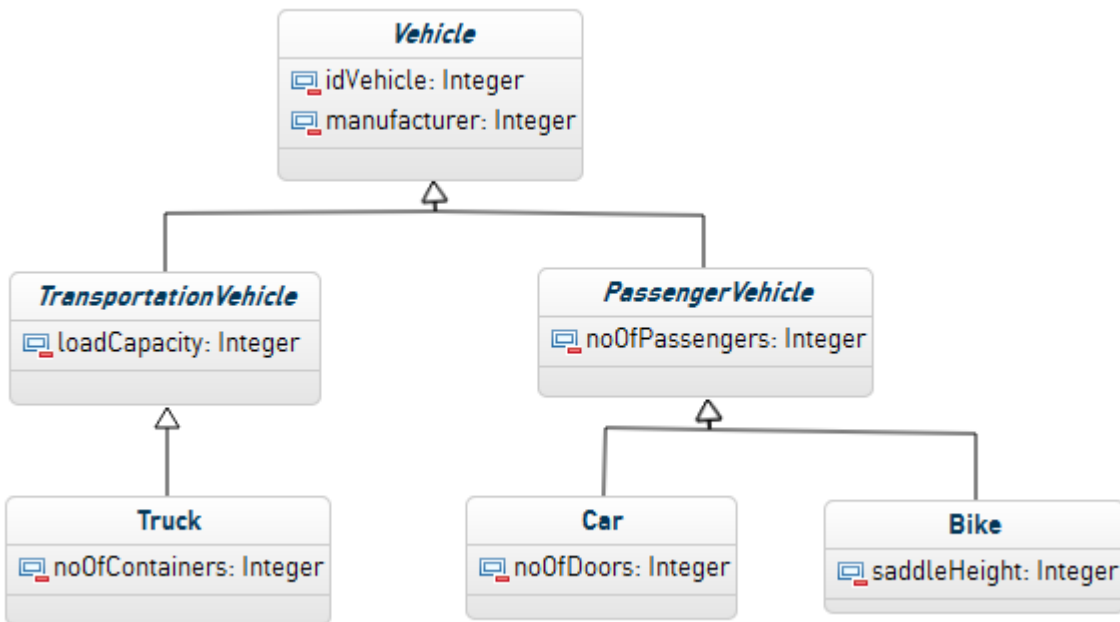
<https://riptutorial.com/zh-TW/jpa/topic/6529/>

# 4:

@	
@DiscriminatorColumn	ID
@MappedSuperclass	◦ java@MapperSuperClass

## Examples

SampleJPA◦



```
@Entity
@Table(name = "VEHICLE")
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn(name = "VEHICLE_TYPE")
public abstract class Vehicle {

    @TableGenerator(name = "VEHICLE_GEN", table = "ID_GEN", pkColumnName = "GEN_NAME",
valueColumnName = "GEN_VAL", allocationSize = 1)
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "VEHICLE_GEN")
    private int idVehicle;
    private String manufacturer;

    // getters and setters
}
```

### TransportationVehicle.java

```
@MappedSuperclass
public abstract class TransportationVehicle extends Vehicle {

    private int loadCapacity;

    // getters and setters
}
```

## Truck.java

```
@Entity
public class Truck extends TransportationVehicle {

    private int noOfContainers;

    // getters and setters
}
```

## PassengerVehicle.java

```
@MappedSuperclass
public abstract class PassengerVehicle extends Vehicle {

    private int noOfpassengers;

    // getters and setters
}
```

## Car.java

```
@Entity
public class Car extends PassengerVehicle {

    private int noOfDoors;

    // getters and setters
}
```

