**ORM (Object-Relational Mapping)**

**Un ORM es una herramienta de software que actúa como un puente entre un lenguaje de programación orientado a objetos y una base de datos relacional.  
Su propósito es permitir que los desarrolladores interactúen con la base de datos usando objetos y clases, sin escribir SQL manualmente, mediante un mapeo automático y transparente entre objetos y tablas.**

**🔑 Conceptos Fundamentales**

* **Mapeo: Relaciona clases, atributos y relaciones del lenguaje orientado a objetos con tablas, columnas y claves foráneas de la base de datos.**
* **Abstracción: Proporciona una capa que oculta los detalles del SQL, permitiendo realizar operaciones CRUD con objetos.**

**⚠️ Problema del "Impedance Mismatch"**

**Se refiere a las diferencias fundamentales entre el modelo relacional y el modelo orientado a objetos, como:**

1. **Estructura: Las clases pueden tener jerarquías y métodos; SQL no tiene equivalentes para estas estructuras.**
2. **Instancia vs. Fila: Un objeto puede tener un estado complejo, mientras que una fila es una representación tabular de datos.**
3. **Encapsulamiento: Los objetos controlan el acceso a sus datos; las filas pueden ser modificadas directamente.**
4. **Identidad: En OOP, la identidad no depende del contenido; en SQL, depende de la clave primaria.**
5. **Modelo de procesamiento: OOP es imperativo y basado en navegación; SQL es declarativo y basado en conjuntos.**
6. **Propiedad: El esquema de base de datos puede ser gestionado por otro equipo, lo que dificulta los cambios sincronizados.**

**✅ Ventajas del uso de ORM**

* **Desarrollo más rápido: Reduce el código repetitivo.**
* **Abstracción del motor de base de datos: Independencia del dialecto SQL.**
* **Menor riesgo de inyección SQL.**
* **Mantenibilidad: Código más limpio y desacoplado.**
* **Funcionalidades avanzadas: Lazy loading, eager loading, cacheo, migraciones, manejo de transacciones, etc.**

**❌ Desventajas del ORM**

* **Curva de aprendizaje.**
* **Rendimiento: Menor que el SQL manual optimizado.**
* **Complejidad oculta: Puede dificultar la depuración.**
* **Limitaciones para consultas muy específicas o complejas.**
* **Sobreuso de recursos (memoria, CPU).**
* **Dependencia del ORM: A largo plazo puede dificultar cambios.**
* **"Leaky abstraction": A veces es necesario conocer SQL de todos modos.**

**🔧 Data Access Object (DAO)**

**Es un patrón de diseño que proporciona una interfaz para acceder a los datos, desacoplando la lógica de negocio del mecanismo de persistencia.**

**Ventajas del DAO:**

* **Aísla el acceso a la base de datos.**
* **Centraliza las operaciones CRUD.**
* **Facilita la migración entre motores de base de datos.**

**🐍 Introducción a SQLAlchemy (Python)**

**SQLAlchemy es uno de los frameworks ORM más potentes y flexibles en Python. Proporciona:**

**Componentes:**

1. **SQLAlchemy Core**
   * **Permite construir consultas SQL mediante expresiones Python.**
   * **Incluye abstracción de esquemas, tipos de datos y pooling de conexiones.**
2. **SQLAlchemy ORM**
   * **Proporciona mapeo objeto-relacional.**
   * **Implementa patrones como *Unit of Work* e *Identity Map*.**

**Esta arquitectura dual permite trabajar tanto a bajo nivel (SQL programático) como a alto nivel (modelo orientado a objetos), adaptándose a distintos estilos de desarrollo.**