Vývoj informačních systémů

Vzory: Objektově-relační struktury

2021-22

Objektově-relační chování

- Unit of Work
  - Maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems.
- Identity Map
  - Ensures that each object gets loaded only once by keeping every loaded object in a map. Looks up objects using the map when referring to them.
- Lazy Load
  - An object that doesn't contain all of the data you need but knows how to get it.

Unit of Work (kdy?)

- + Když můžeme odložit ukládání objektů do DB
- + Když se objekty mění často
- - Kritické systémy

```
class UnitOfWork...

   public void registerNew(DomainObject obj) {
      Assert.notNull("id not null", obj.getId());
      Assert.isTrue("object not dirty", !dirtyObjects.contains(obj));
      Assert.isTrue("object not removed", !removedObjects.contains(obj));
      Assert.isTrue("object not already registered new", !newObjects.contains(obj));
      newObjects.add(obj);
   }
   public void registerDirty(DomainObject obj) {
      Assert.notNull("id not null", obj.getId());
      Assert.isTrue("object not removed", !removedObjects.contains(obj));
      if (!dirtyObjects.contains(obj) && !newObjects.contains(obj)) {
         dirtyObjects.add(obj);
      }
   }
   public void registerRemoved(DomainObject obj) {
      Assert.notNull("id not null", obj.getId());
      if (newObjects.remove(obj)) return;
      dirtyObjects.remove(obj);
      if (!removedObjects.contains(obj)) {
         removedObjects.add(obj);
      }
   }
```

Identity Map (kdy?)

- + Když nechceme aby dva objekty nereprezentovaly stejný záznam v DB

- - Když objekty nemění svůj stav

```
private Map people = new HashMap();
public static void addPerson(Person arg) {
    soleInstance.people.put(arg.getID(), arg);
}
public static Person getPerson(Long key) {
    return (Person) soleInstance.people.get(key);
}
public static Person getPerson(long key) {
    return getPerson(new Long(key));
}
```

Lazy Load (kdy?)

- + Když pro získání objektu z DB potřebujeme extra volání
- - Když nejsou objekty ve složitých kompozicích

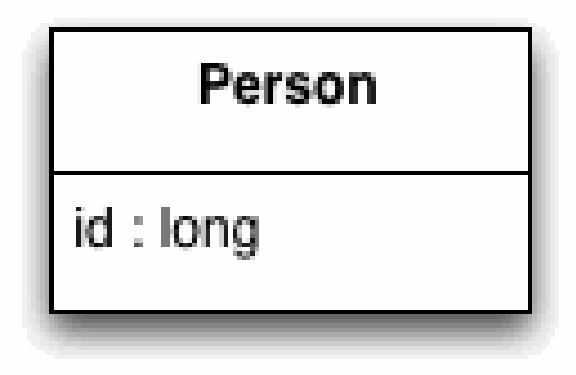
```
class Supplier...

   public List getProducts() {
      if (products == null) products = Product.findForSupplier(getID());
      return products;
   }
```

Objektově-relační struktury

- Identity Field
- Foreign Key Mapping
- Association Table Mapping
- Dependent Mapping
- Embedded Value
- Serialized LOB

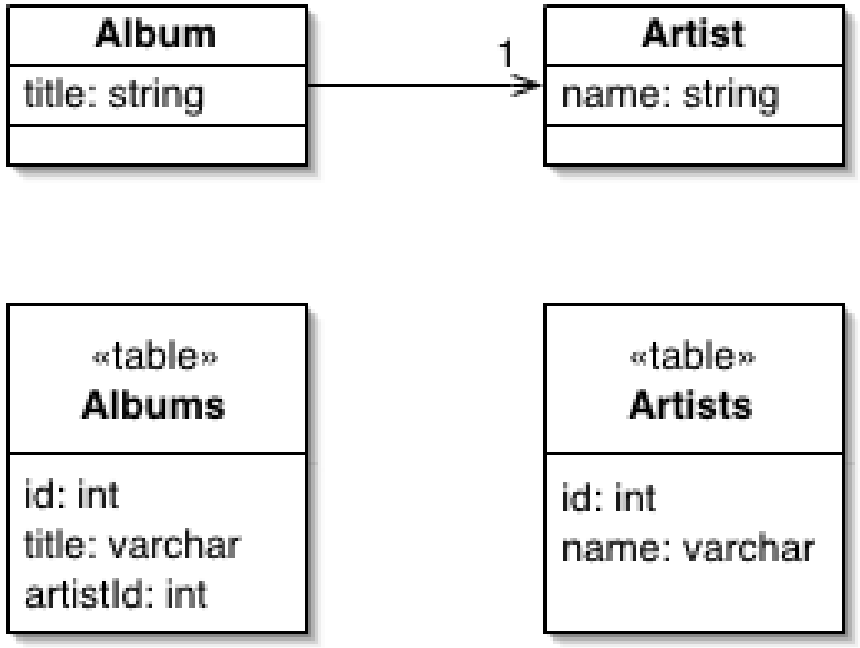
Identity Field

- Saves a database ID field in an object to maintain identity between an in-memory object and a database row.



