HEXAWARE

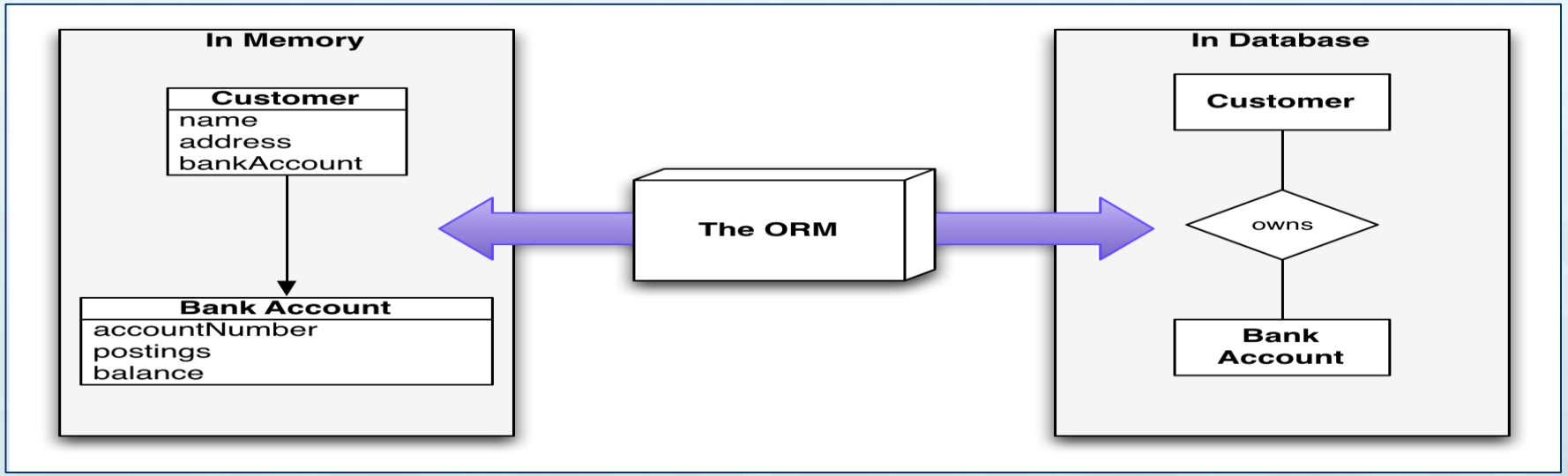
Hibernate

Session Objective

- ORM
- JPA
- Hibernate
- Hibernate Overview
- Hibernate Configuration
- Hibernate relationship
- Hibernate Inheritance
- Hibernate Association

Object Relational Mapping – ORM

Object-relational mapping (ORM, O/RM, and O/R mapping tool) in computer science is a programming technique for converting data between incompatible type systems using object-oriented programming languages.



Java Persistence API (Jakarta Persistence)

JPA is a Java application programming interface specification that describes the management of relational data in applications.

API defines :

- ➔ javax.persistence package
- ➔ Java Persistence Query Language (JPQL)
- ➔ object/relational metadata

