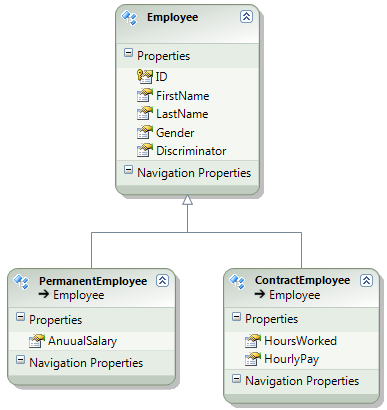
**Table Per Hierarchy (TPH) Inheritance in Entity Framework**

Inheritance in entity framework provides a way to create the required logical entities to act on a set of database tables and also to create a more meaningful relationship between entities using inheritance.

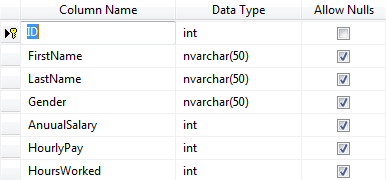
There are three different approaches to representing an inheritance hierarchy:

* **Table per Hierarchy (TPH)**: Enable polymorphism by de-normalizing the SQL schema, and utilize a type discriminator column that holds type information.
* **Table per Type (TPT)**: Represent "is a" (inheritance) relationships as "has a" (foreign key) relationships.
* **Table per Concrete class (TPC)**: Discard polymorphism and inheritance relationships completely from the SQL schema.

In this article we would see the usage of **Table per Hierarchy inheritance** in Entity Framework. In TPH inheritance one database table is used to store data for all of the entity types in the inheritance hierarchy.

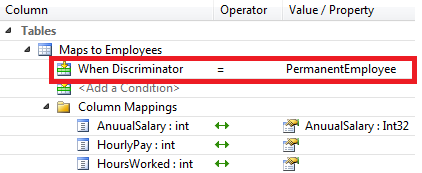


In the graph above you can see that employee entity is divided based on their types even though in the database there is a single table called Employees



To create such divided entities from single table follow the steps below:

1. Open Entity Framework designer, Right click on the designer surface and select Add - Entity option, and select base type as employee and click OK.
2. Cut and paste entity specific properties from employee entity and paste in respective sub entities.
3. Right click on PermanentEmployee entity and select Table Mapping option, and map PermanentEmployee entity to Employees table. Also notice the conditional mapping, in which we are using Discriminator column to determine when an Employee can be permanentEmployee.



1. Since we are using Discriminator column in conditional mapping we cannot use it again with property mapping in Employee entity. So delete it from there.
2. Finally right click on Employee entity and select properties. In the properties window set Abstract=true. This should make Employee class an abstract class.

That’s it, your divided entities are now mapping to the same database table. Now you can query this table and work with separated employee types using these entities.

Attached is the sample application that demonstrates usage of TPH using Entity Framework 6

EF, Entity Framework 6, Entity Framework,TPH,Table per Hierarchy