

# Desarrollo de Aplicaciones Web

Tema II: Capa de Acceso de Datos

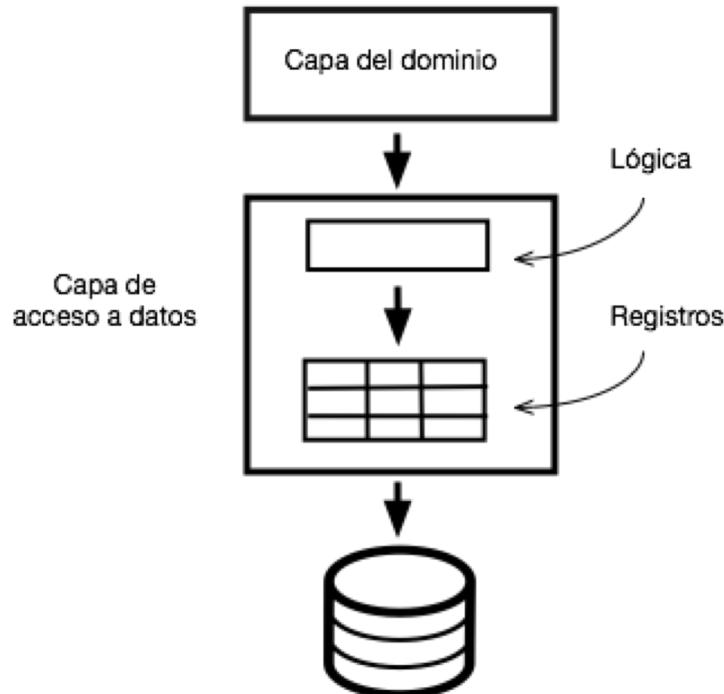
# Capa de Acceso a Datos

- Su papel es comunicarse con la infraestructura
- Permite que la aplicación use los datos
- Una parte dominante del problema es
  - Hablar solamente de BD relacionales
  - Ya que es muy generalizado el uso de BD
  - Las BD relacionales utilizan lenguaje SQL
- Existe todavía desarrolladores que no entienden bien el lenguaje SQL

# Capa de Acceso a Datos (Cont)

- Lo ideal es separar las instrucciones de SQL
  - Dejarlas aparte de la lógica del dominio
  - Colocarlas en clases separadas
- Se puede crear una clase por cada tabla
  - La clase forma una compuerta a la tabla de datos
  - Esto facilita el acceso a la información
  - El resto de la app no necesita conocer SQL
  - Todas las instrucciones SQL se encuentran fácilmente

# Capa de Acceso a Datos (Cont)



# Compuerta de Tabla de Datos

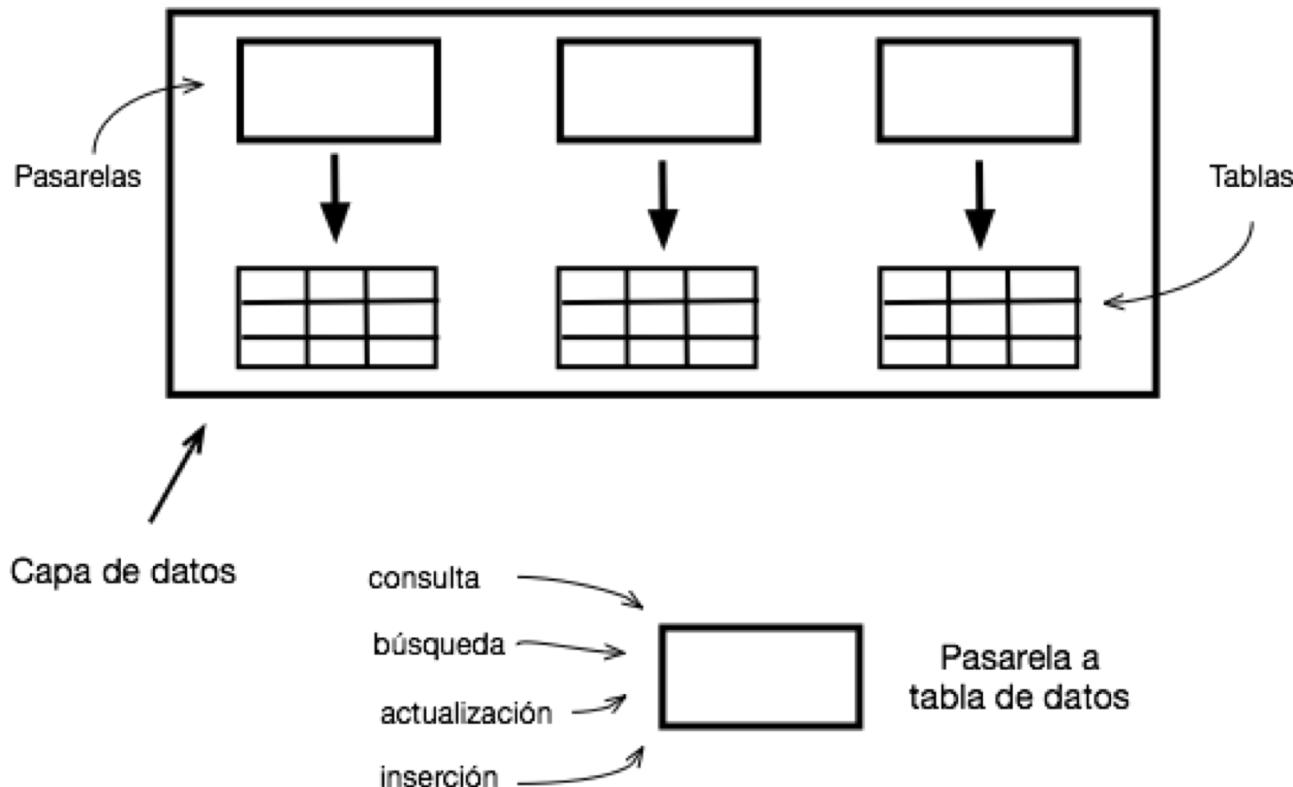
# Capa de Acceso de Datos (Cont)