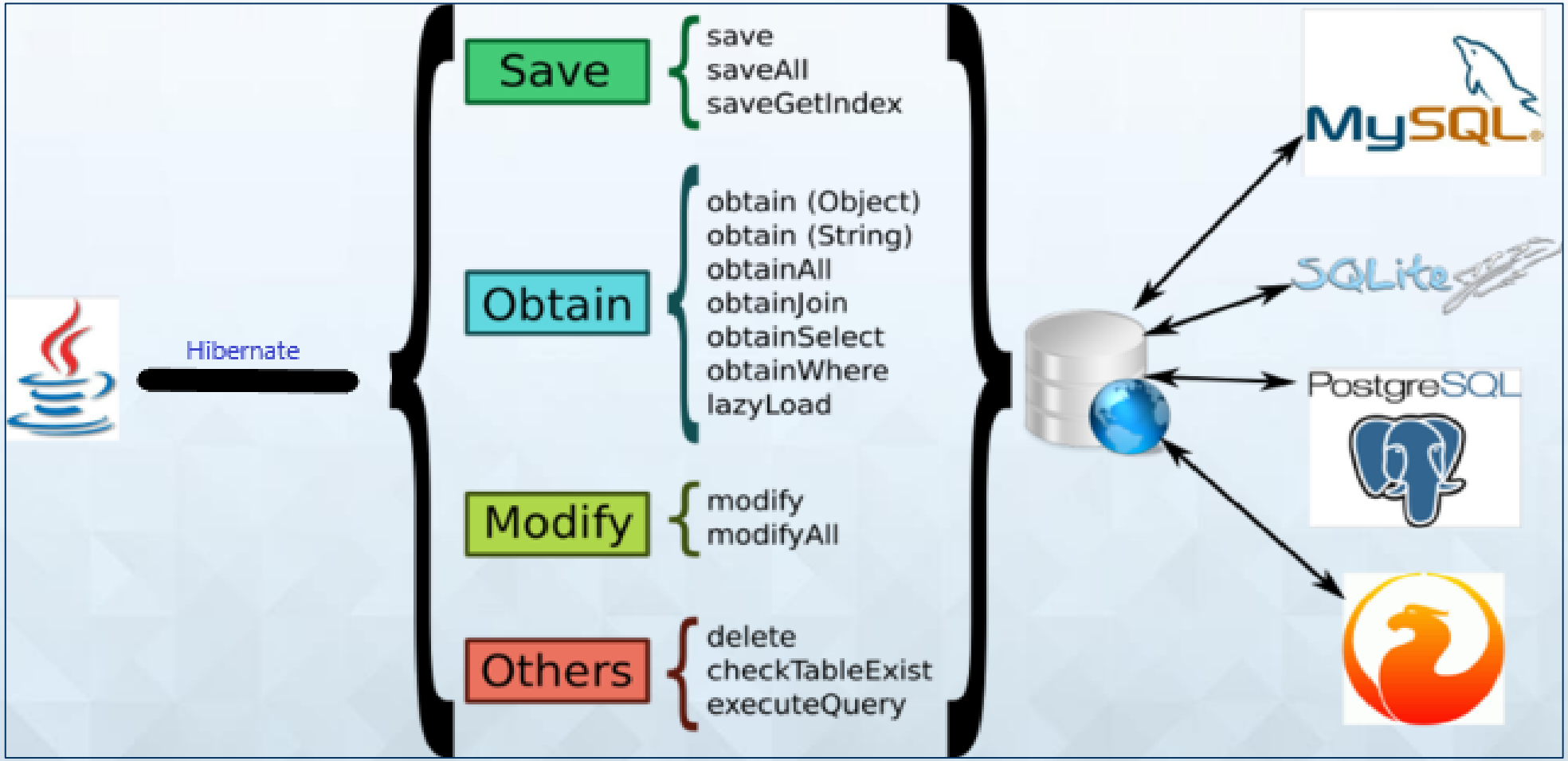
The reference implementation for JPA is EclipseLink.

Java Persistence API (Jakarta Persistence)

Introduction to Hibernate

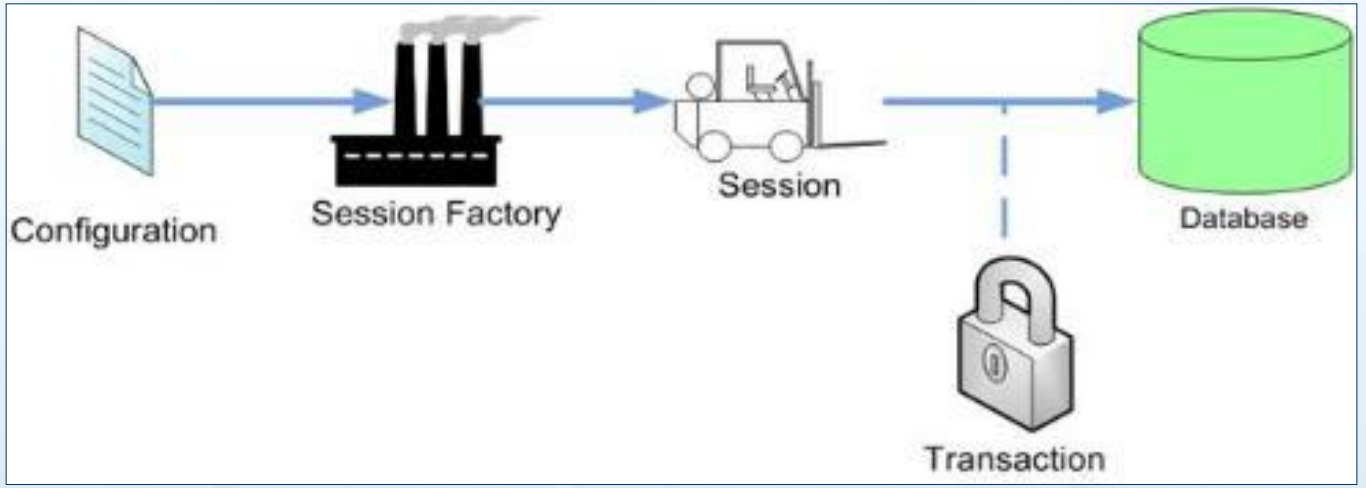
- ➔ Hibernate is a Java framework that simplifies the development of Java application to interact with the database.
- ➔ It is an open source, lightweight, ORM (Object Relational Mapping) tool.
- ➔ Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

