ORM

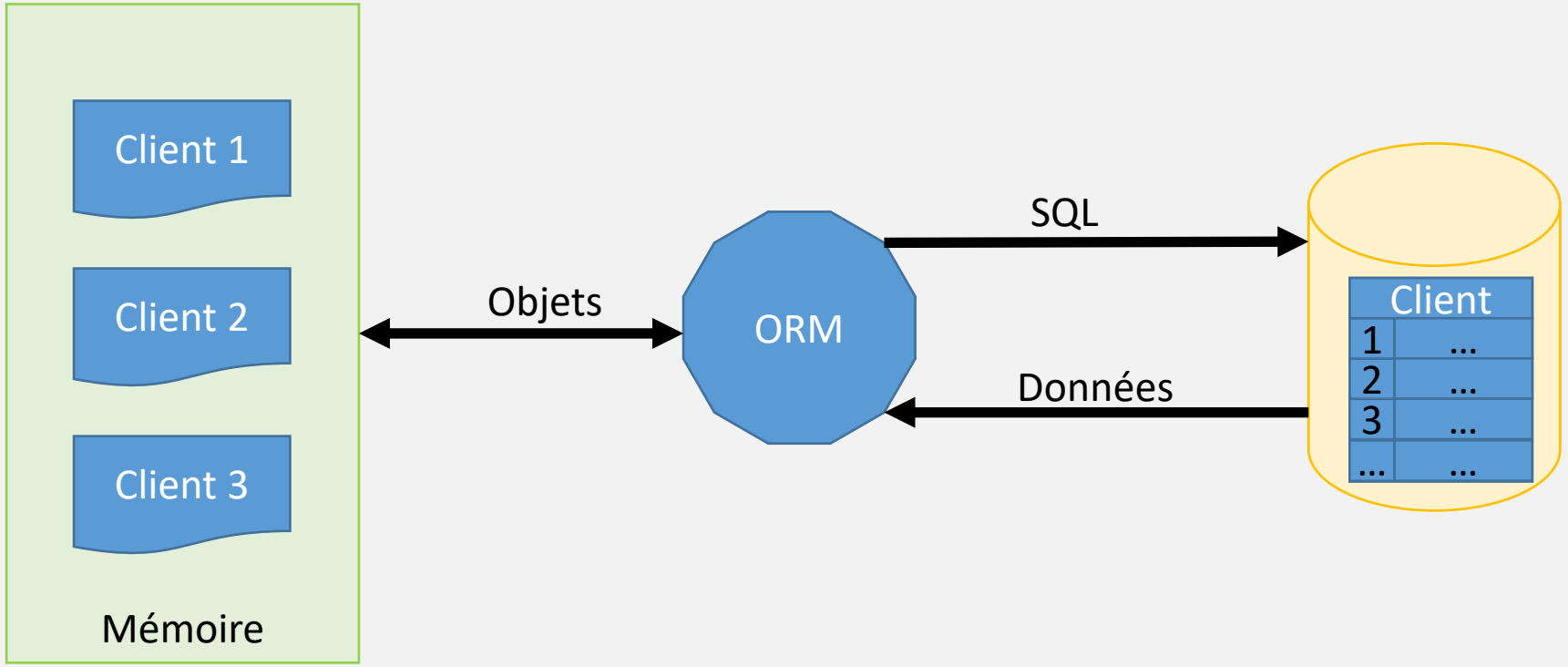
Photo : https://unsplash.com

Objectifs

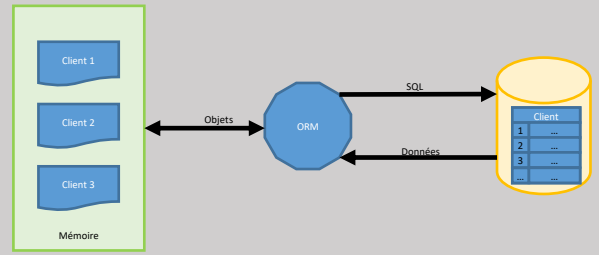
- Généralités sur les ORMs
- Entity Framework

Généralités – ORM (Object Relationnal Mapping)

- Fait le lien entre les classes de votre code et des tables
- Chaque objet correspond à un enregistrement
- Il s'occupe de convertir les actions du développeur en requêtes SQL