## Bike.java

```
@Entity
public class Bike extends PassengerVehicle {

    private int saddleHeight;

    // getters and setters
}
```

```
/* Create EntityManagerFactory */
EntityManagerFactory emf = Persistence
    .createEntityManagerFactory("AdvancedMapping");

/* Create EntityManager */
```

```

EntityManager em = emf.createEntityManager();
EntityTransaction transaction = em.getTransaction();

transaction.begin();

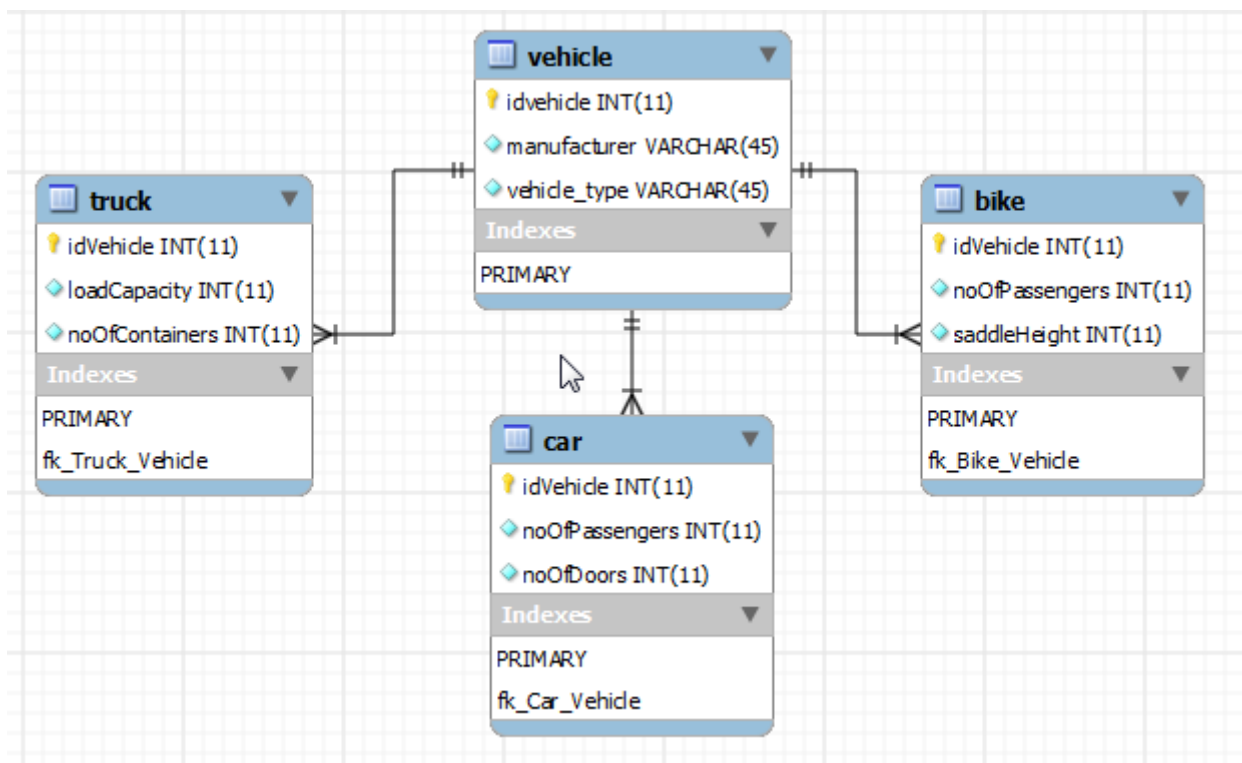
Bike cbr1000rr = new Bike();
cbr1000rr.setManufacturer("honda");
cbr1000rr.setNoOfpassengers(1);
cbr1000rr.setSaddleHeight(30);
em.persist(cbr1000rr);

Car aventador = new Car();
aventador.setManufacturer("lamborghini");
aventador.setNoOfDoors(2);
aventador.setNoOfpassengers(2);
em.persist(aventador);

Truck truck = new Truck();
truck.setLoadCapacity(1000);
truck.setManufacturer("volvo");
truck.setNoOfContainers(2);
em.persist(truck);

transaction.commit();

