

Desarrollo de Aplicaciones Web

Tema VII: Estructura
Objeto/Relacional

Proyecto 2

- Comparativo de 2 Sitios Web
 - Desarrolle el trabajo en Grupo
 - Trabaje sobre empresas con múltiples sitios web
 - Diferentes países
 - Preferiblemente que tenga sitio web en CR
 - En Caso de no encontrar, puede trabajar con
 - La empresa de CR
 - Compara con un sitio web de una empresa homóloga
 - Debe generar un comparativo del servicio
 - Para cada una de las empresas
 - Debe hacer un análisis y comparar los servicios disponibles

Proyecto 2 (Cont)

- Debe generar un comparativo del servicio
 - Que tan amigable es el sitio
 - Que permite hacer al usuario
 - Dar un detalle de la tecnología encontrada
 - Al menos en la capa visible
 - Y dar una posible estructura interna de la misma
- Cree una lista de mejoras
 - Debe proveer al menos 5 mejoras
 - Principalmente basado en el sitio de CR
- Entregue un informe completo
 - Formato PDF
 - Se abrirá espacio para su entrega en el Aula virtual

Temas de la Unidad

- Campo Identidad
- Mapeo de Llave Externa
- Objeto Binario Serializado
- Jerarquía Mediante Tabla Sencilla
- Herencia Mediante Tablas de Clase
- Jerarquía Mediante Tablas Concretas

Campo Identidad

Campo Identidad

- El campo identidad de la BD se guarda
 - Dentro de un objeto
 - Esto mantiene la identidad del objeto en memoria
 - La información concuerda con la BD
- La base de datos relacional
 - Necesita llaves (primarias y foráneas)
 - Para identificar los datos almacenados
 - Esto asegura la identidad correcta en la BD

Campo Identidad (Cont)

- Los objetos en memoria no ocupan PK
- Por lo anterior las lecturas no dan problema
- Las escrituras
 - Es necesario vincular la BD
 - Al sistema de objetos en memoria
 - Esto para evitar inconsistencias

Campo Identidad

Implementación

- Se debe seleccionar el tipo de llave en la BD
 - Esto puede ser un escenario ideal
 - No siempre se puede crear una llave nueva
 - La base de datos es una estructura actual
 - Ya contiene sus llaves definidas
- Es ideal trabajar con llaves de BD
 - Estas son únicas para todas las tablas de la BD
 - Es más conveniente si se usa el mapeo de identidad
 - Máxime si hay jerarquía de objetos
 - Y esto genera jerarquía de tablas en la BD

Campo Identidad

Implementación (Cont)

- Para crear un objeto se necesita una llave
- Lo anterior se puede hacer de 3 maneras
 - Que la BD cree la llave automáticamente
 - Utilizar un Global Unique Identifier (GUID)
 - Generar la llave por cuenta de la aplicación

Llaves Autogeneradas

- Son las más fáciles
- La BD genera las llaves únicas
 - Cada vez que se introduce 1 registro nuevo
 - La llave se genera por la BD
 - El problema es que esto depende del motor de BD
 - Esto puede causar problemas para el mapeo
 - Objeto-Relacional

Llaves Autogeneradas

- El método más común es
 - Crear un campo autogenerado
 - El mismo es auto-incremental
 - Todo el proceso sucede al insertar una fila
 - Este valor es difícil de determinar por el App
- En escenarios donde hay una factura
 - Contiene artículos y número de orden y factura
 - Esto implica que se necesita el número de la orden
 - Para generar la llave única externa de cada artículo
 - La llave es necesaria a priori, para guardar la información
 - Pero esto no es posible por como trabajan las BD

Llaves Autogeneradas(Cont)

- Las razones anteriores hacen difícil su uso
 - Por escenarios donde hay objetos conectados
 - Esto implica información cruzada entre tablas

Global Unique Identifier (GUID)

