Capítulo 1

Desnormalización
(Parte 2)

Tablas Pre-joined

- Si dos o más tablas necesitan unirse de forma regular por una aplicación, pero el costo del join es prohibitivo, considere la creación de tablas pre-joined. Estas deberían:
  - no contener columnas redundantes
  - contener sólo las columnas que sean absolutamente necesarias para la aplicación para cumplir con sus necesidades de procesamiento
  - ser creadas periódicamente usando SQL para unir las tablas normalizadas

El costo del join se va a efectuar una sola vez cuando las tablas pre-joined sean creadas. Una tabla pre-joined se puede consultar de manera muy eficiente porque cada nueva consulta no incurre en la sobrecarga del proceso de join de la tabla.

Tablas de Reportes

- Muchas veces es imposible desarrollar un informe para el usuario final utilizando solo SQL.
- Si algunos informes críticos deben ser vistos en un entorno en línea, considere la creación de una tabla que represente el informe.
- La tabla de informe debería:
  - Contener una columna para cada columna del informe
  - Tener un índice de agrupamiento en las columnas que proporcionan la secuencia de presentación de informes
  - no subvertir principios relacionales (por ejemplo, 1FN y los elementos atómicos de datos)

Este tipo de informes requieren un formato especial o manipulación de datos. Los reportes de tablas son ideales para llevar a los resultados de los outer joins u otras sentencias SQL complejas. Si un outer join se ejecuta y se carga a continuación en una tabla, una instrucción SELECT simple puede ser utilizada para recuperar los resultados del outer join.

Mirror Tables

- Si un sistema de aplicación es muy activo, puede ser necesario dividir el procesamiento en dos (o más) componentes distintos.
- Esto requiere la creación de duplicados, o tablas espejo.
- Diferentes decisiones de definición de datos tales como indexación y clustering pueden ser usadas para las tablas de espejo.

Considere un sistema de aplicación que tiene el tráfico en línea muy pesado durante la mañana y primeras horas de la tarde. Este tráfico se compone tanto de consulta y actualización de datos. Procesamiento de soporte de decisiones también se realiza en las tablas de la misma aplicación durante la tarde. El trabajo de producción de la tarde siempre parece alterar el procesamiento de apoyo a las decisiones que causando time outs y dead locks.

Repetición de Grupos

- Considere una aplicación que almacena información de grupos de repetición en la siguiente tabla normalizada:

```
CREATE TABLE SALDOS_PERIODICOS
                (ID_CLIENTE          CHAR(11),
                PERIODO_SALDO              INT,
                SALDO                DECIMAL(15,2),
        PRIMARY KEY(ID_CLIENTE, PERIODO_SALDO)
)
```

En esta tabla se puede almacenar un número infinito de los saldos por cliente, sólo limitada por el almacenamiento disponible y los límites de almacenamiento de los RDBMS.

```
CREATE TABLE SALDOS_PERIODICOS
                    (ID_CLIENTE                 CHAR(11),
                    SALDO_PERIODO1              DECIMAL(15,2),
                    SALDO_PERIODO2              DECIMAL(15,2),
                    SALDO_PERIODO3              DECIMAL(15,2),
                    SALDO_PERIODO4              DECIMAL(15,2),
                    SALDO_PERIODO5              DECIMAL(15,2),
                    SALDO_PERIODO6              DECIMAL(15,2),
          PRIMARY KEY(ID_CLIENTE)
)
```

Un ejemplo de esto después de desnormalización se muestra.
En este ejemplo, sólo seis saldos pueden ser almacenadas por cualquier cliente. El número seis no es importante, pero el concepto de que el número de valores es limitado es importante. Esto reduce la flexibilidad del almacenamiento de datos y debe evitarse a menos que las necesidades de rendimiento disponga lo contrario.

