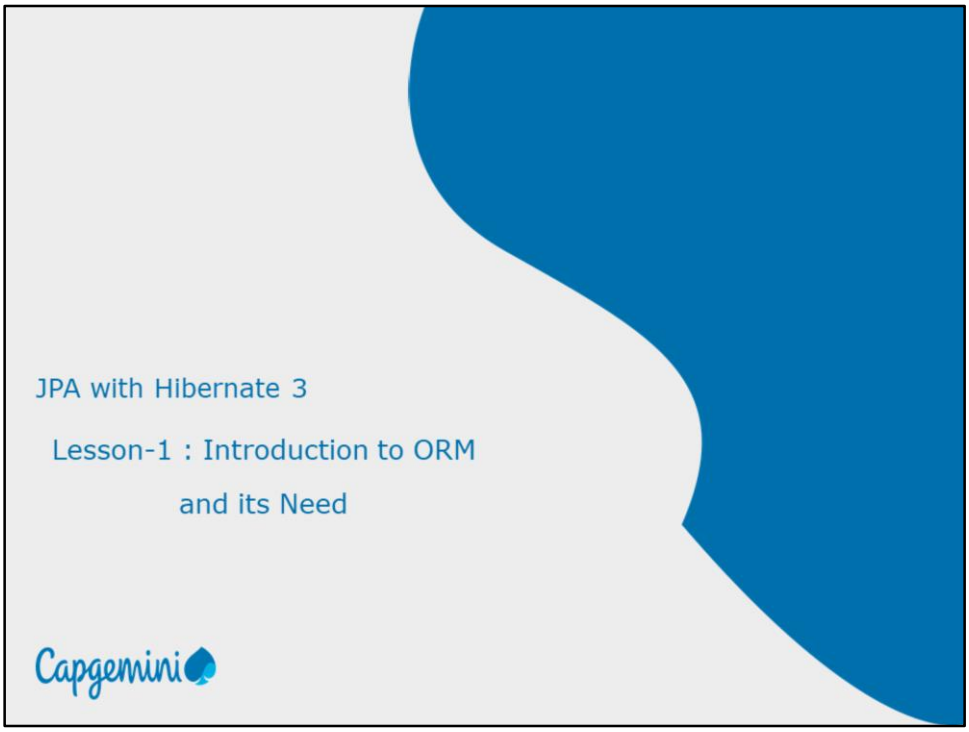
Instructor Notes:

Add instructor notes here.



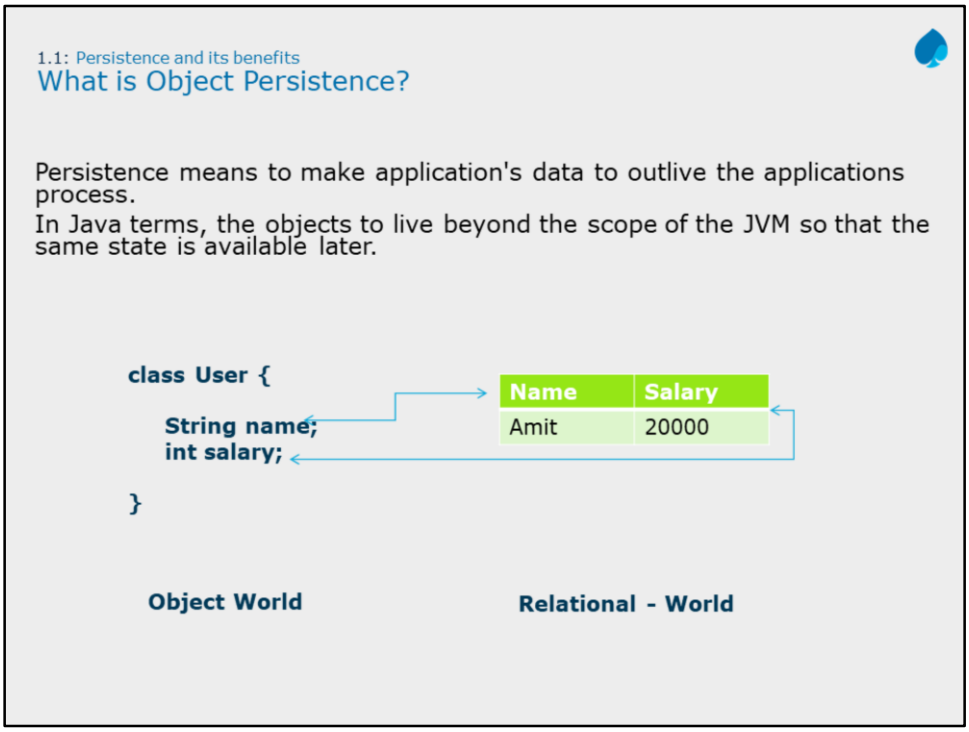
Instructor Notes:

Lesson Objectives

After completing this lesson, participants will be able to understand:

- Persistence and its benefits
- Object-relational Impedance Mismatch
- Object/Relational Mapping
- ORM and its need
- JPA and its benefits