```



<https://riptutorial.com/zh-TW/jpa/topic/6473/>

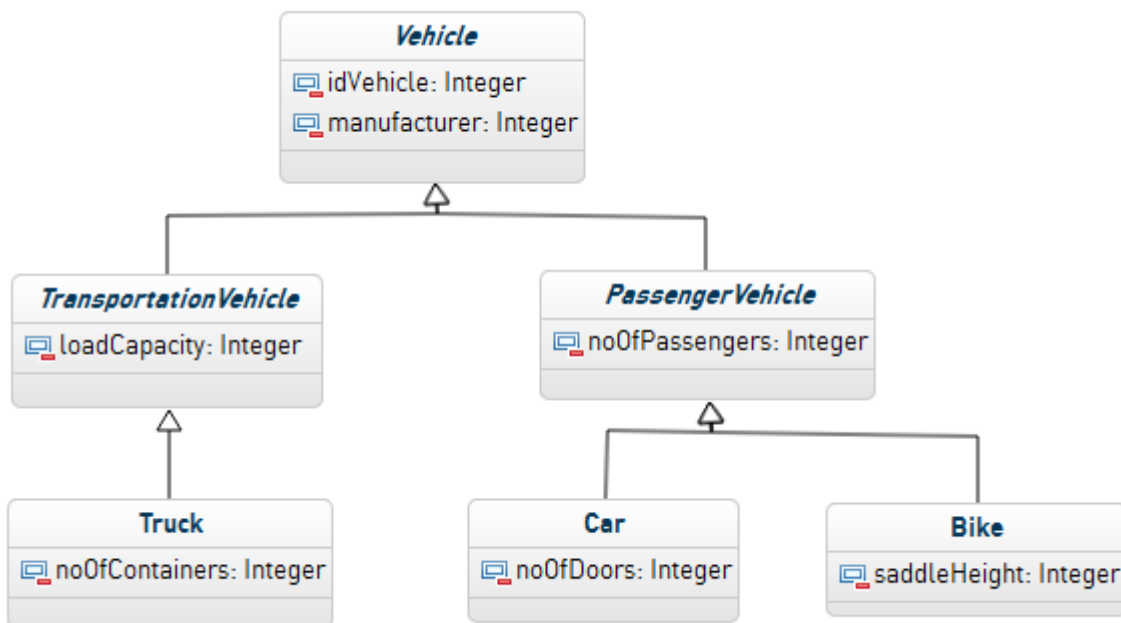
## 5:

@	
@DiscriminatorColumn	ID
@MappedSuperclass	◦ java@MapperSuperClass
@DiscriminatorValue	@DiscriminatorColumn.◦

◦

## Examples

Vehicle◦



```
package com.thejavageek.jpa.entities;

import javax.persistence.DiscriminatorColumn;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.Table;
import javax.persistence.TableGenerator;

@Entity
@Table(name = "VEHICLE")
```

```

@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "VEHICLE_TYPE")
public abstract class Vehicle {

    @TableGenerator(name = "VEHICLE_GEN", table = "ID_GEN", pkColumnName = "GEN_NAME",
valueColumnName = "GEN_VAL", allocationSize = 1)
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "VEHICLE_GEN")
    private int idVehicle;
    private String manufacturer;

    public int getIdVehicle() {
        return idVehicle;
    }

    public void setIdVehicle(int idVehicle) {
        this.idVehicle = idVehicle;
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

}

```

## TransportableVehicle.java

```

import javax.persistence.MappedSuperclass;

```

```

@MappedSuperclass
public abstract class TransportationVehicle extends Vehicle {

    private int loadCapacity;

    public int getLoadCapacity() {
        return loadCapacity;
    }

    public void setLoadCapacity(int loadCapacity) {
        this.loadCapacity = loadCapacity;
    }

}

```

## PassengerVehicle.java

```

package com.thejavageek.jpa.entities;

import javax.persistence.MappedSuperclass;

@MappedSuperclass
public abstract class PassengerVehicle extends Vehicle {

    private int noOfpassengers;
}

```

```

    public int getNoOfpassengers() {
        return noOfpassengers;
    }

    public void setNoOfpassengers(int noOfpassengers) {
        this.noOfpassengers = noOfpassengers;
    }
}

```

## Truck.java

```

package com.thejavageek.jpa.entities;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue(value = "Truck")
public class Truck extends TransportationVehicle{

    private int noOfContainers;

    public int getNoOfContainers() {
        return noOfContainers;
    }

    public void setNoOfContainers(int noOfContainers) {
        this.noOfContainers = noOfContainers;
    }
}

