Applying inheritance strategies

In UML world, you can model classes with similar characteristics with a generalization hierarchy, which groups the common attributes and behaviors into a class known as the superclass, leaving the distinctions in different subclasses that inhertis the features of the superclass. Unlike UML, ERD, as a language for designing relational mapping, has no direct way of representing a generalization hierarchy. In order for an object model to map with and conform to a data model upon synchronization, inheritance strategy has to be chosen to define the way how entities should be created and structured to represent the generalization hierarchy modeled in object model. In Visual Paradigm, there are three strategies you can choose. Your selection will affect the way how entities are created and structured, ultimately affecting the way how data will be stored in database.

Choosing an inheritance strategy

1. Right click on the superclass within a generalization hierarchy and select **Open Specification...** from the popup menu.

2. Open the **ORM Class Detail** tab.
3. Click **Subclasses**....

4. Select the sub-classes. Press **Shift/Ctrl** to perform a multi selection.
5. Select the **Inheritance Strategy**.

6. Click **Apply**.
7. Click **Close**.
8. Click **OK**.

Note: Different inheritance strategies can be applied to different subclasses within a generalization hierarchy in Java project. Applying multiple strategies to different subclasses within a generalization in .NET project will result in error when the generation of code and database.

When you synchronize class diagram to ERD, you will see the entity(ies) created or re-created to respect the inheritance strategy you chose.

Strategy 1 - Table per class hierarchy

With this strategy applied, a single entity will be created for all classes within a hierarchy. Attributes from superclass and all subclasses will become columns of that entity. In order to differenetiate the type of database records in runtime, an extra discriminator column is used to store a value for identification.

Strategy 2 - Table per subclass

With this strategy applied, separate entities will be created for each subclass. A foreign key is included in each "sub-entity" for referencing the "super-entity".

Strategy 3 - Table per concrete class

With this strategy applied, separate entities will be created for each class within the generalization hierarchy. Each entity contains columns that are produced by the corresponding class plus its superclass(es) along the hierarchy.