Instructor Notes:

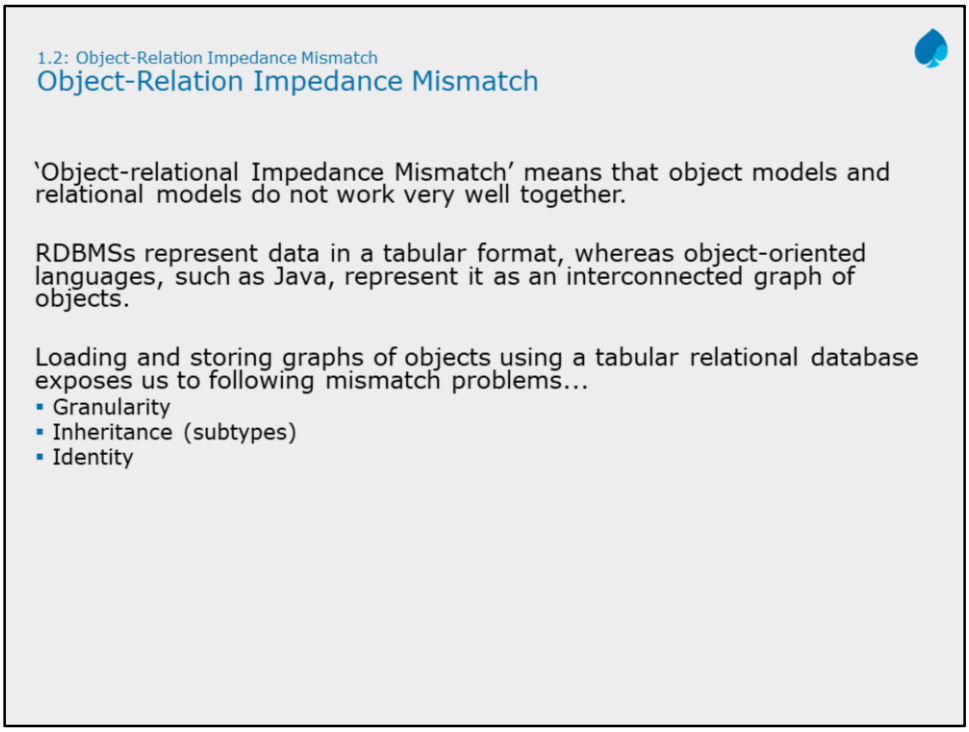


The above diagram depicts mapping of object state into database table columns. To do so, traditionally, we rely on JDBC API, which allows developers to save application data into database, however conversion is required from object format to database table format which un-necessarily increases line of code .

However, there are lot of challenges and mismatch in data processing in these two models. In addition, if database changes, then developer need to make modification in the configuration which is database specific.

So to shorten the development time, and to save application object directly into database, there was need to reinvent the approach of mapping object and relational model.

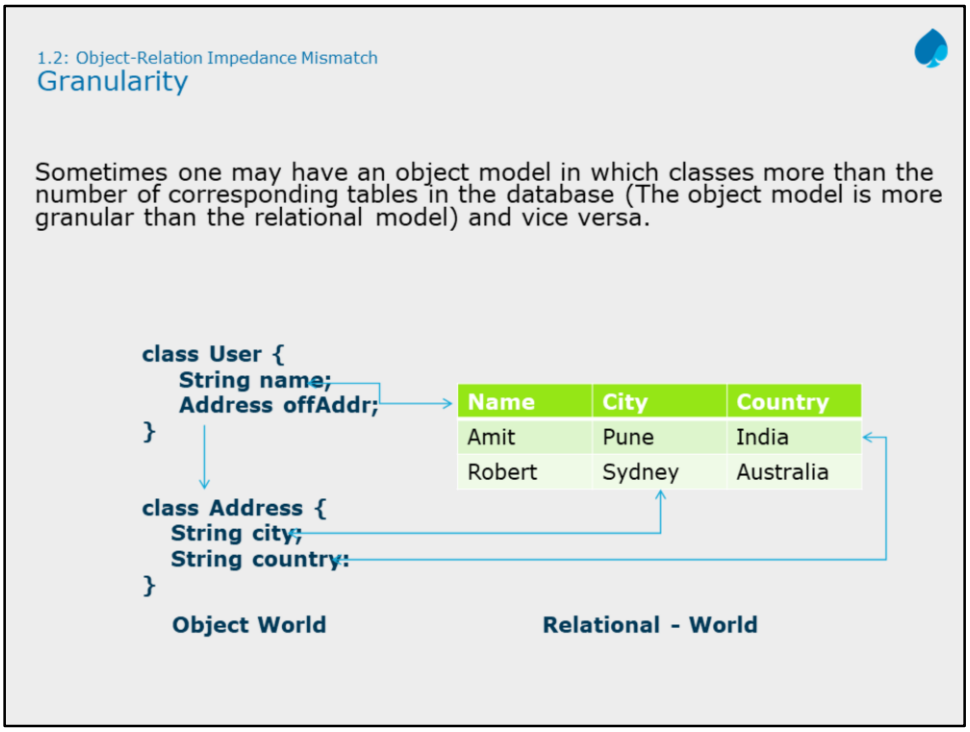
Instructor Notes:



The above slide demonstrates on the challenges of mapping objects and database relations.