- Se puede organizar en
  - Compuerta a tabla de datos
  - Compuerta a Fila de datos
  - Registro Activo
  - Mapeo de datos

# Compuerta a Tabla de Datos

- Se conoce como Table Data Gateway (TDG)
- La idea es no mezclar SQL con la lógica del APP
- El TDG mantiene todas las instrucciones SQL
  - Estas acceden tablas y/o vistas
  - Incluye: **select, insert, update, delete**
- Otros objetos hacen llamados al TDG
- El TDG maneja la interacción con la BD

# Compuerta a Tabla de Datos



# Compuerta a Tabla de Datos (Cont)

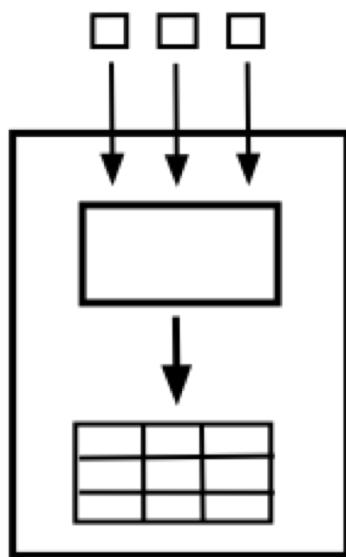
- El TDG tiene una interfaz sencilla
- Consiste de muchos métodos
  - Búsqueda para recuperar datos desde la BD
  - Actualización de los datos existentes
  - Inserción de nueva información
  - Borrado de datos que ya no son necesarios
  - Algunas bases de datos manejan ***track changes***
- Cada método convierte los parámetros
  - En el SQL requerido para ser ejecutado en la BD

# Compuerta a Tabla de Datos (Cont)

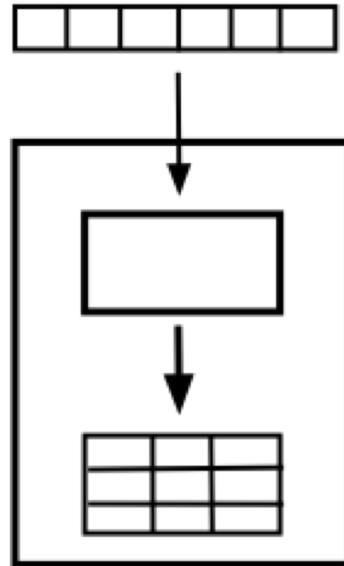
- Altamente útil con la lógica de módulo de tabla
- Se puede utilizar con lógica de dominio
  - Pero la compuerta debe retornar objetos de dominio
  - Esto genera una dependencia bidireccional
    - Capa de Dominio  Capa de Datos
- El uso con rutinas de transacción
  - Depende de si los datos son adecuados
  - En el proceso de manipulación por la rutina

# Paso de Parámetros

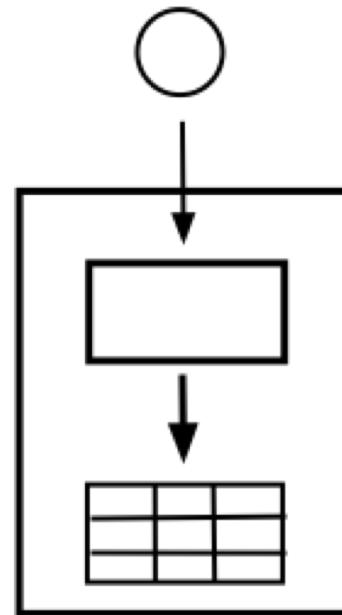
Muchos  
parámetros  
simples



Un arreglo de  
parámetros



Un DTO



# Paso de Parámetros

- Se debe considerar como pasar los parámetros
- Opción 1
  - Pasar en 1 operación de actualización tantos parámetros como campos existan en la BD
  - Esta opción podría ser poco práctica
  - Si se maneja una gran cantidad de campos