Introduction to Hibernate (Contd.)



Overview of Hibernate Architecture (Contd.)

- The following figure shows the working of the Hibernate classes and interfaces.



The `SessionFactory` object retrieves the configuration details from the `Configuration` object. The `Session` object uses this configuration details to send and retrieve data from the database with the help of a `Transaction` object.

Configuring Hibernate

- To connect with a database in Hibernate application, you need to set various properties regarding driver class, user name, and password.
- These properties can be set in an XML file known as Hibernate configuration file or hibernate.cfg.xml file.

Configuring Hibernate (Contd.)

- The structure of the hibernate.cfg.xml file is given in the following code snippet:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC "-
  //Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://hibernate.sourceforge.net/hibernate-
  configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
   <property
  name="connection.username">SYSTEM</property>
   .......
   <mapping resource="Mapped_File.hbm.xml" />
   </session-factory>
 </hibernate-configuration>
```

Defines the version and encoding type used for the XML document.

Specifies the Document Type Definition (DTD) for the XML elements. DTD specifies the grammar rule for the XML document.

Configuring Hibernate (Contd.)

- The following properties are commonly set under the `<property>` tag:
  - JDBC properties:
    - These are used to connect with a relational database.
    - The following JDBC properties are commonly used in a Hibernate configuration file:
      - `hibernate.connection.driver_class`
      - `hibernate.connection.url`
      - `hibernate.connection.username`
      - `hibernate.connection.password`

Specifies the database specific driver name, User Name, Password, URL, port number to connect..

Configuring Hibernate (Contd.)

- Hibernate configuration properties:
- These control the behavior of Hibernate at run time.
- The following configurations are commonly used in the Hibernate configuration file:
  - hibehibernate.dialect
- rnate.show_sql

Specifies the database that is used to communicate with the application.

Specifies that the SQL statements are written on the console during the execution of the application.

Configuring Hibernate (Contd.)

- Hibernate miscellaneous properties:
- Specify the optional properties that are not mandatory to connect with a database.
- For example:
- hibernate.current_session_context_class

Specifies the controller class that controls the scope of the current session in a Hibernate application.

Creating a Hibernate Session

- In a Hibernate application, a session acts as a pipeline between the application and the database.
- A session object in an application is created by using a SessionFactory object.
- To create a SessionFactory object, the application needs various database specific information, such as the:
  - Database driver name
  - User name
  - Password used to connect to the database

Creating a Hibernate Session (Contd.)

- Hibernate specifies the configuration settings and information about the mapping document in an XML file named hibernate.cfg.xml.
- The information stored in this file is used to create a SessionFactory object.

Creating a Hibernate Session (Contd.)

- The org.hibernate.cfg.AnnotationConfiguration class:
- Allows the application to specify the configuration properties and mapping documents to create a SessionFactory object.
- Uses the following methods to create a SessionFactory object:
- configure()
- buildSessionFactory()

Loads the hibernate.cfg.xml file and initializes the object of the `AnnotationConfiguration` class with the mappings and configuration properties specified in this file.

Uses the `AnnotationConfiguration` object returned by the `configure()` method to instantiate a new `SessionFactory` object.

Creating a Hibernate Session (Contd.)

- For example:

```
SessionFactory sessionFactory = new
   AnnotationConfiguration().configure().buildSessionFactory();
```

Creates an object of the `SessionFactory` interface named `sessionFactory`.

- You can use this object to create a session object, as shown in the following code snippet:

The `openSession()` method of the `SessionFactory` interface returns a `Session` object.