Instructor Notes:

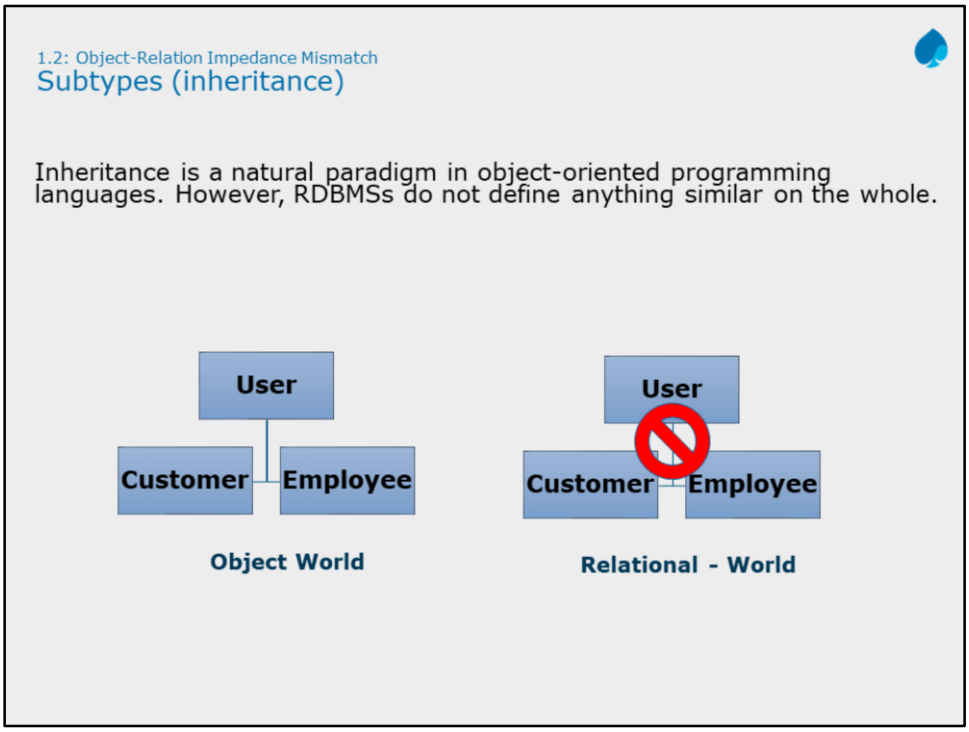
Discuss the scenarios real world scenarios to discuss the Granularity



Above diagram shows how data of two entities are persisted into one relation (table)
Conversion of the java datatypes to underlying database types is done by ORM automatically.

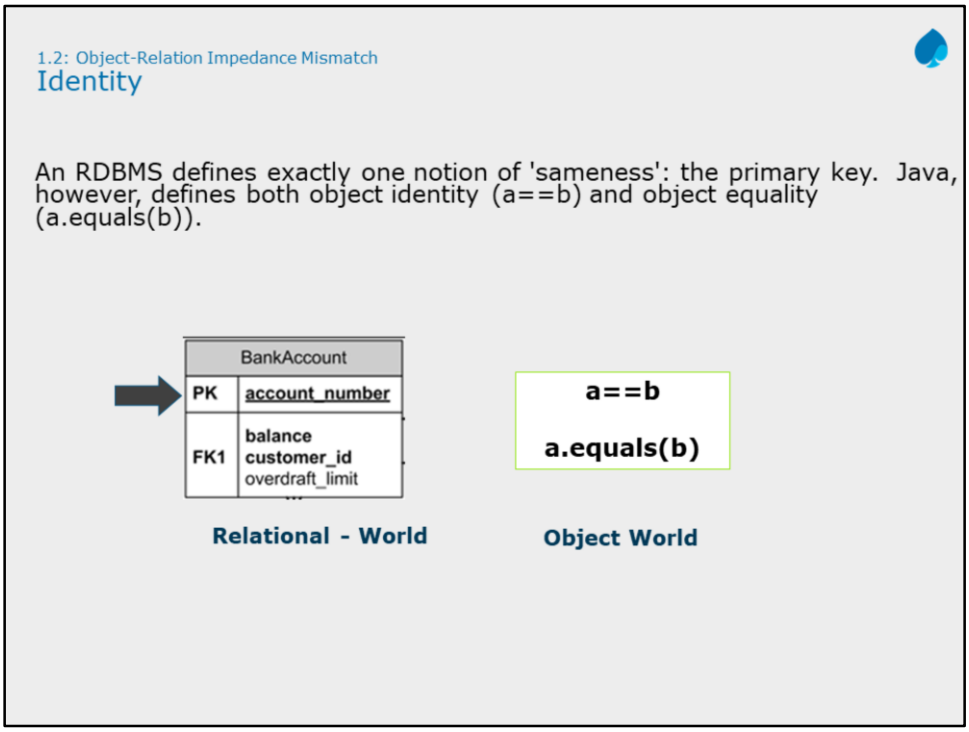
Instructor Notes:

Add instructor notes here.