- Es un número generado por la máquina
- Garantizado como único en todas las máquinas
- Algunas plataformas proveen APIs para esto
- El algoritmo utiliza
 - La dirección de la NIC
 - La hora, en nanosegundos
 - Número de identificación de chip
 - Entre otros datos
- El número es único y una llave segura para usar

Global Unique Identifier (GUID)

- La desventaja
 - El número generado normalmente es muy grande
- Lo anterior puede ser un grave problema
 - Cuando alguien tiene que digitar la llave
 - Cuando hay que construir un SQL
 - La llave larga es difícil de leer y escribir
 - Pueden dar problemas en rendimiento
 - Al utilizar índices principalmente

Llaves Propias: MAX

- Una opción es buscar la llave máxima actual
 - MAX de SQL
- Esto puede ser útil cuando
 - La BD es pequeña
 - Las inserciones son pocas y/o raras
 - Pues esto bloquea la BD
 - Generando problemas de rendimiento
 - Debe manejar aislamiento completo
 - Para que no haya más de 1 transacción con el mismo ID

Llaves Propias: Tabla de Llaves

- Se puede manejar una tabla de llaves
 - Contiene: nombre del id y el próximo valor libre
- Con llaves únicas de BD
 - La tabla tendrá una sola fila
- Con llaves únicas por tabla
 - Se tendrá una tupla por cada tabla de la BD
- En su uso
 - Se lee y anota el número
 - Se incrementa y se escribe de nuevo en la fila

Llaves Propias: Tabla de Llaves (Cont)

- Se pueden tomar muchas llaves al mismo tiempo
 - Esto reduce el número de llamadas a la tabla
 - Reduce las llamadas por tal concepto a la BD
 - Reduce la contención en la tabla de llaves
- Es importante si se usa
 - Hacer la transacción con las llaves separada
 - De la transacción en la tabla normal
 - Para tratar de evitar bloqueos muy largos

Llaves Propias: Tabla de llaves (Cont)

- Los bloqueos creados por la tabla de llaves
 - Llave de BD: bloqueo en cualquier lugar
 - Llave de tabla: bloqueo en la tabla correspondiente

Llaves Propias: Número Aleatorio Generado en la Capa de Dominio

- Este número se genera en la aplicación
- Debe ser de rango amplio
 - Esto evita que el número sea difícil de repetir
 - Evitando la duplicidad de 1 llave específica
 - Si la coincidencia ocurre
 - Se aborta el proceso y se genera otro número
 - Este esquema es posiblemente el más simple

Ventajas del Campo de Identidad

- Es útil cuando
 - Hay una relación directa entre la memoria y la BD
- Lo anterior se ve claramente cuando se usa
 - Modelo del Dominio
 - Pasarela a Fila de Datos
- No tiene ventaja alguna para
 - Rutinas de transacción
 - Módulo de Tabla
 - Pasarela de Tabla de Datos

Mapeo de Llave Externa

Mapeo de Llave Externa

- Consiste en mapear una asociación entre
 - Objetos de la aplicación
 - Con referencia a una llave foránea entre tablas
- Los sistemas más simples orientados a objetos
 - Tienen todo tipo de relaciones entre los objetos
 - Esas referencias deben ser guardadas en la BD
 - Pero no se puede guardar los datos en crudo
 - Tampoco se puede guardar su referencia en Memoria
 - Algunos objetos referencian colecciones de otros
 - Esto hace el proceso de guardado todavía más complejo

Mapeo de Llave Externa

Implementación

- Se puede utilizar el campo Identidad
 - Cada objeto usa la llave de la BD de su tabla
- Si dos objetos están asociados (unidos)
 - Se puede sustituir la llave por una llave externa
 - Esto se puede dar cuando los datos son simples
- Cuando se trabaja con colecciones de objetos
 - La colección no se puede guardar en la BD
 - Se necesita invertir el sentido de la referencia

Mapeo de Llave Externa

Implementación

- Cuando se trabaja con colecciones
 - Se complica al actualizar datos
 - Puede incluir agregar o quitar algunos objetos
 - Cómo se puede saber qué modificar en la BD?
 - Para manejar lo anterior hay tres opciones
 - Borrar e insertar
 - Añadir un puntero hacia atrás
 - Encontrar las diferencias

