# ORM (Object-Relational Mapping)

## Définition

ORM signifie Object-Relational Mapping. Un ORM est un ensemble de classes permettant de manipuler les tables d’une base de données relationnelle comme s’il s’agissait d’objets.

Un ORM est une couche d’abstraction d’accès à la base de données qui donne l’illusion de ne plus travailler avec des requêtes mais de manipuler des objets.

L’avantage de cette couche d’abstraction est qu’il n’y a plus besoin de se soucier du système de base de données utilisé, c’est l’ORM qui a la charge de transformer les requêtes pour les rendre compatibles avec la base de données.

L’ORM va s’appuyer sur des modèles. Un modèle représente une table de la base de données.

## Des solutions ORM

## Linq to SQL

**Avantages**

* Intégré au framework .NET et au designer de Visual Studio
* Customisation (procédures stockées, lazy load…)
* Simplicité d’utilisation (Linq…)
* Rapidité de mise en œuvre
* Performances

**Inconvénients**

* Limité à SqlServer
* Mapping limité 1 table = 1 classe
* Nécessite .NET 3.5+

### Entity Framework

**Avantages**

* Solution privilégiée par MS, sorte de Linq To Sql v2.0
* Simplicité d’utilisation
* Simplicité de mise en œuvre
* Customisation très poussée : lazy load, héritage de tables, développement « class driven » ou « data driven »

**Inconvénients**

* Performances légèrement en deça de Linq To Sql
* Nécessite .NET 3.5+
* Complexité parfois masquée par une interface simple (comme souvent avec les outils Microsoft)

### Nhibernate (solution tierce)

**Avantages**

* Maturité
* Robustesse
* Fort degré de customisation

**Inconvénients**

* Paramétrage fastidieux (XML)
* Performances
* Difficulté au débuggage
* Courbe d’apprentissage très forte
* Pas de support de Linq
* Dépendance à un projet open-source