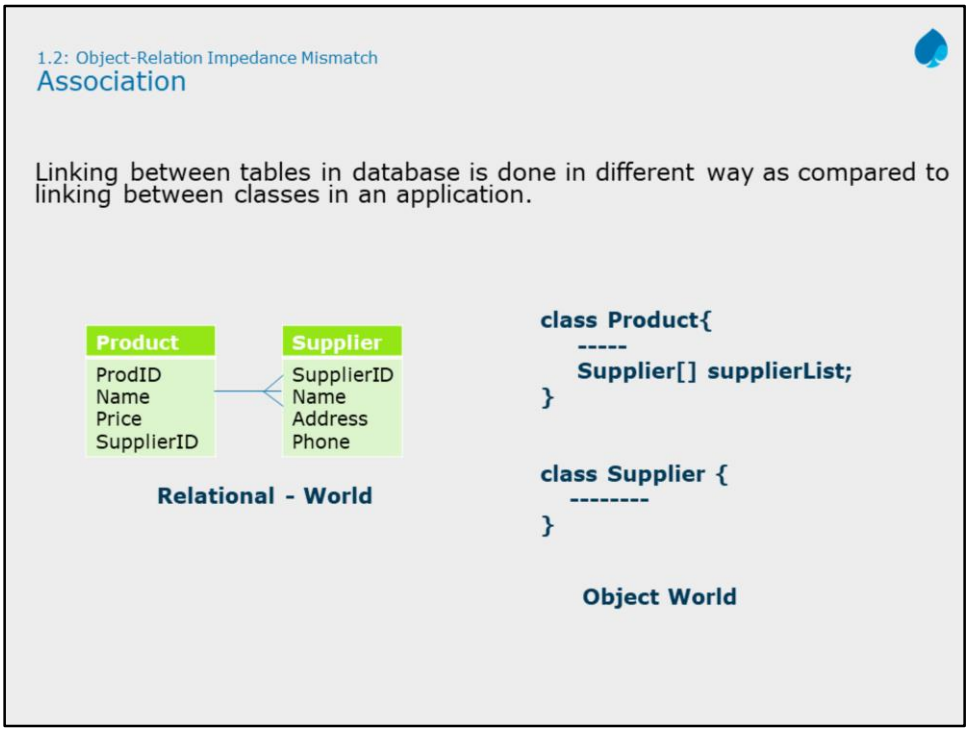
Instructor Notes:

Add instructor notes here.



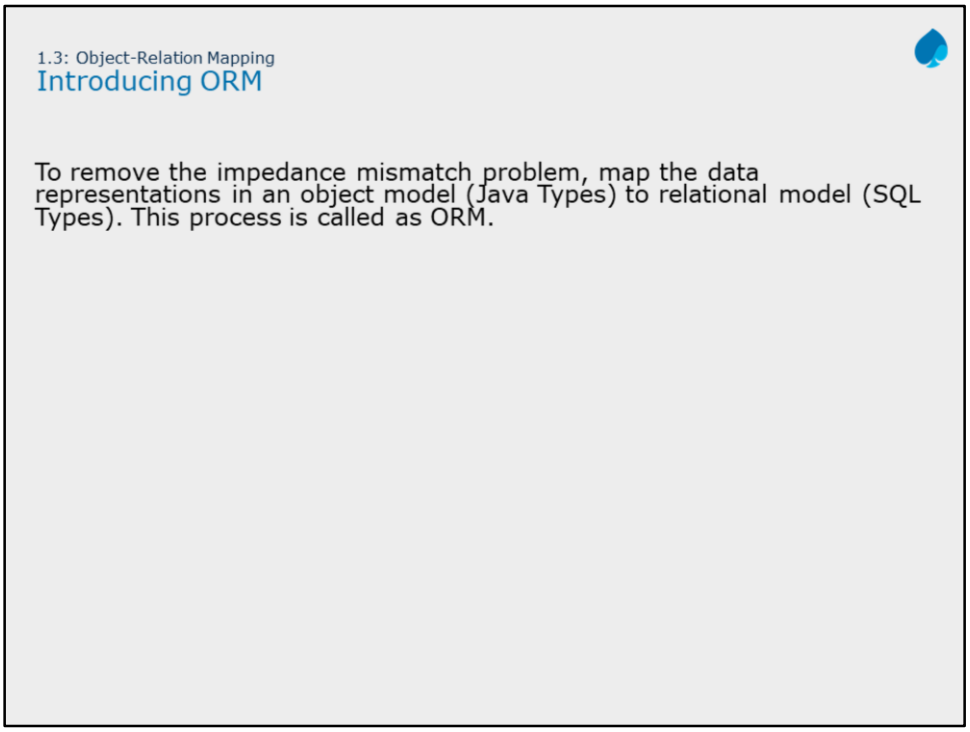
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.