# Paso de Parámetros(Cont)

- Opción 2: Arreglo o Correspondencia **Hasmap**
  - Se copia los valores al arreglo
  - Se pasa el arreglo al método como único parámetro
- Opción 3: DTO Data Transfer Object
  - El DTO se llena con los datos necesarios
  - Se pasa el DTO por referencia como parámetro

# Paso de Parámetros(Cont)

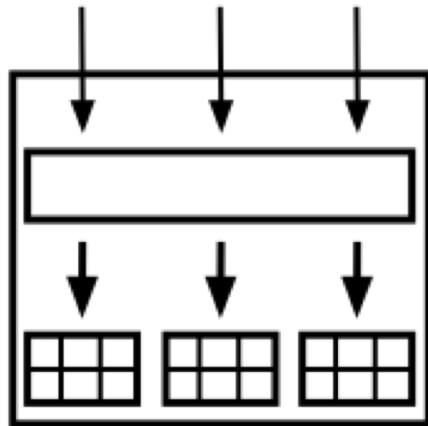
- Opción 3: DTO Data Transfer Object
- Aspectos a considerar
  - Se debe escribir nuevas clases para cada DTO
  - Manejo de datos retornados
    - Es más práctico utilizar un DTO
    - Ya que un Hasmap puede generar inconsistencias
    - Puede que el Hasmap acceda a campos erróneos
    - Se puede retornar el RecordSet de la BD
      - Esto genera dependencia con la BD
      - Si se usa un RecordSet propio dentro del App (evita dependencia)
        - » Genera más trabajo y control en su programación

# Estructura del TDG

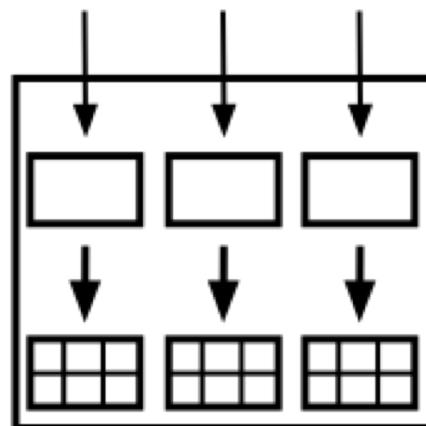
- Se pueden utilizar 3 técnicas
  - Utilizar una única instancia con métodos genéricos
  - Utilizar una instancia por tabla con métodos genéricos
  - Utilizar una instancia por tabla con métodos específicos

# Estructura del TDG

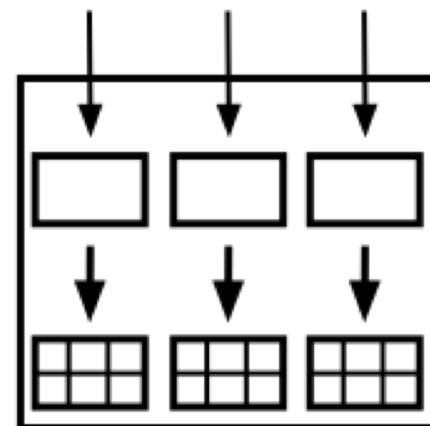
Singleton con  
métodos  
genéricos



Una instancia por  
tabla con métodos  
genéricos



Una instancia por  
tabla con métodos  
específicos



gateway.insertar(tabla,  
parámetros)

profesor.insertar(  
parámetros)

profesor.insertar(cédula,  
nombre,dirección,  
teléfono)

# Única Instancia/Métodos Genéricos

- Esta única instancia accede a todas las tablas
- Cada método recibe parámetros
  - Indican la tabla donde realizar la operación
  - Indican los parámetros necesarios de la operación
- Opera muy bien en apps sencillas

# Una Clase por Tabla

## Métodos Genéricos

- No es necesario dar el nombre de la tabla
- Los nombres de datos y parámetros
  - No reflejan los nombres de los campos en la BD
- Permite realizar verificaciones de datos
  - Verificaciones de datos específicos de la tabla
- Tiene un nivel más alto de granularidad

# Una Clase por Tabla

## Métodos Específicos

- Está asociada con la BD
  - Nombres de los métodos
  - Nombres de los parámetros

# Ventajas del TDG

- Se encapsulan las instrucciones SQL
  - El desarrollador no necesita conocer SQL
  - Las instrucciones SQL quedan ocultas en el TDG
- Se puede manipular vistas
  - Interpretarlas como tablas
  - La complejidad se maneja internamente

# Desventajas del TDG

- Está orientado a modelo de datos
  - Las clases siguen modelos de datos
  - No el modelo de dominio
  - Para dominios complejos
    - Esto puede aumentar la complejidad del ambiente
- El paso de parámetros es laborioso
  - El intercambio de información puede ser laborioso
  - Es importante analizar su impacto