Mapeo de Llave Externa

Implementación

- Borrar e Insertar
 - Se elimina toda la asociación entre
 - Los registro de la Colección
 - El objeto principal
 - Se cree una nueva asociación
 - Que incluya cualquier cambio
 - Con los objetos que aún estén en la colección
 - Puede sonar simplista y mediocre
 - Máxime cuando no hay que sacar objetos de la colección
 - Pero es simple de implementar y funciona bien

Mapeo de Llave Externa

Implementación

- Añadir un puntero hacia atrás
 - Agrega un puntero en el objeto nuevo
 - El puntero apunta al objeto principal
- Esto facilita las actualizaciones
 - Con enfoque para los campos con valor único
 - Los cuales están en el otro lado de la relación

Mapeo de Llave Externa

Implementación

- Encontrar las diferencias tiene 2 métodos
 - Buscar el estado actual de la BD
 - Se lee la colección de nuevo en la BD
 - Si está en la BD y no en la memoria: se debe eliminar
 - Si está en memoria y no en la BD: se debe agregar
 - Después la app decide que hacer con cada elemento

Mapeo de Llave Externa

Implementación

- Encontrar las diferencias tiene 2 métodos
 - Cuando se lee la primera vez
 - Se debe mantener lo que se lee
 - Cualquier cosa que se agregue a la colección se revisa
 - Si es un nuevo objeto, se debe añadir a la BD
 - Se debe buscar la fila enlazada a la BD
 - Y se actualiza su llave externa
 - Con esto se señala al registro principal actual

Mapeo de Llave Externa

Implementación

- Encontrar las diferencias tiene 2 métodos
 - Cuando se lee la primera vez (Borrado de datos)
 - Hay que saber si el elemento
 - Se movió a otra colección
 - » Se debe actualizar cuando actualiza la colección
 - Si no tiene colección asociada
 - » Se debe borrar la llave externa asociada al objeto
 - Si se ha eliminado por completo
 - » Se debe borrar el objeto cuando se haga limpieza

Objeto Binario Serializado

Objeto Binario Serializado

- Se guarda un grafo de los objetos
- Se utiliza la serialización en un único objeto
 - Dicho objeto es un LOB (Large Object)
- Este objeto se almacena en un campo de la BD
- Los modelos de objetos suelen
 - Manejar mucha complejidad para objetos pequeños
 - Mucha de su información es de relaciones
 - Que representa los vínculos entre objetos
- Todo esto se puede solucionar con el grafo

Objeto Binario Serializado

- Se guarda un grafo de los objetos
- Se utiliza la serialización en un único objeto
 - Dicho objeto es un LOB (Large Object)
- Este objeto se almacena en un campo de la BD
- Los modelos de objetos suelen
 - Manejar mucha complejidad para objetos pequeños
 - Mucha de su información es de relaciones
 - Que representa los vínculos entre objetos
- Todo esto se puede solucionar con el grafo

Objeto Binario Serializado (Cont)

- El modelo muestra relaciones con naturalidad
 - El patrón de composición
 - Representa jerarquías organizacionales
 - Se pueden crear métodos simples para extraer
 - Ascendientes
 - Hermanos
 - Descendientes
 - Otras relaciones comunes entre objetos de la jerarquía

Objeto Binario Serializado (Cont)

- Para el esquema relacional propuesto
 - No es simple introducir toda la información
 - El esquema básico es una tabla
 - Con una llave externa progenitor
 - La manipulación del esquema requiere
 - Muchas combinaciones
 - Que la hacen lenta y torpe de manipular

Objeto Binario Serializado (Cont)

- Para facilitar todo el proceso se usa el LOB
 - Esto cuando la información no se necesita separar
 - Se serializa la información para guardarla
 - La información se escribe en un único objeto
 - Que consiste de un Large Object
 - El cual se guarda en una tabla especial