```

## Bike.java

```

package com.thejavageek.jpa.entities;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue(value = "Bike")
public class Bike extends PassengerVehicle {

    private int saddleHeight;

    public int getSaddleHeight() {
        return saddleHeight;
    }

    public void setSaddleHeight(int saddleHeight) {
        this.saddleHeight = saddleHeight;
    }
}

```

## Car.java

```

package com.thejavageek.jpa.entities;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue(value = "Car")
public class Car extends PassengerVehicle {

    private int noOfDoors;

    public int getNoOfDoors() {
        return noOfDoors;
    }

    public void setNoOfDoors(int noOfDoors) {
        this.noOfDoors = noOfDoors;
    }

}

```

```

/* Create EntityManagerFactory */
EntityManagerFactory emf = Persistence
    .createEntityManagerFactory("AdvancedMapping");

/* Create EntityManager */
EntityManager em = emf.createEntityManager();
EntityTransaction transaction = em.getTransaction();
transaction.begin();

Bike cbr1000rr = new Bike();
cbr1000rr.setManufacturer("honda");
cbr1000rr.setNoOfpassengers(1);
cbr1000rr.setSaddleHeight(30);
em.persist(cbr1000rr);

Car avantador = new Car();
avantador.setManufacturer("lamborghini");
avantador.setNoOfDoors(2);
avantador.setNoOfpassengers(2);
em.persist(avantador);

Truck truck = new Truck();
truck.setLoadCapacity(100);
truck.setManufacturer("mercedes");
truck.setNoOfContainers(2);
em.persist(truck);

transaction.commit();

```

<https://riptutorial.com/zh-TW/jpa/topic/6277/>



## 6:

@Id	/
@Basic	◦ String DateCalendar ◦ ◦
@Transient	transient◦

- Java;◦

## Examples

```
@Entity
class Note {
    @Id
    Integer id;

    @Basic
    String note;

    @Basic
    int count;
}
```

JPA◦

### JavaPostgres

```
CREATE TABLE Note (
  id integer NOT NULL,
  note text,
  count integer NOT NULL
)
```

JPADDDL◦ DDL◦

```
@Entity
class Note {
    @Id
    Integer id;

    @Basic
    String note;

    @Transient
    String parsedNote;

    String readParsedNote() {
        if (parsedNote == null) { /* initialize from note */ }
        return parsedNote;
    }
}
```

```
}  
}
```

@Transient ◦ null ◦

**Java** Date Calendar LocalDate LocalDateTime ◦ Timestamp Instant ZonedDateTime **Joda-time** ◦ time  
date timestamp ◦

## Java 8

**Java-8** java.util.Date java.util.Calendar java.sql.Timestamp **SQL** timestamp ; java.sql.Date date ◦

```
@Entity  
class Times {  
    @Id  
    private Integer id;  
  
    @Basic  
    private Timestamp timestamp;  
  
    @Basic  
    private java.sql.Date sqldate;  
  
    @Basic  
    private java.util.Date utildate;  
  
    @Basic  
    private Calendar calendar;  
}
```

```
CREATE TABLE times (  
    id integer not null,  
    timestamp timestamp,  
    sqldate date,  
    utildate timestamp,  
    calendar timestamp  
)
```

◦ **Java** Date Calendar ◦ @Temporal ◦

```
@Entity  
class Times {  
    @Id  
    private Integer id;  
  
    @Temporal(TemporalType.TIME)  
    private Date date;  
  
    @Temporal(TemporalType.DATE)  
    private Calendar calendar;  
}
```

## SQL

```
CREATE TABLE times (
  id integer not null,
  date time,
  calendar date
)
```

1 @Temporal DDL; date Date @Basic

2 Calendar time

## Java 8

JPA 2.1 Java 8 java.time JPA 2.1

DataNucleus @Temporal

Hibernate Hibernate 5.2+ @Basic Hibernate 5.0-5.1 org.hibernate:hibernate-java8

- date LocalDate
- Instant LocalDateTime ZoneDateTime timestamp

java.time Java 8 java.time JPA 2.1 AttributeConverter

## Id

userUid SEQUENCE SEQUENCE USER\_UID\_SEQ DBAJPA

```
@Entity
@Table(name="USER")
public class User implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @SequenceGenerator(name="USER_UID_GENERATOR", sequenceName="USER_UID_SEQ")
    @GeneratedValue(strategy=GenerationType.SEQUENCE, generator="USER_UID_GENERATOR")
    private Long userUid;

    @Basic
    private String userName;
}
```