# Compuerta a Fila de Datos

# Compuerta a Fila de Datos

- En inglés Row Data Gateway (RDG)
- Desventajas de poner código de BD en objetos
  - El objeto tiene lógica del dominio interna
    - Agregar el acceso a BD incrementa la complejidad
    - Complejidad que se traduce en todo aspecto
      - Se afecta el desarrollo
      - Se afecta las pruebas
      - Aumenta el consumo en memoria
      - Disminuye la velocidad en general

# Compuerta a Fila de Datos (Cont)

- El RDG crea objetos que lucen como registros
- Opera dentro del margen del programa
  - Utiliza el mismo lenguaje de programación
  - El detalle de consultas a BD está oculto

# Compuerta a Fila de Datos (Cont)

- Se comporta como un registro simple
  - Como si fuera una fila de una base de datos
  - Cada columna (BD) está asociada con un campo
  - Maneja cualquier conversión necesaria
    - Entre los datos de la BD y los datos en memoria RAM
    - Dichas conversiones son bastante simples
- Los datos se mantienen en memoria en RDG
  - El usuario los puede consultar directamente
  - Actúa como una interfaz para cada fila de datos

# Compuerta a Fila de Datos (Cont)

- Trabaja muy bien con Rutinas de Transacción
- Debe contar con objetos finder
  - Este busca el registro en la BD
  - Coloca la info dentro de la instancia de RDG
  - Es normal que haya uno por cada tabla de la BD
- Se puede confundir con el Registro Activo
  - La compuerta no incluye lógica del dominio
  - El Registro activo sí
  - La compuerta tiene sólo la lógica para acceder la BD

# Compuerta a Fila de Datos (Cont)

- Se puede utilizar con una tabla o con una vista
  - Esto puede complicar las actualizaciones
  - Se debe actualizar todas las tablas en la vista
- Se puede tener más de una RDG por tabla
  - El primero puede deshacer el trabajo del segundo
  - Este problema no se puede prevenir
  - El programador debe ser consciente de esto

# Compuerta a Fila de Datos (Cont)

- Casi siempre se utilizan con rutinas de transacción
  - Así el código de acceso no se duplica en las rutinas
  - El código de acceso a las BD están en la compuerta
- No se recomienda usarlo con Modelo de Dominio
  - Si la asociación entre tablas y objetos es simple
    - Es mejor utilizar el Registro Activo
    - Así se evita una capa adicional de código
  - Si la asociación es compleja
    - Es mejor utilizar el Mapeador de Datos
    - Esto desacopla la BD de los objetos de dominio

# Compuerta a Fila de Datos (Cont)

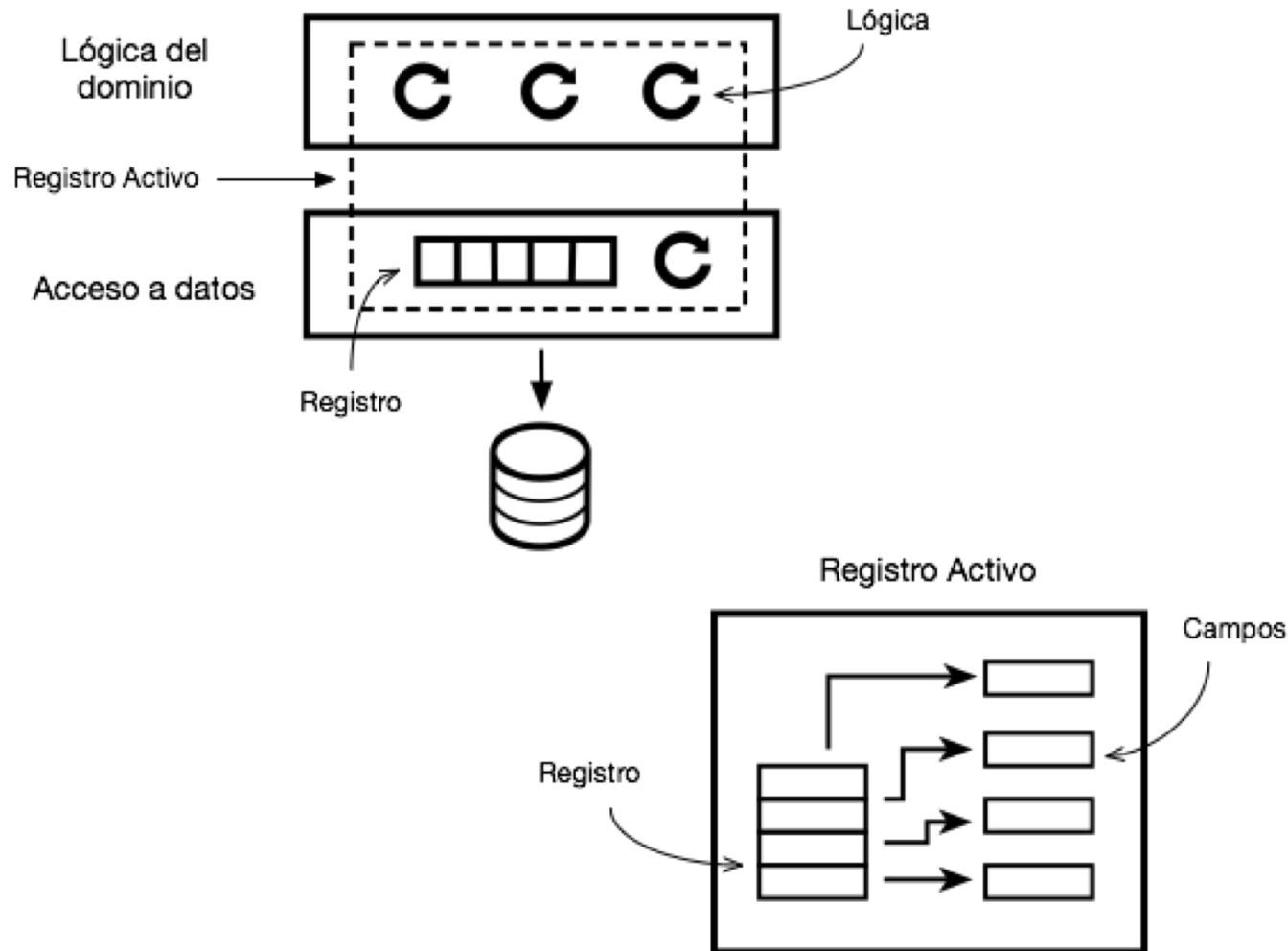
- La RDG permite aislar objetos del dominio
  - Evitando mezclarlos con la BD
  - Muy conveniente si la BD cambia constantemente
- Cuando se usa a gran escala es complicado
  - Maneja 3 representaciones de datos
    - Una de la lógica del dominio
    - Una de la compuerta
    - Una de la base de datos
  - Por eso se recomienda un RDG que refleje la estructura de la BD