Objeto Binario Serializado

Implementación

- Se puede crear mediante
 - Binary Large Object (BLOB)
 - Caracter Large Object (CLOB)

Objeto Binario Serializado

Implementación

- Binary Large Object (BLOB)
 - Es más fácil de implementar
 - Muchas plataformas permiten la serialización
 - De forma automática de un grafo de objetos
 - Se serializa a un buffer
 - Luego se guarda el buffer en el campo correspondiente
 - Quedando almacenado en la BD
 - Es fácil de programar pero
 - La BD debe soportar tipos binarios de datos
 - El grafo no se puede reconstruir sin el objeto
 - Si una clase cambia y hay histórico de su serialización
 - Puede que no se puedan leer esos datos históricos

Objeto Binario Serializado

Implementación (Cont)

- Caracter Large Object (CLOB)
 - Se usa una cadena de texto que contiene el grafo
 - La cadena contiene toda la información necesaria
 - La cadena es lo que se serializa
 - La cadena es de fácil lectura
 - Esto ayuda a su fácil navegación en la BD
 - Este enfoque suele necesitar más espacio
 - Puede ser necesario generar un programa
 - Para manejar los datos de la cadena de texto
 - Es normalmente más lento que la serialización binaria

Objeto Binario Serializado

Implementación (Cont)

- Uso de XML
 - Muchas de las desventajas de CLOB se eliminan
 - Hay muchos analizadores XML
 - Esto elimina la necesidad de escribir uno propio
 - XML es un estándar altamente reconocido
 - Esto da ventajas al manipular datos adicionalmente
 - El XML consume más espacio
 - Hace que el problema de espacio sea mucho peor
 - Esto se puede comprimir haciendo un BLOB del XML
 - Pero elimina su fácil lectura

Objeto Binario Serializado

Utilización

- Cuando se usa LOB
 - Se debe cuidar los problema de identidad
- Ejemplo: Manejo de ordenes por cliente
 - Si queremos que los datos del cliente estén bien
 - En cada orden de compra no importa la fecha
 - Dependemos de los cambios que se hacen al cliente
 - Se puede entonces crear una tabla con 2 campos
 - El Id del cliente
 - El LOB para sus datos relacionales

Objeto Binario Serializado

Utilización

- Cuando se usa LOB
 - Se debe cuidar los problema de identidad
- Ejemplo: Manejo de ordenes por cliente
 - Si queremos que los datos del cliente estén bien
 - En cada orden de compra no importa la fecha
 - Dependemos de los cambios que se hacen al cliente
 - Se puede entonces crear una tabla con 2 campos
 - El Id del cliente
 - El LOB para sus datos relacionales

Objeto Binario Serializado

Utilización (Cont)

- Cuando se usa LOB
 - Se debe tener cuidado de datos duplicados
 - A veces se da duplicación parcial
 - Es bueno asegurarse que el LOB se accede
 - Únicamente desde el objeto propio
 - Esto elimina la posibilidad de errores
 - También elimina la opción de duplicación
 - Así como la superposición de datos

Objeto Binario Serializado

Utilización (Cont)

- El LOB no se utiliza tanto
- El XML es más generalizado y utilizado
 - Este tiene un enfoque textual fácil de utilizar
- Algunas extensiones de SQL leen XML
 - Pero no es portable entre motores de BD
- El uso del LOB es más práctico si
 - Se puede cortar el modelo y representar el LOB
 - La idea es agrupar una gran cantidad de objetos
 - Los cuales no son susceptibles a consultas externas

Objeto Binario Serializado

Utilización (Cont)

- El LOB serializado
 - No funciona bien cuando
 - Tiene objetos fuera de los objetos de referencia
 - Para corregir este problema se puede
 - Trabajar con un esquema de referencia
 - Que soporte la referencias a objetos dentro de un LOB
 - Esto es difícil de encontrar
 - De nuevo se vuelve muy atractivo usar XML
 - O se puede trabajar con XPath