<https://riptutorial.com/zh-TW/jpa/topic/3691/>

# 7:

@TableGenerator	ID
@GeneratedValue	
@ID	
@ManyToOne	EmployeeDepartment ◦ ◦ ◦ Employee@ManyToOne ◦
@JoinColumn	

## Examples

### ManyToOne

```
@Entity
public class Employee {

    @TableGenerator(name = "employee_gen", table = "id_gen", pkColumnName = "gen_name",
valueColumnName = "gen_val", allocationSize = 1)
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "employee_gen")
    private int idemployee;
    private String firstname;
    private String lastname;
    private String email;

    @ManyToOne
    @JoinColumn(name = "iddepartment")
    private Department department;

    // getters and setters
    // toString implementation
}
```

```
@Entity
public class Department {

    @Id
    private int iddepartment;
    private String name;

    // getters, setters and toString()
}
```

```
public class Test {

    public static void main(String[] args) {
```

```
EntityManagerFactory emf = Persistence
    .createEntityManagerFactory("JPAAExamples");
EntityManager em = emf.createEntityManager();
EntityTransaction txn = em.getTransaction();

Employee employee = new Employee();
employee.setEmail("someMail@gmail.com");
employee.setFirstname("Prasad");
employee.setLastname("kharkar");

txn.begin();
Department department = em.find(Department.class, 1);//returns the department named
vert
System.out.println(department);
txn.commit();

employee.setDepartment(department);

txn.begin();
em.persist(employee);
txn.commit();

}

}
```

<https://riptutorial.com/zh-TW/jpa/topic/6531/>

# 8:

ManyToMany@ManyToMany◦

@OneToMany ManyToMany◦

@TableGenerator	GeneratedValue
@GeneratedValue	◦ Id◦
@ManyToMany	EmployeeProject◦
mappedBy="projects"	EmployeeProject
@JoinColumn	
@JoinTable	

- @TableGenerator@GeneratedValuejpaId◦
- @ManyToManyEmployeeProject◦
- @JoinTablejpaname = "employee\_project"EmployeeProject◦ jpa◦
- @JoinColumn@ inverseJoinColumn◦ ◦ ◦ Employee@JoinColumnemployee\_project  
idemployee@InverseJoinColumnidprojectjpa◦
- Project@ManyToManymappedBy = projectsEmployee◦

## Examples

◦

```
package com.thejavageek.jpa.entities;

import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.TableGenerator;

@Entity
public class Employee {

    @TableGenerator(name = "employee_gen", table = "id_gen", pkColumnName = "gen_name",
valueColumnName = "gen_val", allocationSize = 100)
    @Id
```

```

    @GeneratedValue(strategy = GenerationType.TABLE, generator = "employee_gen")
    private int idemployee;
    private String name;

    @ManyToMany(cascade = CascadeType.PERSIST)
    @JoinTable(name = "employee_project", joinColumns = @JoinColumn(name = "idemployee"),
inverseJoinColumns = @JoinColumn(name = "idproject"))
    private List<Project> projects;

    public int getIdemployee() {
        return idemployee;
    }

    public void setIdemployee(int idemployee) {
        this.idemployee = idemployee;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<Project> getProjects() {
        return projects;
    }

    public void setProjects(List<Project> projects) {
        this.projects = projects;
    }
}

```

```

package com.thejavageek.jpa.entities;

import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.TableGenerator;

@Entity
public class Project {

    @TableGenerator(name = "project_gen", allocationSize = 1, pkColumnName = "gen_name",
valueColumnName = "gen_val", table = "id_gen")
    @Id
    @GeneratedValue(generator = "project_gen", strategy = GenerationType.TABLE)
    private int idproject;
    private String name;

    @ManyToMany(mappedBy = "projects", cascade = CascadeType.PERSIST)
    private List<Employee> employees;
}