```
Session session = sessionFactory.openSession();
```

- A single instance of the SessionFactory object is required if the application needs to communicate with one database.
- However, if there are multiple databases, you can create multiple SessionFactory objects, one for each database.
- The SessionFactory object must be instantiated once during application initialization.

Creating a Hibernate Session (Contd.)

- You can create a Hibernate utility class that takes care of application startup and enables easy retrieval of Hibernate SessionFactory objects.
- This class uses the static global variable and static initialization to create the SessionFactory object.

Creating a Hibernate Session (Contd.)

- For example:

```
public class HibernateUtil {
 private static final SessionFactory sessionFactory;
 static {
    try { sessionFactory = new
   AnnotationConfiguration().configure().buildSession
   Factory();
    } catch (Throwable ex) {
        System.err.println("Initial SessionFactory
           creation failed." + ex);
        throw new ExceptionInInitializerError(ex);
    }
 }
 public static SessionFactory getSessionFactory() {
    return sessionFactory;}}
```

The `getSessionFactory()` method returns the `SessionFactory` object.

Is created to build the `SessionFactory` object. This class contains a static initializer block to start up Hibernate.

Creating a Hibernate Session (Contd.)

- You can use the getSessionFactory() method to access the Hibernate session in your application, as shown in the following code snippet:

```
Session session =
  HibernateUtil.getSessionFactory().getCurrentSession
  ();
```

The `getSessionFactory()` method is used to access the `SessionFactory` object. This object then executes the `getCurrentSession()` method to open a new Hibernate session if no session exists.

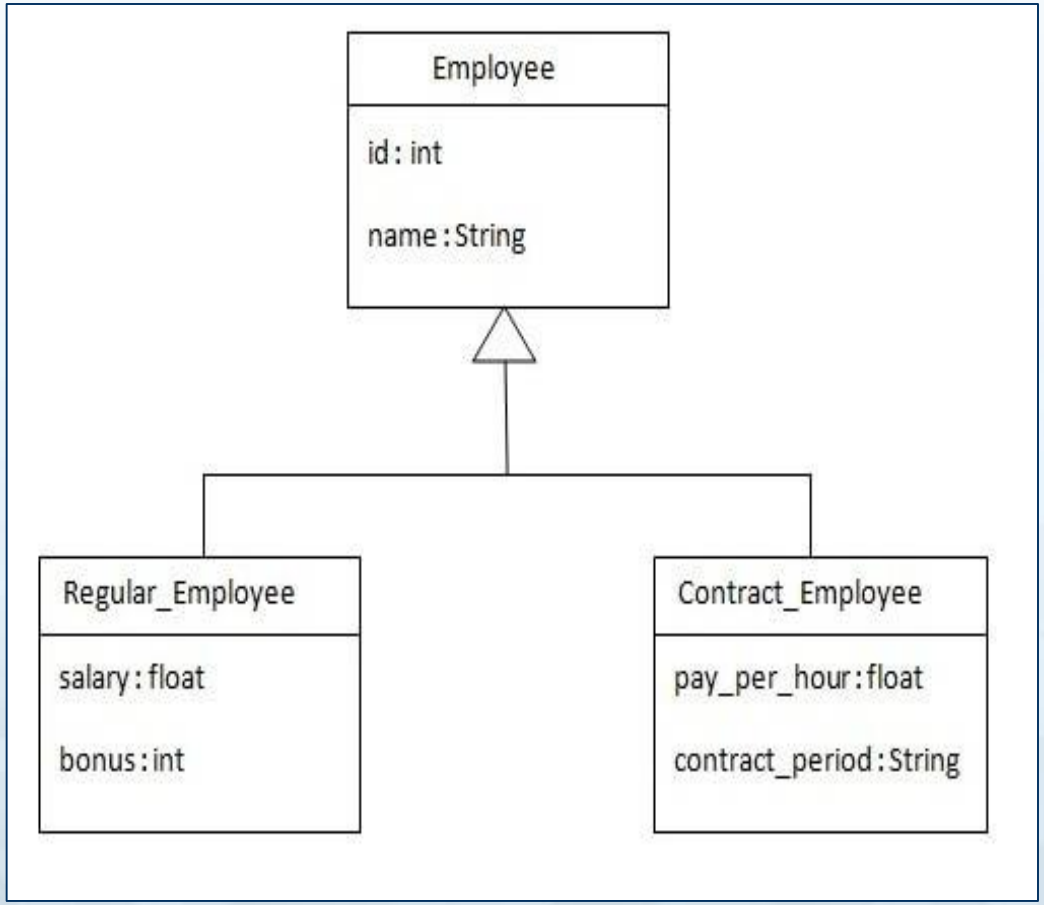
Let us see how to create a Database and Tables.

Let us see how to create a Hibernate Application.

Hibernate - Inheritance

- Table per Hierarchy.
- Table per Concrete class
- Table per class
- Many to Many

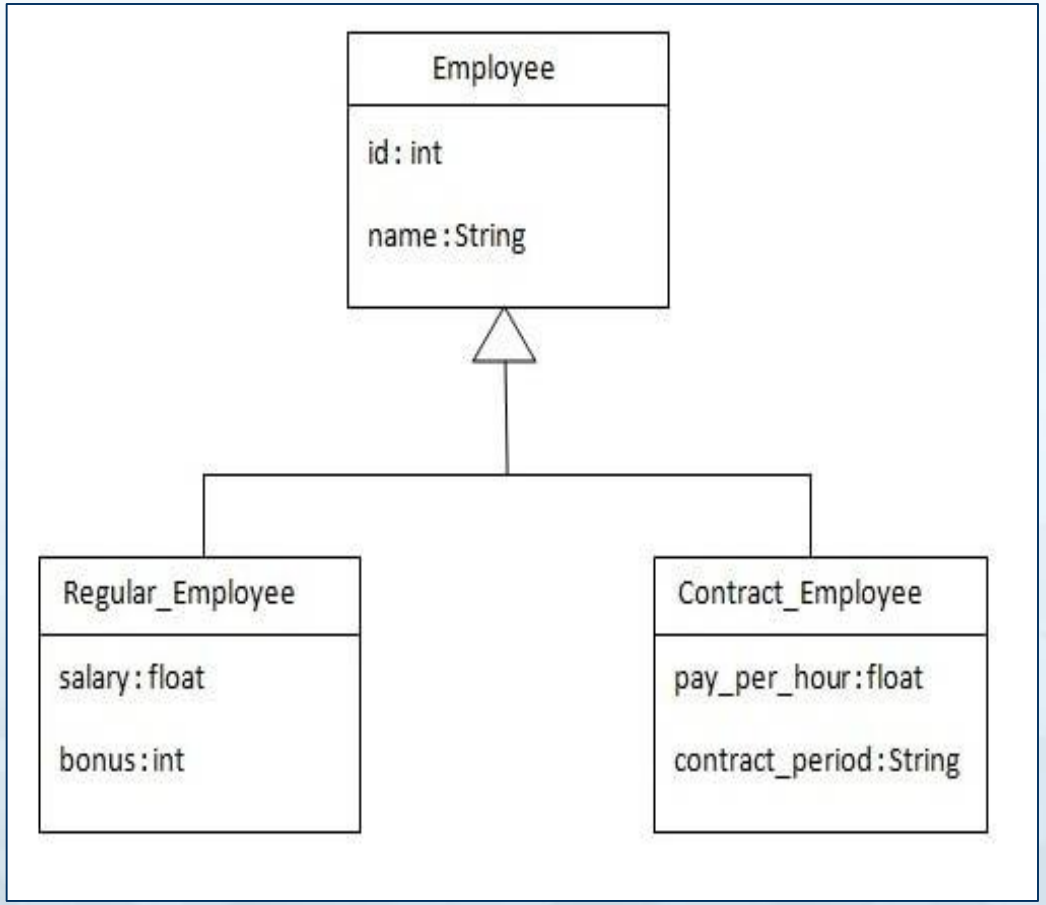