# Compuerta a Fila de Datos (Cont)

- Cuando se utiliza con Rutinas de Transacción
  - Suele aparecer código repetido en las rutinas
  - Ese código se puede mover a la compuerta
- Si se transfiere la lógica del Dominio
  - En vez de la lógica de acceso a los datos
  - La compuerta se convertirá en un Registro Activo

# Registro Activo

# Registro Activo



# Registro Activo (RA)

- Un objeto envuelve un registro de la tabla de la BD
- El objeto encapsula el acceso a los datos
- El objeto agrega la lógica del dominio a dichos datos

# Registro Activo (RA)

- El objeto tiene tantos datos como columnas tiene la tabla
- Los datos se modifican en los métodos
- El registro activo trabaja mejor con
  - Modelo de Dominio
  - Las clases se asocian a la estructura de la BD
  - El RA es responsable de guardar/cargar los datos desde la BD
  - También por la lógica del dominio que actúe sobre el

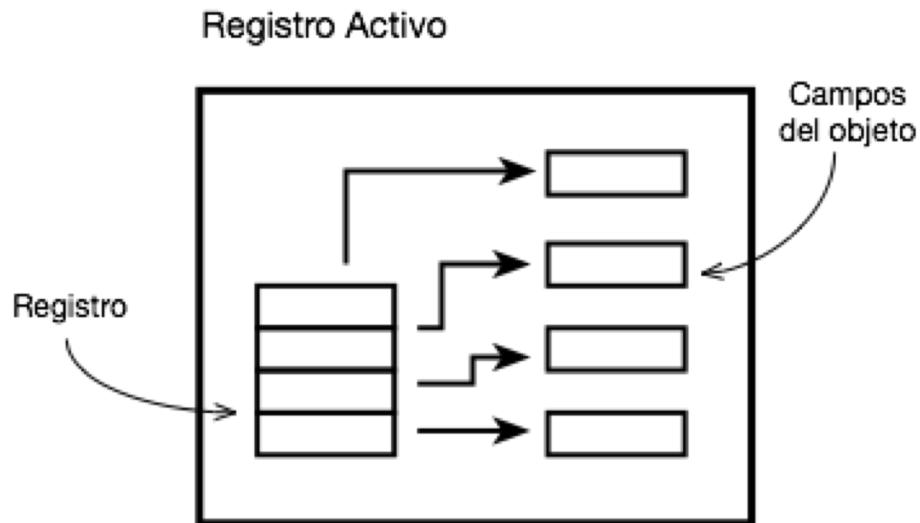
# Registro Activo (RA)

- El RA tiene el mismo tipo de dato por campo/columna que la BD
- No es necesario realizar conversiones
- El RA representa una fila de datos de la tabla
- El RA trabaja con tablas y vistas de la BD
- Las vistas
  - Implica más trabajo el conectar el RA con ellas
  - Son más útiles cuando se trabaja con reportería