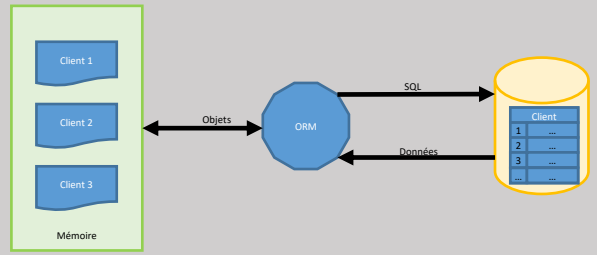


Entity Framework



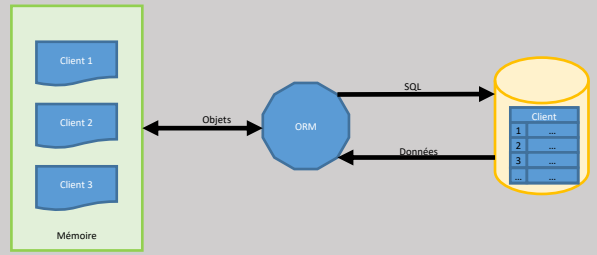
- ORM open source de Microsoft
- Permet d'interragir avec plusieurs SGBD (SQL Server, MySQL/MariaDB, Oracle, SQLLite, etc.) avec sensiblement le même code
- Fonctionne par conventions :
  - Le nom de la classe correspond au nom de la table (Ex. : classe Client <-> table client)
  - Chaque nom de propriété C# (get; set;) correspond au nom d'une colonne dans la table
  - Le nom de la clef primaire est « <nomClasse>Id » (Ex. : ClientId)
  - Les noms des clefs étrangères peuvent être :
    - <nomPropriété>
- Ces conventions peuvent être remplacées par des annotations (Attributs en C#) sur les classes et propriétés
- On peut aussi utiliser une approche de description dite « fluent »
  - Nous utiliserons les attributs

Entity Framework



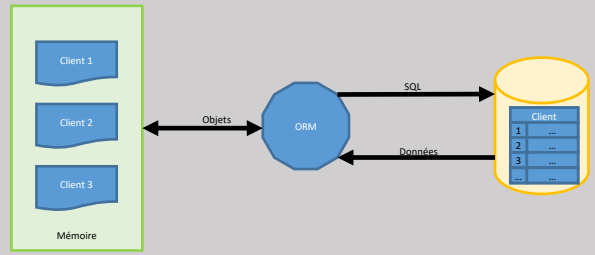
- Il existe deux approches
  - Code First : la base de données est créée par EF à partir de ce qui a été configuré
  - Database First : il faut créer la configuration pour se coller à la base de données
- Nous allons utiliser l'approche DB First
- EF utilise massivement Linq pour les reqêtes

Entity Framework



- Une classe correspond à la description d'un type d'enregistrement
- Chaque propriété correspond à une colonne
- Les annotations permettent de définir des contraintes unitaires sur les colonnes ou de configurer EF. Exemple :
  - Key : Identifiant unique de l'objet
  - DataType : Type de la donnée (Exemple date)
  - Required : Le champ devient obligatoire
- Ces informations sont décrites dans l'espace de nommage : « System.ComponentModel.DataAnnotations » (https://msdn.microsoft.com/fr-fr/library/system.componentmodel.dataannotations(v=vs.110).aspx)
- Le contexte de base de données fait le lien entre une classe et une table
- Le contexte permet aussi de se connecter à la base de données

Entity Framework



- Généralement :
  - Il faut créer une classe par table et une propriété par champ
  - Il faut créer une classe qui hérite de la classe « DbContext » afin de configurer l'accès vers la base de données :
    - Il faut spécifier le fournisseur spécifique à la base de données utilisé avec Use<NomSGBD> (Ex. : UseSqlServer, UseOrable, UserMySQL)
  - **Une instance de DbContext est à créer par unité de travail, c'est-à-dire par tâche / transaction**
  - Il est conseillé d'utiliser une approche de type « Dépôt » afin de protéger votre code (Attention : c'est un anti-pattern mais nous allons quand même l'utiliser comme cela ici)
- Pour Oracle :
  - Package nuget « Oracle.EntityFrameworkCore »
  - Exemple de chaine de connexion :
    - `"Data source=localhost:1521/orcl; User Id=pfleon; Password=Bonjour01.+;"`

Création des entités - exemple

```
public class Personne {
    public int PersonneId { get; set; }

    [Display(Name = "Le nom")]
    [Required()]
    public string Nom { get; set; }

    [Required()]
    public string Prenom { get; set; }

    public int Age { get; set; }
}
```

Description d'un modèle de données - exemple

```
public class MaBDContext : DbContext {
    public MaBDContext(DbContextOptions<MaBDContext> options)
        : base(options) { }

    public DbSet<Personne> Personnes { get; set; }
}
```

Exemple de configuration

Pour accéder au dépôt de données, il faut enregistrer une chaine de connexion.
Dans le cadre du cours, nous allons utiliser une base de données de type SQL Server
Pour réaliser la configuration du dépôt de données « MaBaseDeDonnees », ouvrir le fichier appsettings.json et ajouter la chaine suivante :

```
  "ConnectionStrings": {
    "DefaultConnection": "Data source=localhost:1521/orcl; User Id=pfleon; Password=Bonjour01."
  },
```

En .Net MVC, dans la méthode « ConfigureServices » de la classe « Startup », ajouter la chaine de connexion suivante et l'enregistrer :

```
services.AddDbContext<MaBDContext>(
    options =>
                options.UseOracle(Configuration.GetConnectionString("DefaultConnection")));
```

Références

- Entity Framework core : https://docs.microsoft.com/en-us/ef/core/
- Configuration du contexte : https://docs.microsoft.com/en-us/ef/core/dbcontext-configuration/
- Fournisseurs BD :
  - https://docs.microsoft.com/en-us/ef/core/dbcontext-configuration/#dbcontextoptions : Liste incomplète mais plus détaillée
  - https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli
- Optimisation mémoire - Context pooling (regardez « Without dependency injection pour ce cours car on n'utilise pas ASP.Net Core app) : https://docs.microsoft.com/en-us/ef/core/performance/advanced-performance-topics