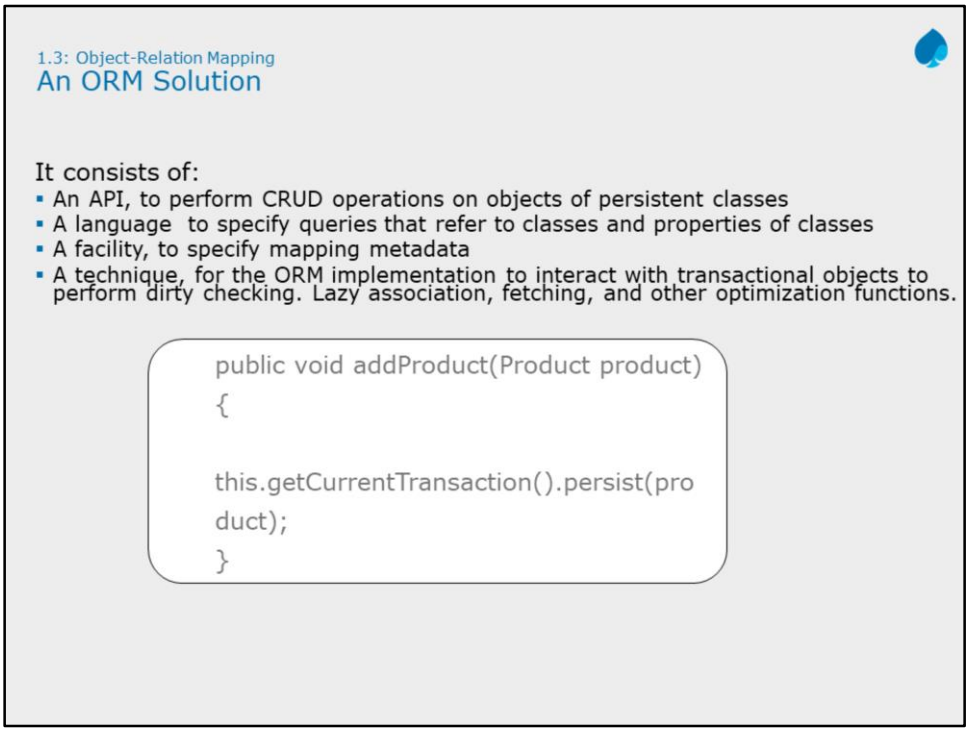
Storing object-oriented entities in a relational database is often not a simple task and requires a great deal of repetitive code along with conversion between data types.

Object-relational mapper, or O/RM, were created to solve this problem. An O/RM persists entities in and retrieves entities from relational databases without the programmer having to write SQL statements and translate entity properties to statement parameters and result set columns to entity properties.

Instructor Notes:

Compare the given code snippet with JDBC insert task. Do mention about much of the boilerplate code now goes behind the screen like:

1. Obtaining connection
2. Creating statement
3. Converting Object types to RDBMS types
4. Closing opened resources



Dirty Checking:

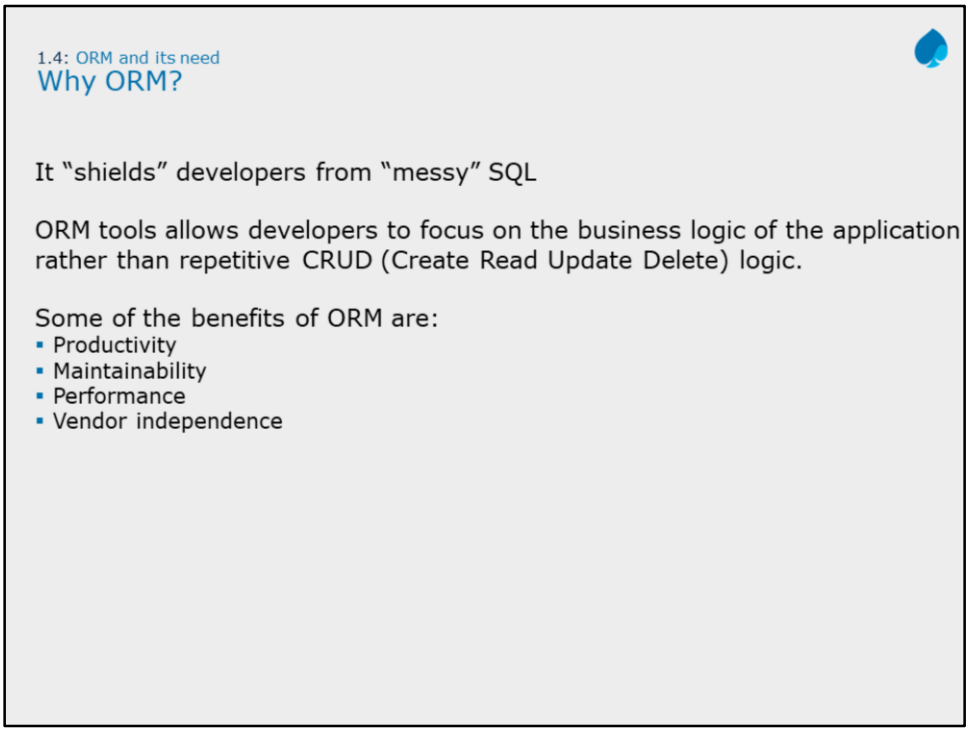
A dirty checking feature avoids unnecessary database write actions by performing SQL updates only on the modified fields of persistent objects. For example, If you modify salary of employee on object model, only salary field will be updated instead of updating entire employee object.

