### El Desfase Objeto-Relacional

* El **desfase objeto-relacional** surge porque el mundo de los objetos en programación (Java) es diferente al modelo relacional de las bases de datos.
* En Java, trabajamos con **objetos** mientras que las bases de datos usan **tablas** y relaciones.
* Ejemplo: Las bases de datos no tienen conceptos de herencia, y manejar estructuras complejas de objetos como listas dentro de una tabla es complicado.
* **ORM (Object-Relational Mapping)**, como Hibernate, es una solución para este desfase, pero en JDBC el desarrollador gestiona esta discrepancia manualmente.
* El **desfase objeto-relacional** (también conocido como "impedance mismatch" en inglés) se refiere a las diferencias entre el paradigma de programación orientado a objetos, utilizado en lenguajes como Java, y el modelo relacional que siguen las bases de datos tradicionales. Este desfase surge cuando intentas mapear objetos de un lenguaje de programación a las estructuras de una base de datos relacional, que trabaja con tablas, filas y columnas.

### Principales causas del desfase objeto-relacional:

1. **Estructuras de datos diferentes**:
   * En la programación orientada a objetos, los datos se organizan en clases con atributos y métodos que encapsulan tanto el estado como el comportamiento.
   * En las bases de datos relacionales, los datos se organizan en tablas con filas y columnas, donde cada fila representa un registro y cada columna un atributo.
   * El desafío es cómo convertir objetos complejos en filas y columnas de una tabla relacional, ya que los objetos pueden tener herencias, relaciones de agregación o composición que no encajan directamente en el modelo relacional.
2. **Relaciones entre objetos vs relaciones entre tablas**:
   * En el modelo orientado a objetos, las relaciones entre objetos pueden ser de herencia, composición o asociación. Estas relaciones son flexibles y dinámicas.
   * En bases de datos relacionales, las relaciones se representan mediante **claves foráneas** que conectan una tabla con otra, y no soportan de manera nativa conceptos como la herencia.
3. **Persistencia de objetos**:
   * En un sistema orientado a objetos, los objetos viven en memoria y desaparecen cuando el programa finaliza.
   * En bases de datos relacionales, los datos persisten de manera duradera en disco. La necesidad de convertir (o mapear) los objetos en tablas y luego volver a reconstruirlos para usarlos en memoria es un desafío.

### Ejemplos del desfase:

* **Herencia**: Si en Java tienes una clase Persona y una clase Empleado que hereda de Persona, en el mundo relacional no hay una forma directa de representar la herencia. Debes decidir si mapeas ambas clases en una sola tabla, o si creas dos tablas con claves foráneas para conectar las entidades.
* **Colecciones**: Un atributo de una clase puede ser una colección de objetos (como una lista o un conjunto). Sin embargo, las bases de datos no tienen un tipo de dato que represente directamente una colección, por lo que debes usar varias tablas con relaciones uno-a-muchos o muchos-a-muchos.

### Soluciones para el desfase objeto-relacional:

1. **Mapeo Objeto-Relacional (ORM)**:
   * Las herramientas ORM como **Hibernate** en Java intentan resolver este desfase proporcionando una capa de abstracción entre el código orientado a objetos y la base de datos relacional. Estas herramientas generan automáticamente el SQL necesario para mapear los objetos en tablas, facilitando la persistencia y recuperación de los datos.
2. **JDBC**:
   * Aunque JDBC no es una solución ORM, permite a los desarrolladores interactuar directamente con las bases de datos mediante SQL. Con JDBC, el desarrollador es responsable de gestionar el mapeo entre objetos y tablas, lo que requiere más trabajo manual, pero ofrece mayor control sobre las consultas.

### Conclusión:

El desfase objeto-relacional es un desafío que surge de las diferencias fundamentales entre cómo los datos se manejan en la programación orientada a objetos y en las bases de datos relacionales. Aunque existen herramientas como ORM para ayudar a reducir esta brecha, siempre será importante entender las diferencias para optimizar la interacción entre ambos mundos.