# Registro Activo (RA)

- Trabaja muy bien con la Lógica del Dominio
  - Esta no es muy compleja
  - Maneja: Crear, Leer, Actualizar y Borrar (CRUD)
- Los métodos típicos de un RA
  - Incluyen métodos para obtener datos
  - Operaciones CRUD
  - Validaciones
  - Cálculos específicos del dominio

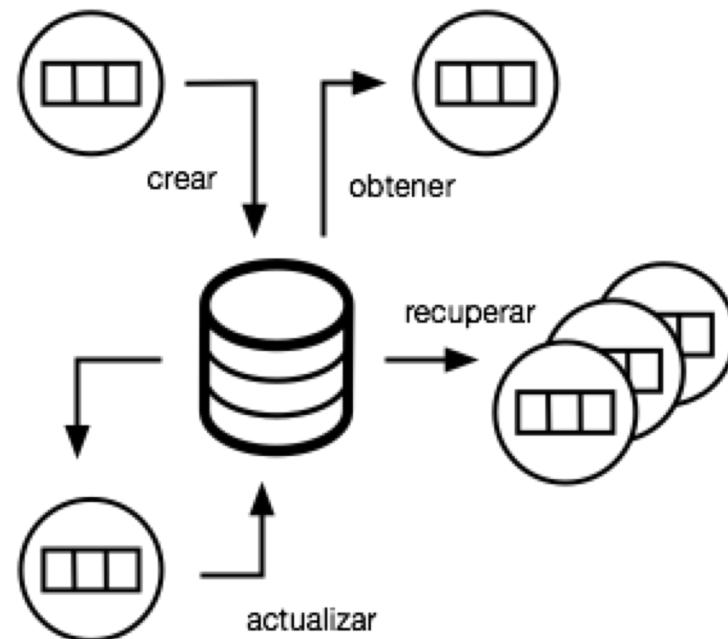
# Registro Activo (RA)



# Tipos de Métodos

- Construye una instancia de un RA
  - Basado en una fila, resultado de un SQL
- Construye una nueva instancia
  - Para posteriormente insertarla en la tabla
- Busca registros para las consultas comunes
  - Esto retorna objetos de tipo RA
- Actualiza la BD e inserta los datos en el RA
- SET/GET para los valores de los campos del RA
- Implementa piezas de la lógica del negocio

# Tipos de Métodos



# Tipos de Métodos

- Los Set y los Get puede hacer operaciones adicionales
  - Convertir los tipos de dato de SQL a los del App
  - Puede retornar un AR del tipo de una tabla relacionada
- Los métodos de búsqueda
  - Normalmente son estáticos
  - También se puede configurar una clase finder
    - Esta implementa métodos de búsqueda

# Acceso a Datos Relacionados

- Al tener un campo en el RA por cada columna
  - Hay que tomar en cuenta las llaves foráneas
- Para las llaves foráneas hay dos enfoques
  - Mantener el Valor Crudo
  - Manejar la referencia del objeto asociado

# Mantener el Valor Crudo

- Es dejar que el RA tenga un valor adicional
  - El valor crudo de la llave foránea
  - Esto obliga a recuperar el objeto asociado
    - El proceso es manual
    - Se utiliza el valor curdo de la llave foránea
- Inconvenientes
  - Puede manejarse valores inconsistentes
  - El programador debe generar código adicional
    - Para poder recuperar el objeto asociado

# Manejar la Referencia del Objeto Asociado

- Se maneja la referencia a otro objeto tipo RA
- Cuando se accede al RA asociado
  - Se debe recuperar la info adecuada desde la BD
- El proceso es transparente al programador
  - No se entera de donde se recuperan los datos
- Algunas librerías cargan los RA relacionados antes
  - Esto sucede sin que se haga la solicitud del registro
  - Esto mejora el rendimiento y utiliza más memoria

# Registro Activo

## Ventajas

- Oculta la lógica del acceso a la BD
  - El desarrollador no necesita lidiar con código SQL
  - Obtiene los registros como si fueran objetos
- Útil en combinación con rutinas de transacción
  - Evita duplicación de código
  - Se puede iniciar con una Compuerta a Fila de Datos
  - Con el tiempo ir avanzando hasta convertirlo en RA