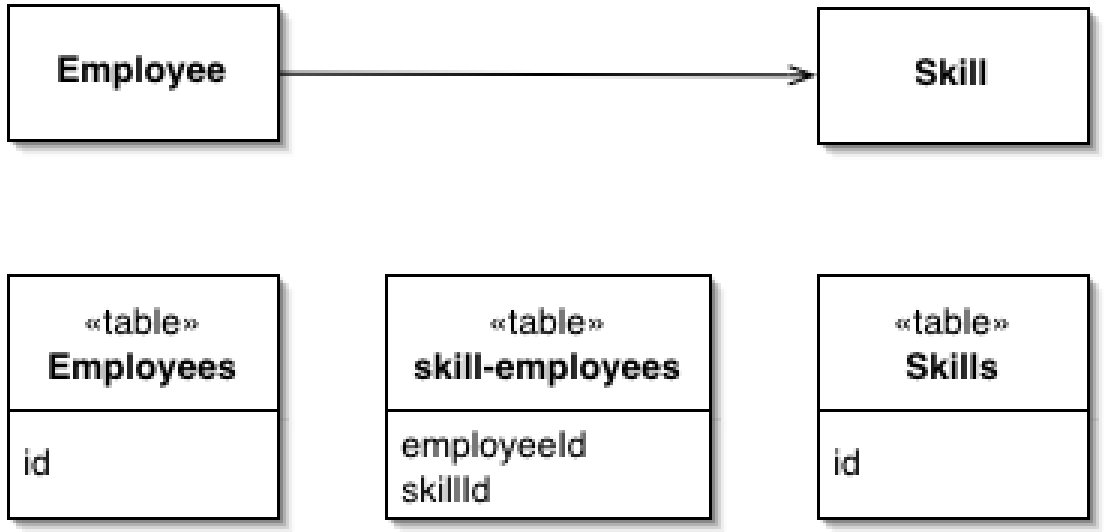
Foreign Key Mapping

- Maps an association between objects to a foreign key reference between tables.



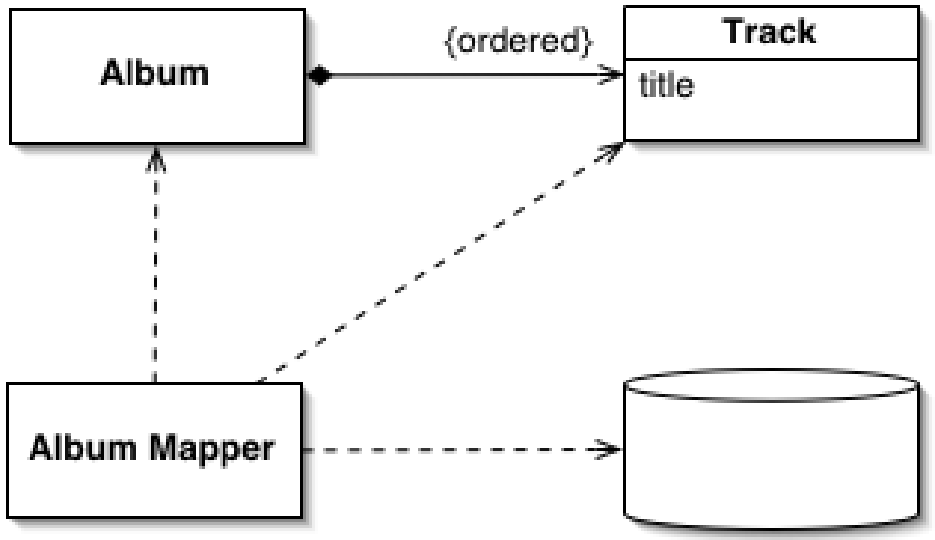
Association Table Mapping

- Saves an association as a table with foreign keys to the tables that are linked by the association.