- Antes de decidir la implementación de grupos de repetición como columnas en lugar de filas, hay que estar seguros de que se cumplan los siguientes requisitos:
  - los datos son rara vez o nunca agregados, promediados, o comparados dentro de la fila
  - los datos se producen en un patrón estadísticamente de buen comportamiento
  - los datos tienen un número estable de apariciones
  - los datos suelen ser accesados en conjunto
  - los datos tienen un patrón predecible de inserción y eliminación

Si cualquiera de estos criterios no se cumplen, SQL SELECT puede ser difícil de codificar. Esto debe ser evitado porque, en general, los datos sin normalizar es sólo para hacerlos disponibles más fácilmente.

Datos Derivados

- Si el costo de obtener datos mediante fórmulas complicadas es prohibitivo considere el almacenamiento de los datos obtenidos en una columna en lugar de calcularlo.
- Sin embargo, cuando los valores subyacentes que componen el valor calculado cambian, es imperativo que los datos almacenados derivados también cambien para no reportar información inconsistente. Esto repercutirá negativamente en la eficacia y la fiabilidad de la base de datos.

A veces no es posible actualizar de inmediato los elementos derivados de datos cuando las columnas de los que dependen cambian.
Esto puede ocurrir cuando las tablas que contienen los elementos derivados están fuera de línea. En esta situación, el tiempo de la actualización de los datos derivados debe producirse inmediatamente cuando la tabla está disponible para su actualización.

```
Select    titulo, sum(avance)
From      tituloAutor ta, titulos t
Where     ta.idTitulo=t.idTitulo
Group by idTitulo
```

tituloAutor

idTItulo	Avance
1	10
1	100
2	30
2	2000

titulos

idTItulo	Titulo
1	Ab
2	Cd
3	Ef
4	Fg

join

```
Select    titulo, sum_adv
From      titulos
```

titulos

idTItulo	Titulo	Sum_adv
1	Ab	110
2	Cd	2030
3	Ef	0
4	fg	0

tituloAutor

idTItulo	Avance
1	10
1	100
2	30
2	2000

- Puede crear y mantener una columna de datos derivados en la tabla de títulos, eliminando la unión y el sum en tiempo de ejecución. Esto aumenta las necesidades de almacenamiento, y requiere un mantenimiento de la columna derivada cuando se realizan cambios en la tabla de títulos.

Jerarquías

- Una jerarquía es una estructura que es fácil de soportar usando una base de datos relacionales, pero es difícil recuperar información de manera eficiente.
- Las aplicaciones que se basan en jerarquías muy a menudo presentan tablas desnormalizadas para hacer veloz la recuperación de datos.

- Jerarquías de Departamentos

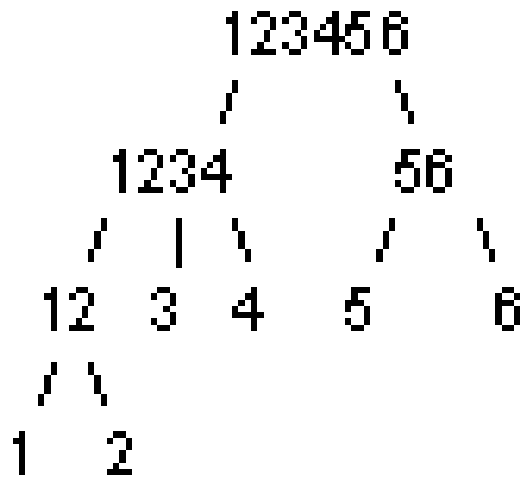

Tabla Departamento

NoDept	NoDeptPadre	…otras columnas
123456	---	
1234	123456	
56	123456	
12	1234	
3	1234	
4	1234	
1	12	
2	12	
5	56	
6	56	

La tabla que se muestra es la aplicación clásica relacional de una jerarquía. Hay dos columnas departamento, una del padre y otra para el hijo.