Lazy association fetching:

Lazy fetching decides whether to load child objects while loading the Parent Object. For example, Consider department entity consist of many employees, and someone query to fetch department details, ORM fetches only department details and defers loading employees. This will be done, when one request details of employees working in that department.

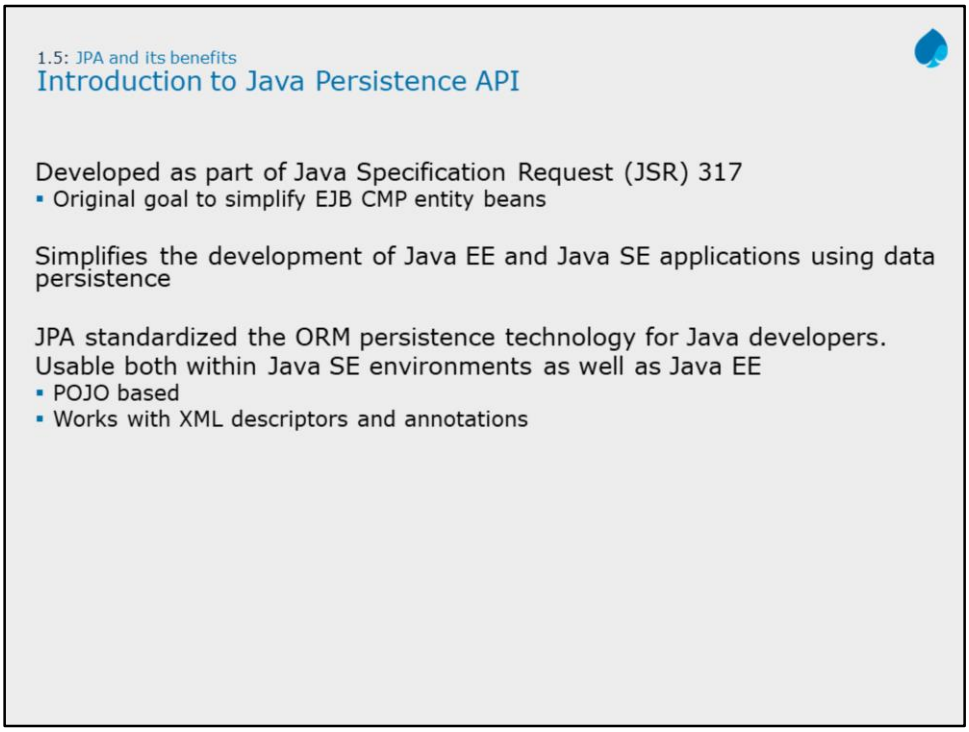
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.

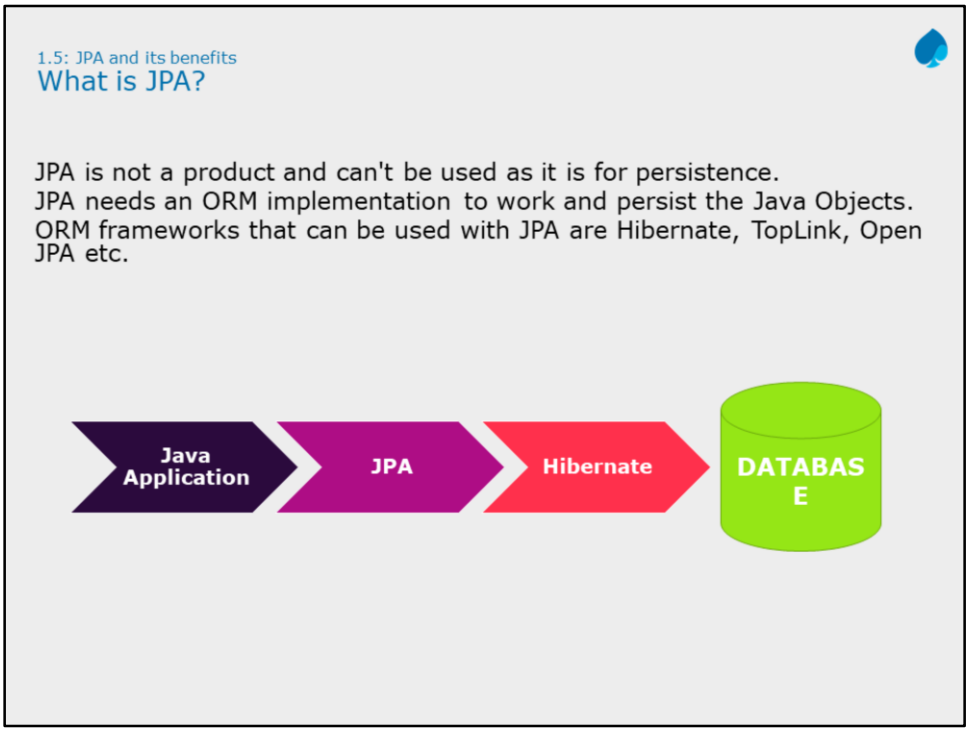


JPA is just an specification from Sun, which is released under JEE 5 specification. JPA standardized the ORM persistence technology for Java developers. JPA is not a product and can't be used as it is for persistence. It needs an ORM implementation to work and persist the Java Objects. ORM frameworks that can be used with JPA are Hibernate, Toplink, Open JPA etc.