Table per Hierarchy



Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
TYPE	VARCHAR2(255)	No	-	-
NAME	VARCHAR2(255)	Yes	-	-
SALARY	FLOAT	Yes	-	-
BONUS	NUMBER(10,0)	Yes	-	-
PAY_PER_HOUR	FLOAT	Yes	-	-
CONTRACT_DURATION	VARCHAR2(255)	Yes	-	-

1 - 7

Table per Concrete class



Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-

1 - 2

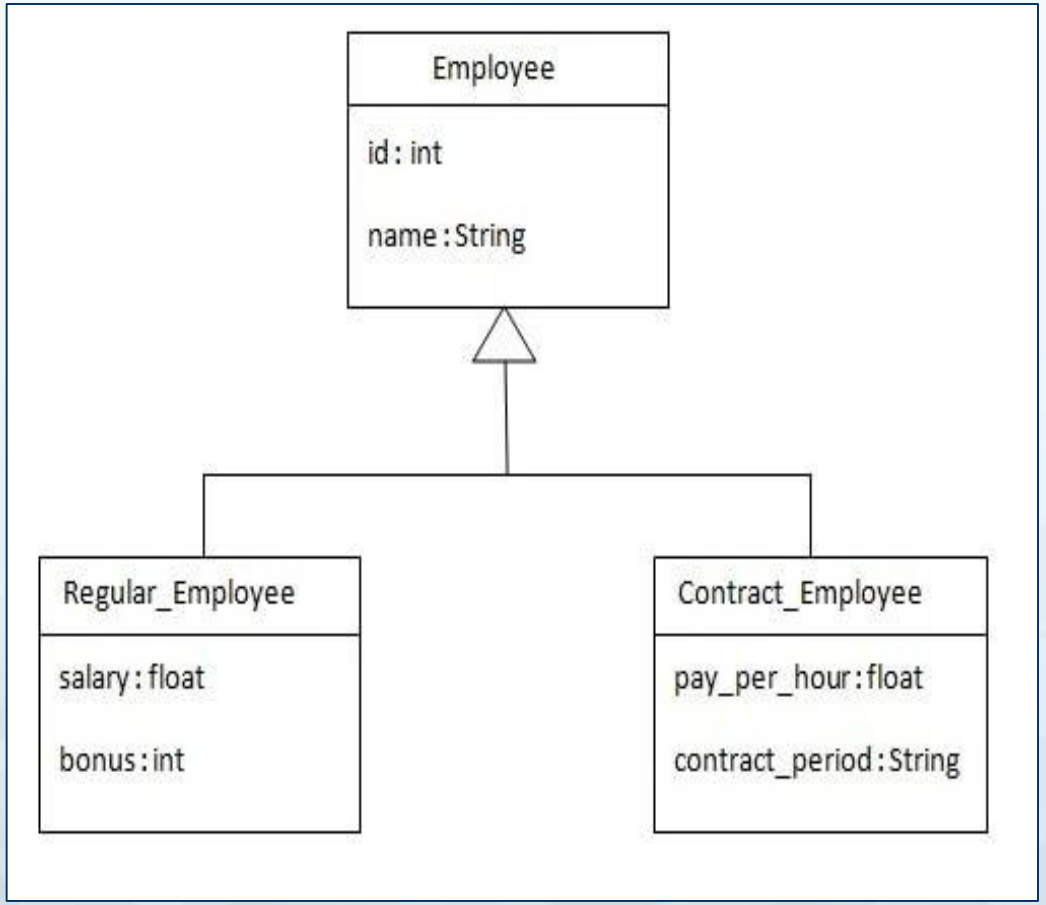
Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
SALARY	FLOAT	Yes	-	-
BONUS	NUMBER(10,0)	Yes	-	-

1 - 4

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
PAY_PER_HOUR	FLOAT	Yes	-	-