```

```

public int getIdproject() {
    return idproject;
}

public void setIdproject(int idproject) {
    this.idproject = idproject;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public List<Employee> getEmployees() {
    return employees;
}

public void setEmployees(List<Employee> employees) {
    this.employees = employees;
}
}

```

**/\*EntityManagerFactory \*/ EntityManagerFactory emf = Persistence .createEntityManagerFactory  
“JPAExamples”;**

```

/* Create EntityManager */
EntityManager em = emf.createEntityManager();

EntityTransaction transaction = em.getTransaction();

transaction.begin();

Employee prasad = new Employee();
prasad.setName("prasad kharkar");

Employee harish = new Employee();
harish.setName("Harish taware");

Project ceg = new Project();
ceg.setName("CEG");

Project gtt = new Project();
gtt.setName("GTT");

List<Project> projects = new ArrayList<Project>();
projects.add(ceg);
projects.add(gtt);

List<Employee> employees = new ArrayList<Employee>();
employees.add(prasad);
employees.add(harish);

ceg.setEmployees(employees);
gtt.setEmployees(employees);

prasad.setProjects(projects);

```



```

harish.setProjects(projects);

em.persist(prasad);

transaction.commit();

```

## Embeddable

```

Role:
+-----+
| roleId | name | discription |
+-----+

Rights:
+-----+
| rightId | name | discription|
+-----+

rightrole
+-----+
| roleId | rightId |
+-----+

```

rightrole **JPA** Embeddable ◦

## rightrole

```

@Entity
@Table(name = "rightrole")
public class RightRole extends BaseEntity<RightRolePK> {

    private static final long serialVersionUID = 1L;

    @EmbeddedId
    protected RightRolePK rightRolePK;

    @JoinColumn(name = "roleId", referencedColumnName = "roleId", insertable = false,
updatable = false)
    @ManyToOne(fetch = FetchType.LAZY)
    private Role role;

    @JoinColumn(name = "rightID", referencedColumnName = "rightID", insertable = false,
updatable = false)
    @ManyToOne(fetch = FetchType.LAZY)
    private Right right;

    .....
}

@Embeddable
public class RightRolePK implements Serializable {
private static final long serialVersionUID = 1L;

    @Basic(optional = false)
    @NotNull

```

```

    @Column(nullable = false)
    private long roleID;

    @Basic(optional = false)
    @NotNull
    @Column(nullable = false)
    private long rightID;

    .....
}

```

◦

```
rightsrole store(persist) roleflushid ◦ rightrole◦
```

◦

```

@Entity
@Table(name = "role")
public class Role {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @NotNull
    @Column(nullable = false)
    private Long roleID;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "role", fetch = FetchType.LAZY)
    private List<RightRole> rightRoleList;

    @ManyToMany(cascade = {CascadeType.PERSIST})
    @JoinTable(name = "rightrole",
        joinColumns = {
            @JoinColumn(name = "roleID", referencedColumnName = "ROLE_ID")},
        inverseJoinColumns = {
            @JoinColumn(name = "rightID", referencedColumnName = "RIGHT_ID")})
    private List<Right> rightList;

    .....
}

```

**@JoinTable**rightrolerolerightid◦

```

Role role = new Role();
List<Right> rightList = new ArrayList<>();
Right right1 = new Right();
Right right2 = new Right();
rightList.add(right1);
rightList.add(right2);
role.setRightList(rightList);

```

**inverseJoinColumns@ManyToManycascade = {CascadeType.PERSIST}**◦

<https://riptutorial.com/zh-TW/jpa/topic/6532/>

---

# 9:

- 
- 
- 

## Examples

- ◦
- ◦
- ◦
- ◦

◦ @OneToOne◦

*VehicleParkingPlace*◦

◦

@OneToMany◦

mappedBy **ManyToOne**

```
@OneToMany (mappedBy="attribute")
```

◦

@ManyToOne◦

@JoinColumn (name="FK\_name") **foreign**◦

◦

@ManyToMany◦

◦

## @JoinTable

JPA@JoinTable

```
@Entity  
public class EntityA {  
    @Id
```

```

@Column(name="id")
private long id;
[...]
@ManyToMany
@JoinTable(name="table_join_A_B",
           joinColumns=@JoinColumn(name="id_A", referencedColumnName="id",
                                   inverseJoinColumns=@JoinColumn(name="id_B", referencedColumnName="id"))
private List<EntityB> entitiesB;
[...]
}

@Entity
public class EntityB {
    @Id
    @Column(name="id")
    private long id;
    [...]
}

