TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE:

- ¿Qué es ORM?
- ¿Qué es un modelo?
- ¿Qué son las relaciones?

¿QUÉ ES UN ORM?

Son las siglas de Object-Relational Mapping (ORM), que es una técnica de programación para convertir datos entre bases de datos relacionales, y lenguajes de programación. Es decir, corresponde a un modelo que permite mapear las estructuras de una base de datos relacional, sobre una estructura lógica de entidades, con el objetivo de simplificar y acelerar el desarrollo de nuestras aplicaciones.

Para facilitar la programación, y reducir la posibilidad de cometer errores, muchos desarrolladores prefieren no ejecutar directamente las sentencias SQL, sino construir un modelo de objetos que refleje la estructura de datos. En tiempo de ejecución, los datos serán recuperados de la base de dato, y llenados en el modelo de objetos. Entonces, los desarrolladores pueden trabajar completamente con los objetos, sin necesidad de escribir una sentencia SQL.

Las estructuras de la base de datos relacional quedan vinculadas con las entidades lógicas, o base de datos virtual, definida en el ORM. De este modo, las acciones CRUD (Create, Read, Update, Delete) a ejecutar sobre la base de datos física, se realizan de forma indirecta por medio del ORM.

Así, los objetos o entidades de la base de datos virtual creada en nuestro ORM, podrán ser manipulados por medio de algún lenguaje de nuestro interés, según el tipo de ORM utilizado. Por ejemplo: **Sequelize** para Nodejs. La interacción con el manejador de base de datos quedará delegada en los métodos de actualización correspondientes, proporcionados por el ORM. Los ORMs más completos ofrecen servicios para persistir todos los cambios en los estados de las entidades, previo seguimiento o tracking automático, sin tener que escribir una sola línea de SQL.

¿POR QUÉ UTILIZARLO?

Las soluciones de ORM son útiles para facilitar el desarrollo de APIs basadas en datos. Los usuarios tienen necesidades concretas, las cuales impulsan el modelo de datos de una aplicación. En el desarrollo, esta arquitectura de datos se implementa típicamente, y se controla la versión utilizando scripts de base de datos, como los SQL. A continuación, se emplea una biblioteca independiente para que la aplicación del servidor ejecute acciones CRUD en la base de datos.

Los ORM funcionan como una API de alto nivel para ejecutar CRUD, y actualmente, los de calidad también nos permiten inicializar los datos a través del código. La manipulación de datos complejos, la limpieza, entre otros, es más fácil en el código. La implementación de extraer, transformar, y cargar con código, permite a un sistema integrar más fácilmente los datos de fuentes que son muy diferentes. Las bases de datos SQL de múltiples sabores, los datos NoSQL, los datos del sistema de archivos, y los datos de terceros, pueden integrarse en un solo lenguaje con un ORM JavaScript.

Por último, el control de datos orientado al código, también permite que un sistema implemente el uso de datos en tiempo de ejecución, o en el proceso de construcción, y que se adapte el uso de forma flexible durante el desarrollo, según sea necesario.

Para reiterar, los ORM mejoran la productividad del desarrollador al proporcionar una API de alto nivel en un solo lenguaje, con una funcionalidad que tradicionalmente requeriría varias herramientas, y conjuntos de habilidades diferentes. El hecho de que se necesiten menos habilidades, herramientas, y horas de trabajo, facilita el margen del proyecto. Los requisitos imprevistos, y la línea de tiempo del proyecto, se preparan mejor con una configuración flexible de datos de construcción y de tiempo de ejecución.

Ventajas de un ORM

- Acelera el desarrollo, pues elimina la necesidad de código SQL repetitivo.
- Reduce el tiempo de desarrollo.
- Reduce los costos de desarrollo.
- Supera las diferencias de SQL específicas de los proveedores: el ORM sabe cómo escribir SQL específicas de los proveedores para que tu no tengas que hacerlo.

- Abstrae el sistema de la base de datos, para que el cambio de MySQL a PostgreSQL, o cualquier otro sistema que prefieras, sea fácil.
- Dependiendo del ORM, obtienes numerosas características avanzadas; tales como: soporte para transacciones, conexiones, migraciones, semillas, flujos, entre otras.
- Las consultas SQL tendrán un mejor rendimiento.

Desventajas de los ORM

- Pérdida de productividad del desarrollador mientras aprende a programar con ellos.
- Los desarrolladores pueden perder la comprensión de lo que el código está haciendo realmente, y tienen más control usando SQL. Como desarrollador, es importante entender lo que está sucediendo. Dado que los ORMs pueden servir como una muleta para evitar la comprensión de las bases de datos y SQL, probablemente te generará debilidad en ese campo.
- Tienen una tendencia a ser lentos.
- No pueden competir con las consultas SQL para consultas complejas.
- Hay una sobrecarga involucrada en el aprendizaje de cómo usar cualquier ORM dado.
- Su configuración inicial puede llevar tiempo.

¿QUÉ ES UN MODELO?

Es una abstracción que representa una tabla en la base de datos. Éstos permiten representar las tablas de la base de datos, y manipular los datos asociados a ellas. Un modelo permite describir:

- El tipo de datos, y la forma en que se relacionan esos datos.
- Restricciones de integridad de los datos.
- Operaciones de manipulación de datos (CRUD).

¿QUÉ SON LAS RELACIONES?

Las relaciones de bases de datos son asociaciones entre tablas, las cuales se crean utilizando sentencias de unión para recuperar datos. Existen varios tipos: unívoca, uno a varios, varios a varios.

También existen relaciones de asociación entre modelos, éstas representan la relación entre las tablas correspondientes en la base de datos. Según la relación del modelo, se pueden realizar operaciones como: la consulta de conexión, la actualización, y la eliminación entre tablas relacionadas.