CONTRACT_DURATION	VARCHAR2(255)	Yes	-	-

1 - 4

Table per Sub class



Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(10,0)	No	-	1
NAME	VARCHAR2(255)	Yes	-	-
				1 - 2

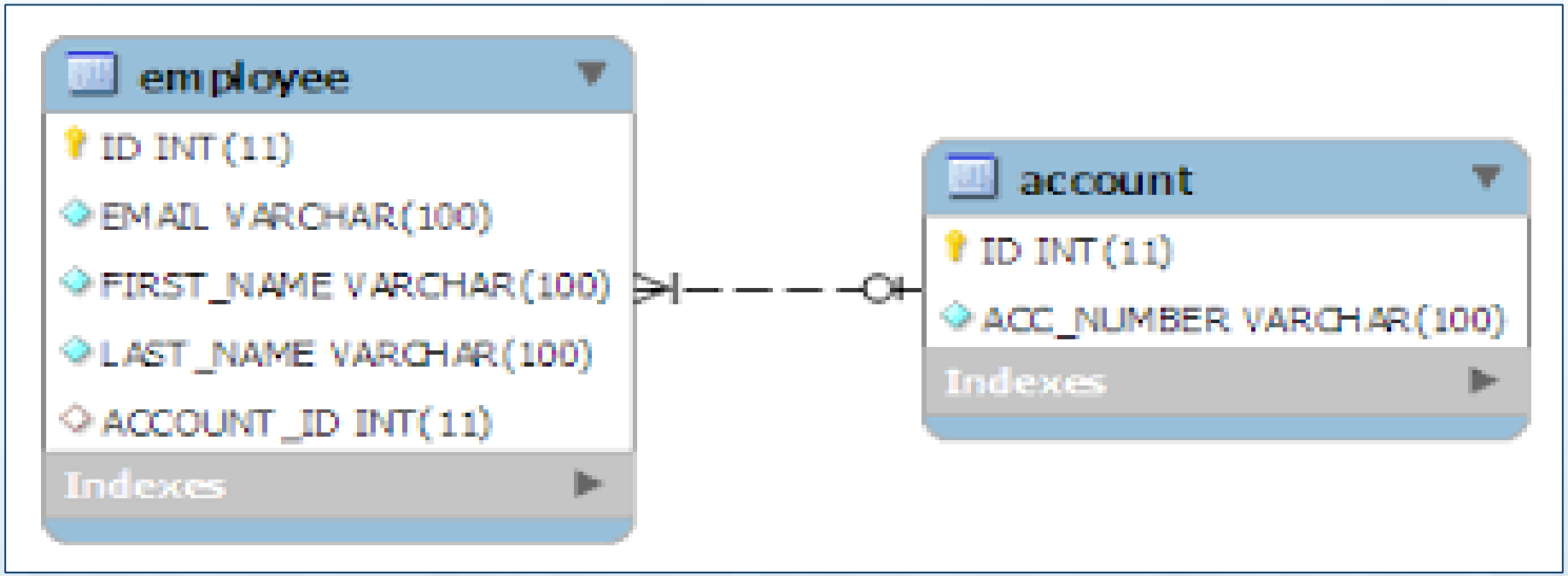
Column Name	Data Type	Nullable	Default	Primary Key
EID	NUMBER(10,0)	No	-	1
SALARY	FLOAT	Yes	-	-
BONUS	NUMBER(10,0)	Yes	-	-
				1 - 3

Column Name	Data Type	Nullable	Default	Primary Key
EID	NUMBER(10,0)	No	-	1
PAY_PER_HOUR	FLOAT	Yes	-	-
CONTRACT_DURATION	VARCHAR2(255)	Yes	-	-
				1 - 3

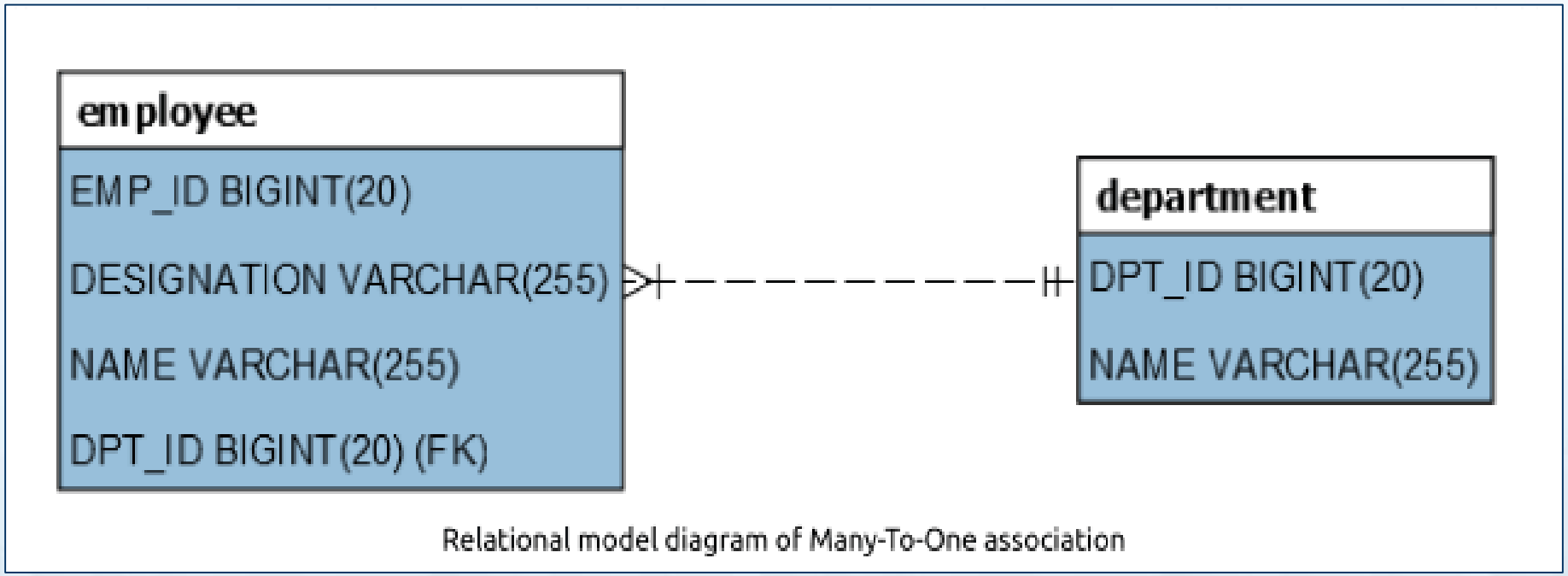
Hibernate - Association

