AD_T4_ACT1: ORM

- ¿Qué es el Mapeo Objeto- Relacional?

El Mapeo Objeto-Relacional (ORM, por sus siglas en inglés Object-Relational Mapping) es una técnica de programación que permite la interacción entre sistemas de gestión de bases de datos relacionales y programas orientados a objetos. En un entorno de desarrollo de software, a menudo se utiliza un enfoque orientado a objetos para modelar y organizar la lógica del programa, mientras que las bases de datos relacionales se utilizan para almacenar y recuperar datos de manera estructurada.

- ¿Por qué usar un ORM? ¿Qué ocurre cuando no se usa un ORM en el desarrollo de una aplicación que accede a una base de datos?

El motivo de usar ORM es porque tiene ventajas en el desarrollo de aplicaciones, como:

1. **Abstracción de la base de datos:** Un ORM proporciona una capa de abstracción que separa la lógica de la aplicación de los detalles específicos de la base de datos. Los desarrolladores pueden trabajar con objetos y clases en su código sin tener que preocuparse por la sintaxis y particularidades de SQL.
2. **Productividad:** Al eliminar la necesidad de escribir consultas SQL manualmente y gestionar la traducción entre objetos y tablas de base de datos, los desarrolladores pueden ser más productivos y centrarse en la lógica de la aplicación en lugar de en detalles de implementación de la base de datos.
3. **Portabilidad:** Los ORM a menudo son compatibles con varios sistemas de gestión de bases de datos, lo que facilita la portabilidad del código entre diferentes plataformas de bases de datos. Esto significa que puedes cambiar de una base de datos a otra sin tener que reescribir gran parte de tu código.
4. **Facilita la persistencia de objetos:** Los ORM simplifican la tarea de almacenar y recuperar objetos desde una base de datos relacional. Esto se traduce en un código más limpio y mantenible, ya que los desarrolladores no tienen que escribir código personalizado para gestionar la persistencia.
5. **Manejo automático de relaciones:** Los ORM pueden gestionar automáticamente las relaciones entre las entidades del modelo de datos, como las asociaciones uno a uno, uno a muchos y muchos a muchos. Esto elimina la necesidad de escribir manualmente código para manejar la lógica de relaciones en la base de datos.
6. **Reducción de código repetitivo:** Al usar un ORM, los desarrolladores pueden evitar escribir repetitivamente código SQL para las operaciones CRUD (Crear, Leer, Actualizar, Eliminar), ya que estas operaciones son manejadas por el ORM de manera transparente.

7. **Seguridad:** Los ORM suelen proporcionar mecanismos para prevenir ataques de inyección SQL, ya que las consultas SQL se generan de manera automática y segura, evitando la concatenación de cadenas y la exposición a vulnerabilidades.
8. **Modelado de datos más natural:** Al trabajar con objetos en lugar de tablas, el código puede reflejar de manera más natural la estructura del dominio de la aplicación, lo que hace que el desarrollo y el mantenimiento del código sean más intuitivos.

Mientras que cuando no se utiliza un Mapeo Objeto-Relacional (ORM) en el desarrollo puede surgir estos obstaculos:

1. Los desarrolladores deben escribir manualmente consultas SQL para interactuar con la base de datos.
2. El código puede volverse más complejo y propenso a errores, especialmente en operaciones CRUD y consultas avanzadas.
3. Mantener y evolucionar la aplicación puede ser más difícil, ya que cambios en la base de datos requieren modificaciones manuales en el código.
4. La portabilidad del código entre diferentes sistemas de gestión de bases de datos puede ser un desafío.
5. La representación de datos en el código puede no seguir un enfoque orientado a objetos.
6. La seguridad puede verse comprometida si no se protege adecuadamente contra ataques de inyección SQL.
7. El desarrollo puede ser más lento y las pruebas pueden ser más complejas debido a la interacción directa con la base de datos.

• Ventajas e inconvenientes de utilizar ORM

Ventajas de utilizar ORM:

1. **Abstracción de la Base de Datos:**
    - Los desarrolladores trabajan con objetos en lugar de consultas SQL, lo que proporciona una abstracción más cercana al paradigma orientado a objetos.
2. **Productividad Mejorada:**
    - Facilita el desarrollo al eliminar la necesidad de escribir consultas SQL manualmente, permitiendo a los desarrolladores centrarse en la lógica de la aplicación.
3. **Portabilidad del Código:**
    - La capa ORM facilita la portabilidad del código entre diferentes sistemas de gestión de bases de datos, ya que la lógica de acceso a la base de datos está encapsulada.
4. **Mantenimiento Sencillo:**

- Simplifica el mantenimiento, ya que los cambios en el esquema de la base de datos se manejan a través de configuraciones en lugar de modificaciones manuales extensas en el código.

5. **Seguridad:**
   - Los ORM a menudo ofrecen protección contra ataques de inyección SQL, ya que utilizan consultas parametrizadas o mecanismos internos para prevenir este tipo de vulnerabilidades.
6. **Operaciones CRUD Simplificadas:**
   - Facilita las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) al proporcionar métodos orientados a objetos para interactuar con la base de datos.
7. **Modelado de Datos en Objetos:**
   - Permite a los desarrolladores modelar datos de manera más natural en el código, reflejando las relaciones y estructuras de objetos en lugar de tablas de base de datos.

Inconvenientes de utilizar ORM:

1. **Rendimiento:**
   - En algunas situaciones, el rendimiento puede ser menor en comparación con consultas SQL manuales, especialmente en aplicaciones que requieren operaciones de base de datos extremadamente eficientes.
2. **Aprendizaje y Curva de Adopción:**
   - Puede haber una curva de aprendizaje para entender y utilizar eficazmente un ORM, especialmente para desarrolladores que no están familiarizados con sus conceptos y configuraciones.
3. **Optimizaciones Específicas del Motor de Base de Datos:**
   - Algunas optimizaciones específicas del motor de base de datos pueden no ser accesibles o explotadas completamente a través del ORM, lo que puede afectar el rendimiento en casos específicos.
4. **Complejidad Adicional:**
   - En casos de modelos de datos complejos, el ORM puede introducir cierta complejidad, especialmente al manejar relaciones avanzadas o consultas complejas.
5. **Personalización Limitada:**
   - En algunas situaciones, los ORM pueden tener limitaciones en la personalización y ajuste fino, lo que podría ser un problema en proyectos que requieren un control preciso sobre el acceso a la base de datos.

- Herramientas actuales de ORM

En la actualidad hay varias herramientas ORM como:
- Hibernate.
- Entity Framework.
- SQLAlchemy.
- Active Record (Ruby on Rails).
- Spring Data JPA.
- Sequelize.
- Rails ActiveRecord (Ruby on Rails).