- A pesar de que la implementación registra efectivamente toda la jerarquía, la creación de una consulta para reportar todos los departamentos a cargo de cualquier otro departamento puede tomar mucho tiempo para codificar e ineficiente para procesar.

```
SELECT      NODEPT
FROM        DEPARTAMENTO
WHERE       NODEPTPADRE='123456'
UNION
SELECT      NODEPT
FROM        DEPARTAMENTO
WHERE       NODEPTPADRE IN
            (SELECT   NODEPT
            FROM      DEPARTAMENTO
            WHERE     NODEPTPADRE='123456')
UNION
SELECT      NODEPT
FROM        DEPARTAMENTO
WHERE       NODEPTPADRE IN
            (SELECT   NODEPT
            FROM      DEPARTAMENTO
            WHERE     NODEPTPADRE IN
            (SELECT   NODEPT
            FROM      DEPARTAMENTO
            WHERE     NODEPTPADRE='123456'))
```

Ejemplo de consulta que devolverá todos los departamentos que informe al nodo empresarial 123456. Sin embargo, esta consulta sólo se puede construir si usted sabe de antemano el número total de posibles niveles que la jerarquía puede lograr. Si hay n niveles en la jerarquía, entonces necesitará n-1 UNIONs.

- Una tabla de "velocidad” puede ser construida.
- Esta tabla muestra el departamento de los padres y todos los departamentos debajo del mismo, independientemente del nivel.
- Contraste esto con la tabla anterior, que sólo se registra los hijos inmediatos para cada padre.
- La "velocidad" tabla también comúnmente contiene otra información pertinente que se necesita por el uso dado. La información típica incluye el nivel dentro de la jerarquía para el nodo dado, si el nodo dado de la jerarquía es un nodo de detalle (en la parte inferior del árbol), y, en caso de pedido corresponde a su nivel es importante, la secuencia de los nodos en el nivel determinado.

NODEPTPADRE	NODEPTHIJO	NIVEL	DETALLE
123456	1234	1	N
123456	56	1	N
123456	12	2	N
123456	1	3	Y
123456	2	4	Y
123456	3	2	Y
123456	4	2	Y
123456	5	2	Y
123456	6	2	Y
1234	12	1	N
1234	1	2	Y
1234	2	2	Y
1234	3	1	Y
1234	4	1	Y
3	3	1	Y
4	4	1	Y
12	1	1	Y
12	2	1	Y
1	1	1	Y
2	2	1	Y
56	5	1	Y
56	6	1	Y
5	5	1	Y
6	6	1	Y

- Luego que la speed table se ha implementado, consultas rápidas se pueden escribir en contra de esta aplicación de una jerarquía.
- La tabla de "velocidad" suele ser construida con un programa escrito en otro lenguaje de alto nivel. SQL solo es por lo general demasiado ineficiente o poco práctico para la creación de una porque el número de niveles en la jerarquía es desconocido o cambiando constantemente.

- Se muestran diferentes consultas informativas que pueden ser muy ineficaces a ejecutar en la jerarquía relacional clásica.

```
SELECT      NODEPTHIJO
FROM        DEPARTAMENTO
WHERE       NODEPTPADRE='123456';

SELECT      NODEPTHIJO
FROM        DEPARTAMENTO
WHERE       NODEPTPADRE='123456'
AND         DETALLE='Y';

SELECT      NODEPTPADRE, NODEPTHIJO, NIVEL
FROM        DEPARTAMENTO
WHERE       NODEPTPADRE='123456';
ORDER BY    NIVEL;
```

Resumen

- Tablas Pre-Joined -> usadas cuando el costo del join es muy alto
- Tablas de Reportes -> Usadas cuando reportes críticos especializados son necesarios
- Tablas Espejo -> Usadas cuando son requeridas tablas concurrentemente por dos tipos diferentes de ambientes
- Tablas Divididas -> Usadas cuando grupos distintos usan diferentes partes de una tabla
- Tablas Combinadas -> Usadas cuando existen relaciones de uno a uno
- Datos Redundantes -> Usadas para reducir el numero de joins de tablas requeridos
- Grupos Repetidos -> Usadas para reducir I/O y posiblemente DASD
- Datos Derivados _> Usadas para eliminar cálculos y algoritmos
- Tablas de “Velocidad” -> Usadas para soportar jerarquías