- One to One
- One to Many
- Many to One
- Many to Many

One to One



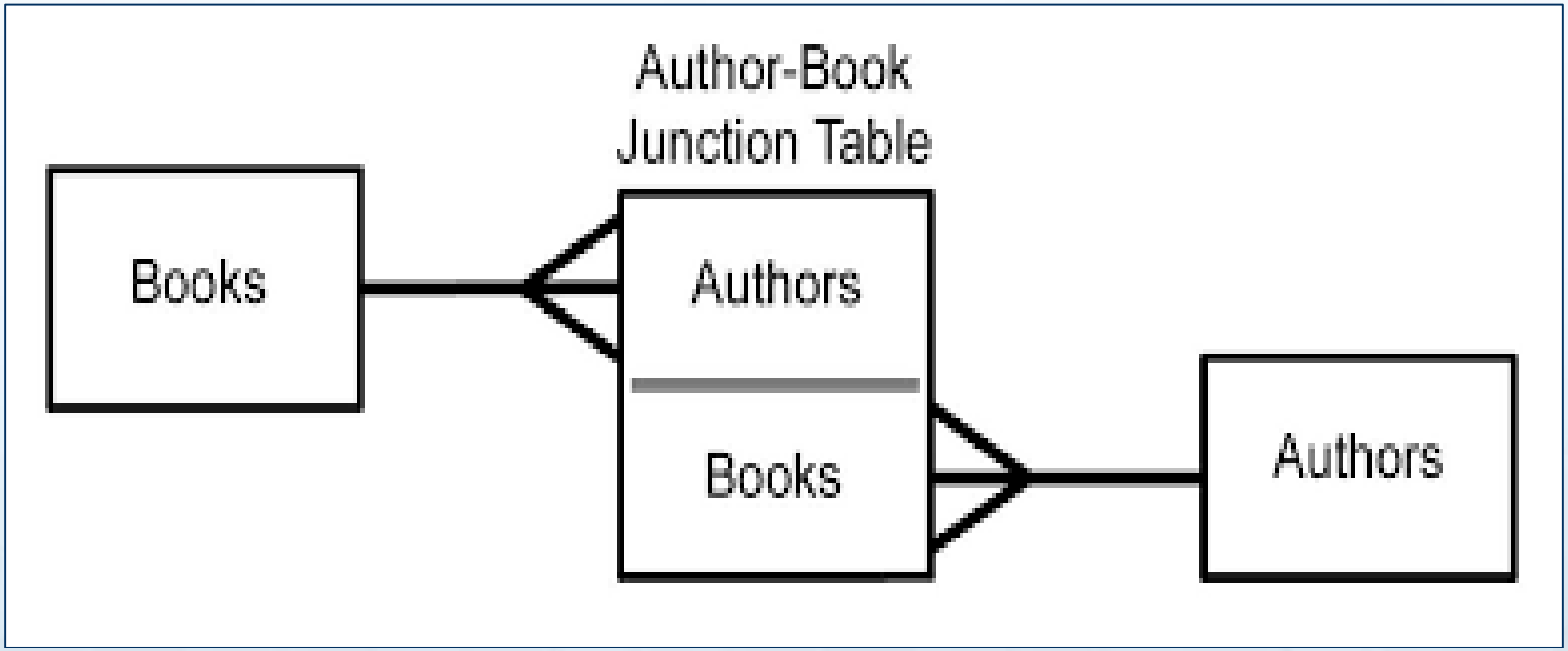
One to Many



Relational model diagram of Many-To-One association

Many to One

Many to Many



Explore More about hibernate with xml and annotation.

Summary

In this session, We learned the

- ➔Relationship between ORM, JPA and Hibernate
- ➔Configuration file and its properties
- ➔ Relationship
  - ➔ Inheritance and its type.
  - ➔ Association and its type.

HEXAWARE

Innovative Services

Passionate Employees

Delighted Customers

Thank you

www.hexaware.com