The Java Persistence API (JPA) is one approach to ORM. Via JPA the developer can map, store, update and retrieve data from relational databases to Java Objects and vice versa, JPA permits the developer to work directly with objects rather then with SQL statements. JPA is a specification and several implementations are available

Instructor Notes:

JPA not only provides database-independence but it also gives us a flexibility to have ORM implementer independence.



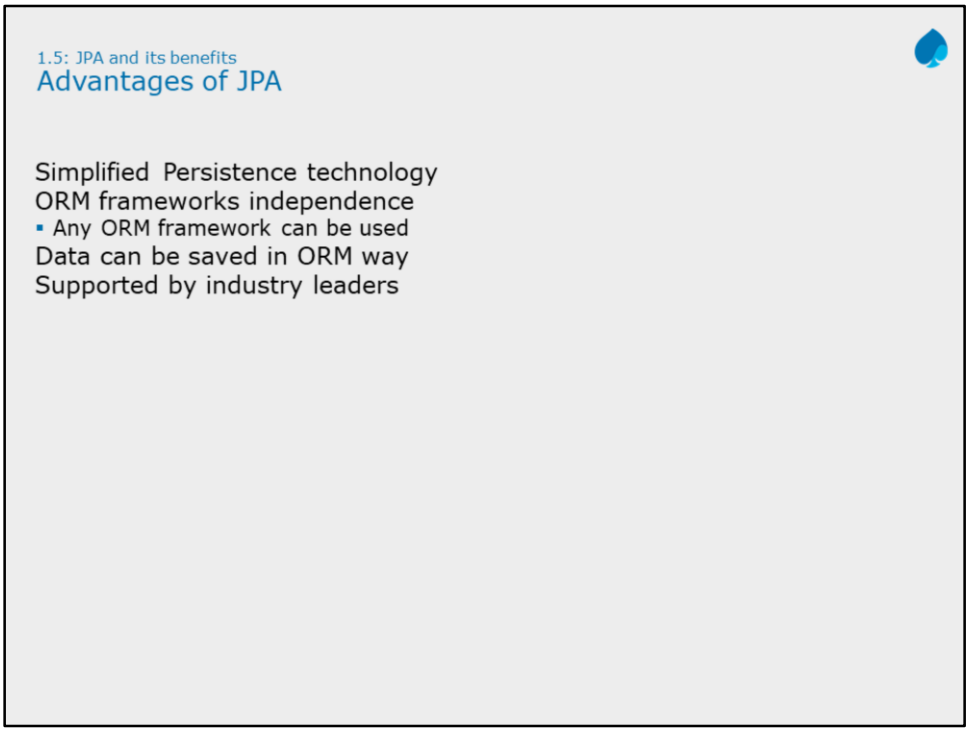
JPA is not the first attempt to create an ORM solution in Java. Before JPA, there were Java Data Objects (JDO) and Enterprise JavaBeans (EJB). JDO used to be popular, but seems to have run out of steam.

EJB, up to version 2.1, was overly complex and hard to use, harder than losing weight. EJB 3.0 simplifies things a lot and even uses JPA as its persistence mechanism. In short, JPA has started as part of EJB 3.0. However, since people want to use JPA without an EJB container, JPA has become an independent specification.

JPA is merely a specification, i.e. a document. In order for it to be useful, it needs a reference implementation, which is a Java API that implements the specification. There are numerous software packages that are JPA reference implementations. Hibernate, EclipseLink, and Apache OpenJPA are some of them.

Instructor Notes:

Add instructor notes here.



Below listed are few advantages of JPA:

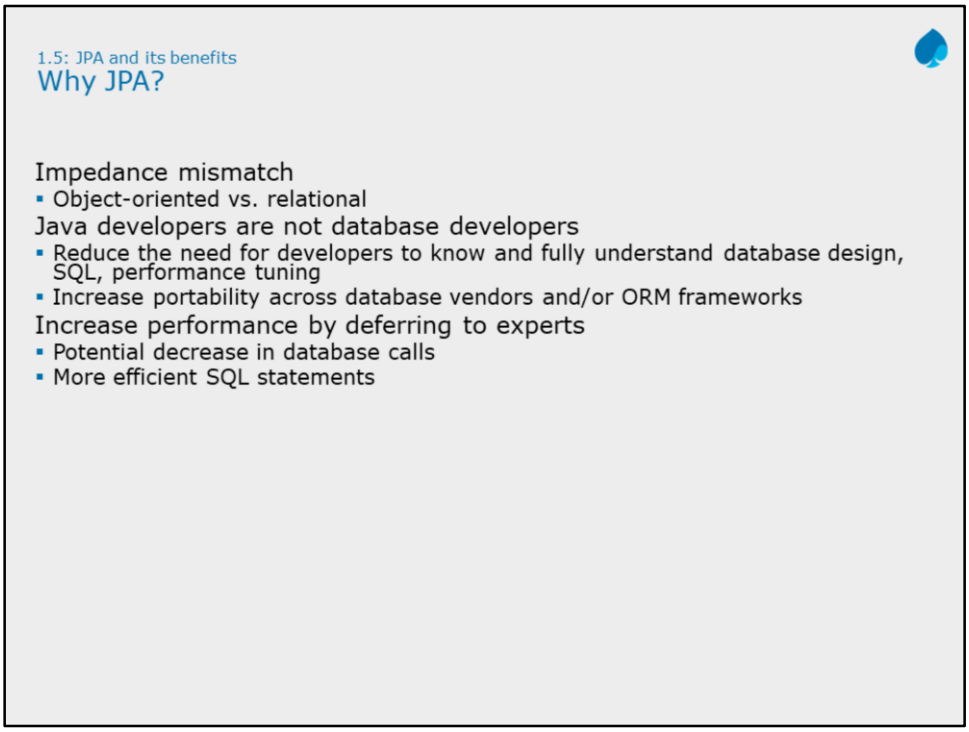
1. You don't need to create tables. In some cases, you don't even need to create a database. If any of your entity classes changes, the modern JPA provider can be configured to adapt the tables.
2. You don't need to write SQL statements, even though sometimes you may have to work with JPQL, the Java Persistence Query Language.
3. Changing databases, say from Oracle to MySQL, is a breeze

There are disadvantages too, but most of them are negligible.

1. JPA adds to the application's memory usage. Negligible in most cases.
2. JPA adds an extra layer to the application, making the system a bit slower than if it accesses the database through JDBC directly. However, the performance penalty is small that it is considered negligible.

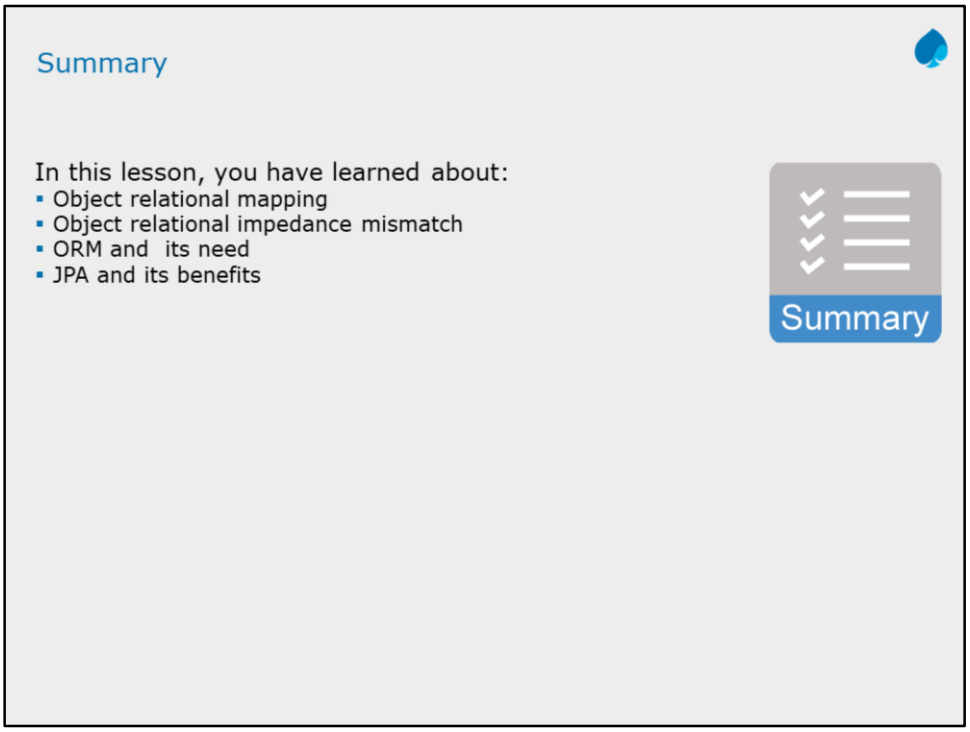
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.



Instructor Notes:

Answers:

1.True

2.Option 1 and 2

Review Question

JPA provides database and ORM implementer independence.

- True/False

Which of the following are ORM implementers?

- Option 1: Hibernate
- Option 2: OpenJPA
- Option 3: JPA
- Option 4: TopLink

Which of the following condition/s indicates granularity impedance mismatch?

- Option 1: Having more classes and less tables
- Option 2: Having more tables and less classes
- Option 3: Having one table per class