Jerarquía Mediante Tabla Sencilla

Jerarquía Mediante Tabla Sencilla

- Consiste en representar una jerarquía
 - Utilizando la herencia de clases
 - Todo esto como una sola tabla
 - Cada campo de la tabla es una distinta clase
 - Las BD no admiten jerarquías o herencias
 - Al hacer el mapeo de los objetos en la BD
 - Se debe considerar como representar la estructura
 - Y como representar la herencia adecuadamente
 - Todo esto en la tabla de la BD

Jerarquía Mediante Tabla Sencilla

- Al hacer el mapeo en una BD relacional
 - Se debe minimizar la dependencia entre tablas
 - Necesaria para procesar la herencia
 - Una única de herencia puede
 - Asignar todos los campos de las clases
 - Presentes en una estructura de herencia
 - Contenido en una única tabla

Jerarquía Mediante Tabla Sencilla: Implementación

- Se tiene una tabla con los datos de las clases
 - Estas clases están presentes en la jerarquía
 - Dicha jerarquía define las herencias
 - Cada clase almacena información relevante
 - En una fila de la tabla
 - Las columnas que no son relevantes se dejan vacías
 - Al cargar un objeto en memoria se necesita
 - Saber que clase instanciar
 - La tabla tiene un campo que indica que clase usar
 - Puede ser el nombre de la clase o un campo de código

Jerarquía Mediante Tabla Sencilla: Implementación

- Si es un campo de código
 - Se debe interpretar para saber que clase utilizar
- El código debe ampliarse
 - Si se agrega una nueva clase a la jerarquía
- Si se usa el nombre de la clase
 - Se puede utilizar para crear una instancia
 - Esta opción ocupa más espacio
 - Puede ser más difícil de procesar por los usuarios
 - Esto puede acoplar más
 - La estructura de clases con la BD

Jerarquía Mediante Tabla Sencilla: Implementación

- En la carga de datos
 - Se lee el código primero
 - Así se conoce la clase a instanciar
- Al guardar los datos
 - El código debe ser escrito por la superclase
 - Dentro de la jerarquía

Jerarquía Mediante Tabla Sencilla: Ventajas

- Solo hay una tabla en la BD para preocuparse
- No hay uniones para recuperar datos
- Si hay refactorización
 - Si se modifican los campos
 - No se requiere cambiar la BD

Jerarquía Mediante Tabla Sencilla: Desventajas

- Los campos son relevantes algunas veces
 - Esto es confuso para los que usan la tabla
- Las columnas que se usan solo por subclase
 - Generan mucho desperdicio de espacio en la BD
- La tabla única puede crecer desmedidamente
 - Muchos índices y bloques frecuentes
 - Esto puede reducir el rendimiento
- Se tiene un solo espacio para nombre de campos
 - Puede ayudar usar el nombre de la clase como prefijo

Herencia Mediante Tablas de Clase

Herencia Mediante Tablas de Clase

- Se representa ja jerarquía de herencias
 - Utilizando una tabla para cada clase
- Un aspecto visible es la falta de correspondencia
 - Entre el objeto y la BD relacional
 - Debido a la ausencia de herencia en la BD
 - Si se requiere que la BD asigne claramente los objetos
 - Permitiendo enlaces en cualquier lugar de la herencia
 - Se puede utilizar este enfoque

Herencia Mediante Tablas de Clase: Implementación

- Es simple pues se tiene
 - 1 tabla por cada clase en el modelo del dominio
 - Los campos de las clases son campos en la tabla
- Es importante como vincular las filas
 - Dentro de las tablas de la BD
- Para esto se puede utilizar una PK común
 - La fila de llave de la subclase y la superclase
 - Corresponden al mismo objeto de dominio

Herencia Mediante Tablas de Clase: Implementación

- Lo anterior pues la tabla de la superclase
 - Tiene una fila por cada fila en las otras tablas
 - Las llaves serán únicas en cada tabla
- Otra alternativa dejar un PK diferente por tabla
 - Y utilizar llaves externas en la tabla de la superclase
 - Esto asocia las filas de las otras tablas
- Esto implica un problema de consulta múltiple
 - Cómo extraer datos de varias tablas de forma eficiente?