# Registro Activo

## Desventajas

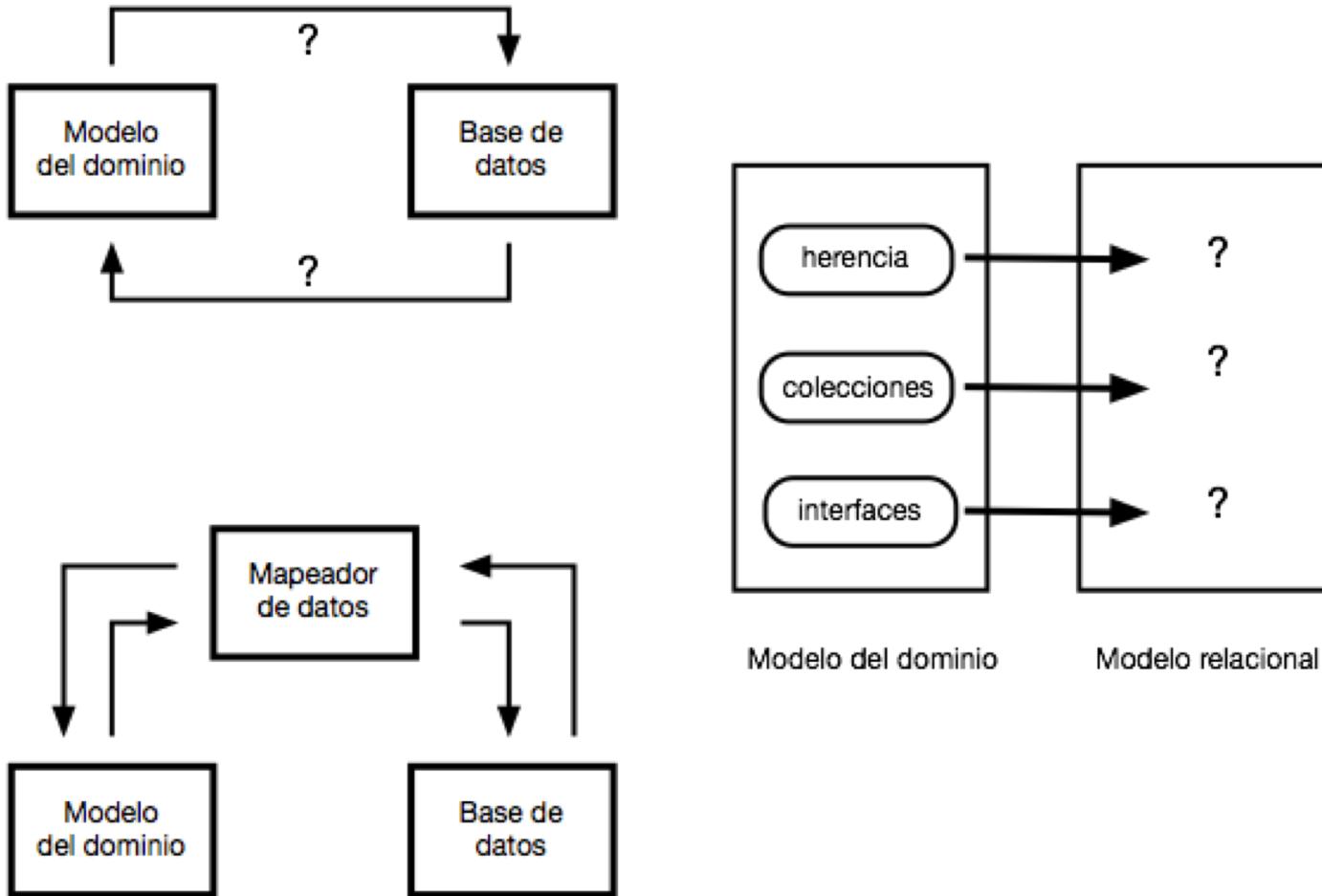
- Solo funciona con modelos de dominio sencillos
  - Se complica con
    - Herencia
    - Relaciones entre clases y Colecciones
- Diseño de objetos dependiente del modelo de datos
  - Si cambia la BD debe cambiar el RA también
- Genera problemas al manipular muchos datos
  - Al manipular cientos de registros
  - El AR no es práctico por el trabajo extra de cargar los datos al objeto

# Mapeador de Datos

# Mapeador de Datos

- Hay gran diferencia entre BD y objetos
  - Colecciones y Herencias no están en las BD
- El modelo de dominio
  - Puede echar mano de las ventajas de las clases
    - Herencia
    - Interfaces
    - Polimorfismo
    - Colecciones
    - Comportamientos asociados
- El modelo relacional de las BD es universal
  - Pero no concuerda en muchos aspectos con clases

# Mapeador de Datos



# Mapeador de Datos

- En ambientes tan heterogéneos
  - Pasar de clases a esquema relacional es complejo
    - En cualquier dirección que se realice el paso
- Todo esto lo resuelve el Mapeador de Datos
  - Es una capa que mueve datos
  - Hace el paso de info entre la BD y los objetos
  - Mantiene independiente todo, incluso el mismo
- La función principal del mapeador es
  - La separación entre Dominio y Fuente de datos

# Mapeador de Datos

- Con el mapeador
  - Los objetos no necesitan conocer la BD
  - Ni siquiera si existe o no una BD
  - No se requiere código SQL de transferencia
  - No se necesita conocer el esquema de la BD

# Búsqueda de Objetos

- Para trabajar el sistema debe cargar objetos
  - La capa de presentación carga objetos iniciales
  - El control pasa a la capa de dominio
  - La capa de dominio ejecuta procesos en sí
  - Esto podría seguir así hasta necesitar cargas de la BD
  - Algunas veces se usa *carga perezosa*
    - Para cargar la info que sea necesaria en el momento