```

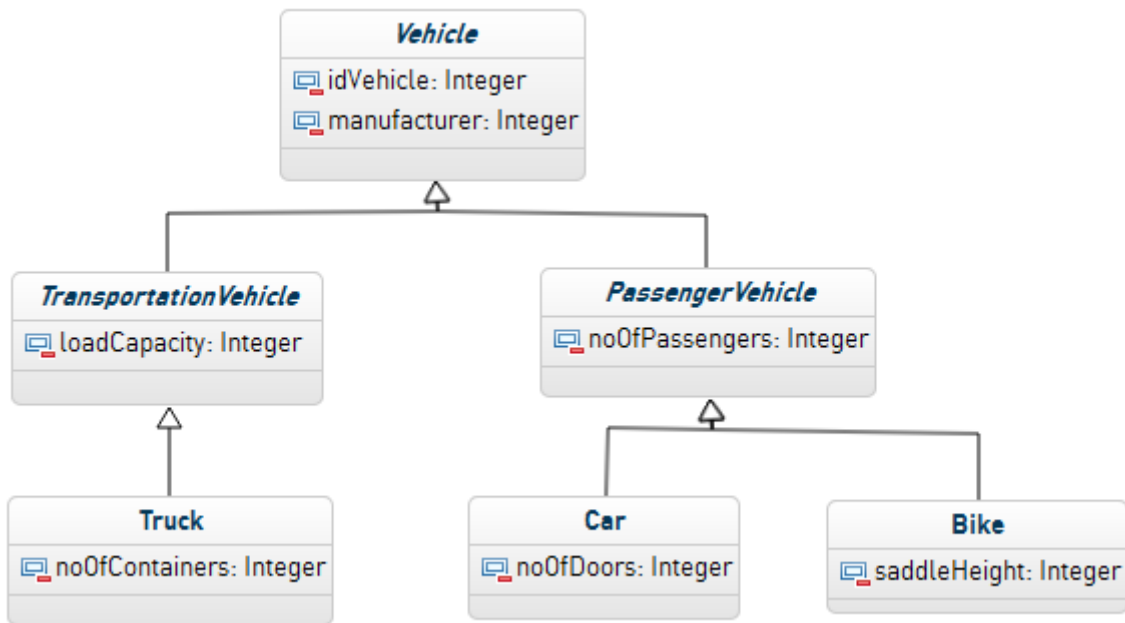
EntityAEntityB @JoinTable table\_join\_A\_B id\_Aid\_B EntityAEntityB id ; (id\_A, id\_B)  
table\_join\_A\_B°

<https://riptutorial.com/zh-TW/jpa/topic/6305/>

# 10:

- VehicleTransportationVehiclePassengerVehicle。
- ◦ @MappedSuperClass。
- TruckTransportationVehicleVehicle。
- CarBikePassengerVehicleVehicle。

## Examples



### Vehicle.java

```
package com.thejavageek.jpa.entities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.TableGenerator;

@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Vehicle {

    @TableGenerator(name = "VEHICLE_GEN", table = "ID_GEN", pkColumnName = "GEN_NAME",
valueColumnName = "GEN_VAL", allocationSize = 1)
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "VEHICLE_GEN")
    private int idVehicle;
    private String manufacturer;
}
```

```

public int getIdVehicle() {
    return idVehicle;
}

public void setIdVehicle(int idVehicle) {
    this.idVehicle = idVehicle;
}

public String getManufacturer() {
    return manufacturer;
}

public void setManufacturer(String manufacturer) {
    this.manufacturer = manufacturer;
}
}