Herencia Mediante Tablas de Clase: Implementación

- Una opción es llamar a cada tabla por aparte
 - Esto genera muchas I/O con la BD
- Otra opción es hacer un **join**
 - Si maneja más de 3 tablas se vuelve muy lento
 - Por como la BD maneja los datos y optimizaciones
- Otro problema es definir que tablas unir
 - Para realizar un **join** efectivo
 - Si hay una tabla vacía se puede hacer un **outer join**
 - O leer primero la tabla raíz
 - Luego usar código para decidir que leer después
 - Esto de nuevo genera múltiples consultas

Heredad Mediante Tablas de Clase: Ventajas

- Todas las columnas son relevantes por fila
 - Son tablas más fáciles de entender
 - No se desperdicia espacio
- La relación es muy simple y sencilla entre
 - El modelo del dominio
 - La estructura de la Base de Datos

Heredencia Mediante Tablas de Clase: Desventajas

- Se debe consultar varias tablas para 1 objeto
 - Esto implica un join, o múltiples consultas
- Refactorización de campos (Arriba o abajo)
 - Implica cambios en la BD
- La tabla supertipo puede ser un cuello de botella
 - Porque es utilizada con mucha frecuencia
- La normalización de la base de datos
 - Se hace difícil de entender
 - Principalmente en consultas ad hoc

Jerarquía de Tablas Concretas

Jerarquía de Tablas Concretas

- Es similar a la anterior
- Se representa la jerarquía de herencia
 - Con una tabla por clase concreta de la jerarquía
- Se piensa en las tablas
 - Desde el punto de vista de objetos
 - Se toma cada objeto en memoria
 - Se le asigna una sola fila en la BD
 - Esto implica herencia concreta de tabla
 - Hay una tabla por clase concreta de la jerarquía
 - En la jerarquía de herencias

Jerarquía de Tablas Concretas

Implementación

- Usa una tabla de la BD para cada clase concreta
- Cada tabla tiene las respectivas columnas
 - Tanto para la clase como sus antecesores
 - Los campos de la superclase se duplican
 - A través de las tablas de las subclases

Jerarquía de Tablas Concretas

Implementación

- Es necesario prestar atención a las llaves
 - Asegurar que las llaves son únicas
 - Tanto para la tabla como para toda la jerarquía
 - Si la llave se puede duplicar entre las tablas
 - Se obtendrá varias filas para una llave determinada
 - El sistema debe dar seguimiento a la asignación de PK
 - A lo largo de la estructura de tablas
 - La BD tampoco es confiable en este aspecto
 - La complejidad aumenta si se conecta con otras BD
 - Que pertenecen a otros sistemas

Jerarquía de Tablas Concretas

Implementación

- Si no se puede garantizar la PK entre tablas
 - Se puede tratar de evitar el uso de superclase
 - Se puede usar una llave compuesta
 - Que utilice el identificador de tabla

Jerarquía de Tablas Concretas

Ventajas

- Cada tabla es independiente
- No tiene campos irrelevantes
 - Muy útil para otras aplicaciones
 - Cuando estas no usan los objetos directamente
- No hay que hacer uniones al leer objetos
 - Se evita el uso de join
- Cada tabla se consulta cuando se usa esa clase
 - Esto puede hacer balanceo de cargas

Jerarquía de Tablas Concretas

Desventajas

- Las PK son difíciles de manejar
- No se puede aplicar relaciones de BD
 - En clases abstractas
- Los cambios en la BD puede alterar la jerarquía
 - Tanto hacia arriba como hacia abajo
 - Esto implica modificar la definición de tablas
- Si la superclase cambia en un campo
 - Hay que hacer el cambio en las tablas que lo contiene
 - Esto por la duplicidad de campos de la superclase
- La búsqueda de la superclase
 - Provoca revisar todas las tablas
 - Provoca múltiples accesos a la BD