# Búsqueda de Objetos (Cont)

- Los objetos de dominio pueden invocar métodos de búsqueda del mapeador
- La carga perezosa evita esta condición
  - Pero esto conlleva cierta complejidad
- Se puede utilizar una interfaz separada
  - Esto evita que el dominio dependa del mapeador
  - Esto pone cualquier método necesario para el dominio dentro de una clase separada
    - Esta clase formará parte del modelo de dominio

# Mapeo de Datos a Campos de Dominio

- El mapeador debe acceder a los campos de objetos del dominio
- Esto puede ser un problema pues usa métodos públicos
  - Estos ayudan al mapeador
  - No se desean en el objeto de dominio
- La solución: utilizar métodos `protected` en los objetos de dominio
  - Puede provocar confusión si incluye objetos que no son parte del dominio

# Mapeo de Datos a Campos de Dominio

- Otra solución
  - El mapeador utiliza reflexión para acceder a los objetos del dominio
  - Este mecanismo sobrepasa las reglas de visibilidad
  - A la hora de codificar y ejecutar esto es más lento
- Opción adicional
  - Crear métodos públicos que una condición especial
    - Esto permite utilizarlo solo en el contexto de carga de la BD
    - En cualquier otro caso provoca una excepción

# Creación de Objetos

- Hay dos opciones para la creación
  - Utilizar un constructor enriquecido
  - Utilizar un constructor vacío
- Constructor enriquecido
  - Se pasan todos como parámetro los valores
  - Esto genera problemas con referencias cíclicas entre objetos
  - Al cargar un objeto puede cargar otro adicional
  - El adicional querrá cargar el primero
  - Esto genera un stack overflow
  - Para evitar eso se requiere de carga perezosa
  - Pero la carga perezosa es complicada de programar

# Creación de Objetos (Cont)

- Constructor vacío
  - Este crea el objeto con sus campos vacíos
  - El objeto se agrega al repositorio adecuado
  - Debe haber métodos que modifique los valores de los campos (deben ser públicos)
  - Puede utilizar la reflexión para cargar los valores

# Ventajas del Mapeo de Datos

- Modelo y Base de Datos independiente
  - Permite que cada uno evolucione a su ritmo
  - El caso más común es con el modelo de dominio
    - Así se puede ignorar la BD y sus cambios
    - El modelo de dominio no tiene idea de la BD
    - El modelo de dominio no le interesa la estructura de la BD

# Ventajas del Mapeo de Datos(Cont)

- Independencia del Motor de BD
  - Ya que el Mapeador realizar toda la invocación de instrucciones propias del motor de BD
  - Se puede cambiar el motor sin cambios en el código
  - No es necesario agregar código adicional
  - Todo esto con el mapeador de datos adecuado
  - Se permite utilizar la BD que se desee

# Ventajas del Mapeo de Datos(Cont)

- Integración de Datos
  - El mapeador puede convertir esquemas de BD externas
  - Muchas veces el utilizar esquemas externos no es simple
  - El mapeador se vuelve altamente útil para esta labor

# Desventajas del Mapeo de Datos

- Complejidad Adicional
  - El mapeador genera una capa extra
  - Dicha capa agrega una complejidad adicional
  - Esto puede disminuir el rendimiento
    - En parte o en todo el sistema

# Proyecto

- La idea es crear un proyecto Web en 3 capas
  - Capa de Presentación
  - Capa de Lógica del Dominio
  - Capa de Datos
- Puede poner en práctica cualquier esquema para las diferentes capas (Dominio y Datos)
  - Debe justificar la decisión tomada
  - Por qué seleccionamos el esquema en cada capa?

# Proyecto (Cont)

- El sitio web debe ser para E-Commerce
- Debe tener una base de datos con productos
  - Los productos se pueden buscar y mostrar
  - La base de datos debe guardar toda la información de cada producto

# Proyecto (Cont)

- El usuario puede usar el website para
  - Buscar por nombre el producto ("like")
  - El sitio debe mostrar los artículos encontrados
    - Nombre, descripción, fotografía, precio
  - El usuario puede agregar el producto al carrito
  - En el carrito de compras el usuario puede
    - Completar la compra
    - Ver cada producto
    - Eliminar ítems del carrito

# Proyecto (Cont)

- Tipos de sitio Web
  1. Venta de cursos en línea
  2. Venta de Medicinas
  3. Venta de equipo de cómputo
  4. Venta de repuestos de carro
  5. Venta de repuestos de motocicletas
  6. Venta de pizzas
  7. Venta de licores (Cerveza, vinos, licores)
  8. Venta de repostería/pastelería
  9. Venta de paquetes vacacionales
  10. Venta de implementos deportivos
  11. Venta de zapatos
  12. Venta de Remodelaciones
  13. Venta de Anteojos
  14. Venta de Servicios Médicos
  15. Venta de Servicios Dentales