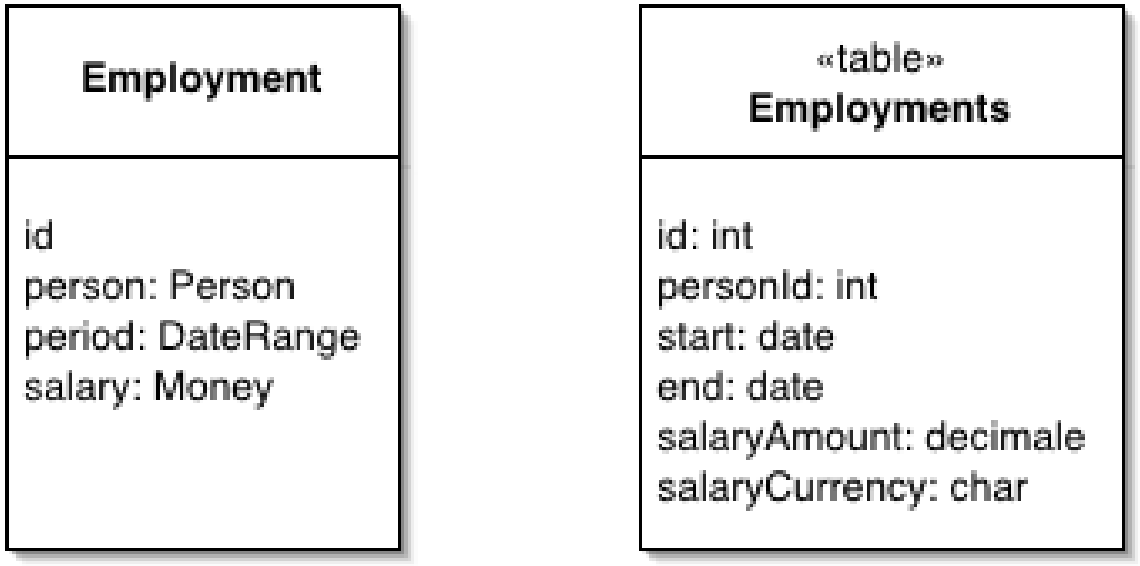
Dependent Mapping

- Has one class perform the database mapping for a child class.



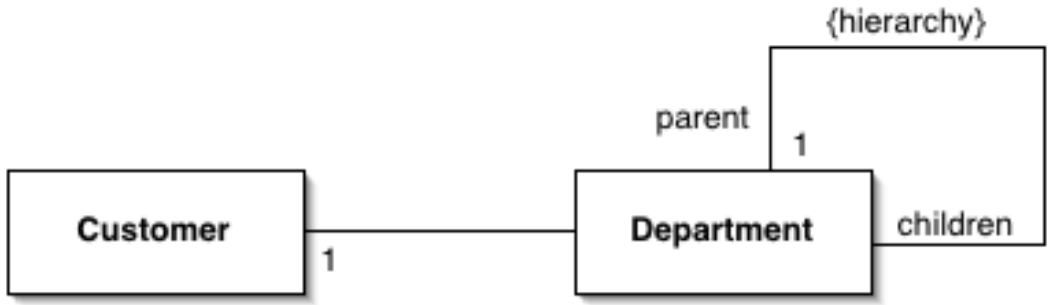
Embedded Value

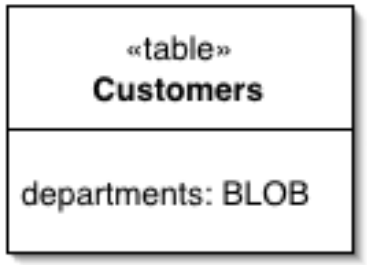
- Maps an object into several fields of another object's table.



Serialized LOB

- Saves a graph of objects by serializing them into a single large object (LOB), which it stores in a database field.





Návrh doménového modelu

- Statický diagram tříd (data + metody), typy asociací, dědičnost.
- Použité vzory.
- Možno v diagramech oddělit čistě doménový návrh od návrhu se vzory.
- Sekvenční diagramy pro klíčové operace (zejména kooperace objektů v rámci vzorů).

Úkoly na cvičení

- Prezentace připravených skic uživatelského rozhraní.
- Diskuze o modelu domény se zaměřením na použité vzory a interakci objektů v rámci vzorů.
- Implementace vzorů objektově relačního mapování pro semestrální úkol.

Kontrolní otázky

1. V jakých situacích byste použili vzor *Identity field* a proč? A kdy naopak ne? Napište fragment kódu, ze kterého bude patrné, proč jste tento vzor využili.

2. Jaký je rozdíl mezi vzory *Foreign key mapping* a *Associate table mapping*? Napište dva fragmenty kódu, ze kterých bude patrné, že jste použili tyto vzory.

3. V čem se liší vzor *Dependent mapping* od vzoru *Data mapper*? Kdy je vhodné jej použít?

4. Napište fragment kódu, ze kterého bude patrné, že jste použili vzor *Dependent mapping*.

5. Vymyslete alespoň dva příklady vhodné pro použití vzoru *Embedded value*. Co je podstatou tohoto vzoru? Proč a kdy je vhodné jej použít?

6. Vymyslete alespoň dva příklady vhodné pro použití vzoru *Serialized LOB*. Co je podstatou tohoto vzoru? Proč a kdy je vhodné jej použít?

K přečtení…

- Martin Fowler. *Patterns of Enterprise Application Architecture.* Addison-Wesley Professional, 2003 [215-277].