```

## TransportationVehicle.java

```

package com.thejavageek.jpa.entities;

import javax.persistence.MappedSuperclass;

@MappedSuperclass
public abstract class TransportationVehicle extends Vehicle {

    private int loadCapacity;

    public int getLoadCapacity() {
        return loadCapacity;
    }

    public void setLoadCapacity(int loadCapacity) {
        this.loadCapacity = loadCapacity;
    }

}

```

## Truck.java

```

package com.thejavageek.jpa.entities;

import javax.persistence.Entity;

@Entity
public class Truck extends TransportationVehicle {

    private int noOfContainers;

    public int getNoOfContainers() {
        return noOfContainers;
    }

    public void setNoOfContainers(int noOfContainers) {
        this.noOfContainers = noOfContainers;
    }

}

```

## PassengerVehicle.java

```
package com.thejavageek.jpa.entities;

import javax.persistence.MappedSuperclass;

@MappedSuperclass
public abstract class PassengerVehicle extends Vehicle {

    private int noOfpassengers;

    public int getNoOfpassengers() {
        return noOfpassengers;
    }

    public void setNoOfpassengers(int noOfpassengers) {
        this.noOfpassengers = noOfpassengers;
    }

}
```

## Car.java

```
package com.thejavageek.jpa.entities;

import javax.persistence.Entity;

@Entity
public class Car extends PassengerVehicle {

    private int noOfDoors;

    public int getNoOfDoors() {
        return noOfDoors;
    }

    public void setNoOfDoors(int noOfDoors) {
        this.noOfDoors = noOfDoors;
    }

}
```

## Bike.java

```
package com.thejavageek.jpa.entities;

import javax.persistence.Entity;

@Entity
public class Bike extends PassengerVehicle {

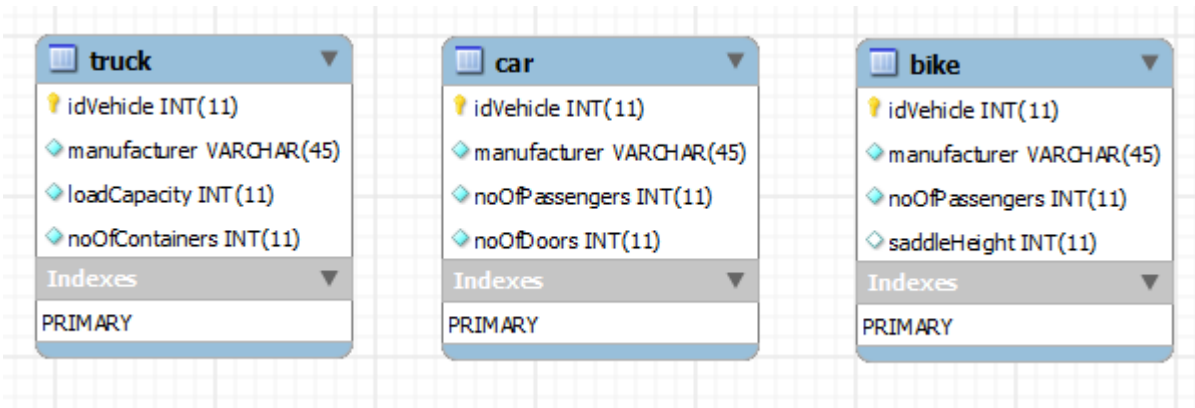
    private int saddleHeight;

    public int getSaddleHeight() {
        return saddleHeight;
    }

    public void setSaddleHeight(int saddleHeight) {
```



```
        this.saddleHeight = saddleHeight;
    }
}
```



<https://riptutorial.com/zh-TW/jpa/topic/6255/>

S. No		Contributors
1	jpa	Billy Frost, Community, DimaSan, , Manuel Spigolon, Michael Piefel, Neil Stockton, ppeterka
2		Prasad Kharkar
3		Prasad Kharkar
4		bw_üezi, Prasad Kharkar
5		Prasad Kharkar
6		Jeffrey Brett Coleman, Michael Piefel, Neil Stockton, Pete
7		Prasad Kharkar
8		Prasad Kharkar, Ronak Patel, Vetle
9		DimaSan,
10		Prasad Kharkar