Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Dirección de la Escuela de Ingeniería en Ciencias y Sistemas

AUMENTO DEL RENDIMIENTO DEL *DATA WAREHOUSE* DE UNA COMPAÑÍA TELEFÓNICA POR MEDIO DEL PARTICIONAMIENTO DE TABLAS E ÍNDICES

AUGUSTO IGNACIO LÓPEZ DE LEÓN
Asesorado por Ing. Everest Medinilla Rodríguez

Guatemala, octubre de 2003

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

AUMENTO DEL RENDIMIENTO DEL *DATA WAREHOUSE* DE UNA COMPAÑÍA TELEFÓNICA POR MEDIO DEL PARTICIONAMIENTO DE TABLAS E ÍNDICES

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

AUGUSTO IGNACIO LÓPEZ DE LEÓN

ASESORADO POR ING. EVEREST MEDINILLA RODRÍGUEZ
AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, OCTUBRE DE 2003

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Sydney Alexander Samuels Milson
VOCAL I	Ing. Murphy Olympo Paiz Recinos
VOCAL II	Lic. Amahán Sánchez Álvarez
VOCAL III	Ing. Julio David Galicia Celada
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Sydney Alexander Samuels Milson
EXAMINADOR	Ing. Marlon Antonio Pérez Turk
EXAMINADOR	Ing. Francisco Javier Guevara Castillo
EXAMINADOR	Ing. Rolando Haroldo Alonso Ordóñez
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

AUMENTO DEL RENDIMIENTO DEL *DATA WAREHOUSE* DE UNA COMPAÑÍA TELEFÓNICA POR MEDIO DEL PARTICIONAMIENTO DE TABLAS E ÍNDICES

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas con fecha 27 de marzo de 2002.

Augusto Ignacio López de León

Guatemala, julio de 2003

Ingeniero
Carlos Alfredo Azurdia Morales
Coordinador de Privados y
Revisión de Trabajos de Graduación
Presente

Estimado Ingeniero Azurdia:

Por este medio me permito informarle que he procedido a revisar el trabajo de graduación titulado **AUMENTO DEL RENDIMIENTO DEL *DATA WAREHOUSE* DE UNA COMPAÑÍA TELEFÓNICA POR MEDIO DEL PARTICIONAMIENTO DE TABLAS E ÍNDICES**, elaborado por el estudiante Augusto Ignacio López de León, y que a mi juicio, el mismo cumple con los objetivos propuestos para su desarrollo.

Sin otro particular, me suscribo de usted,

Atentamente,

Ing. Everest Medinilla Rodríguez
Asesor

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

Guatemala, 17 de septiembre de 2003

Ingeniero
Luis Alberto Vettorazzi España
Director de la Escuela de Ingeniería
en Ciencias y Sistemas

Respetable Ingeniero Vettorazzi:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **AUGUSTO IGNACIO LÓPEZ DE LEÓN**, titulado: **"AUMENTO DEL RENDIMIENTO DEL *DATA WAREHOUSE* DE UNA COMPAÑÍA TELEFÓNICA POR MEDIO DEL PARTICIONAMIENTO DE TABLAS E ÍNDICES"**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia Morales
Coordinador de Privados
y Revisión de Trabajos de Graduación

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos, luego de conocer el dictamen del asesor con el visto bueno del revisor y del licenciado en Letras, del trabajo de graduación titulado: **AUMENTO DEL RENDIMIENTO DEL *DATA WAREHOUSE* DE UNA COMPAÑÍA TELEFÓNICA POR MEDIO DEL PARTICIONAMIENTO DE TABLAS E ÍNDICES**, presentado por el estudiante **Augusto Ignacio López de León**, aprueba el presente trabajo y solicita la autorización del mismo.

ID Y ENSEÑAD A TODOS

Ing. Luis Alberto Vettorazzi España
Director
Ingeniería en Ciencias y Sistemas

Guatemala, 3 de octubre de 2003

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA
DECANATO
Tels. 4760790 al 94 - Ext. 348
Directo: 4769579 - Fax: 4760365
E-mail: ssamuels@usac.edu.gt

REF. DT-119-2003

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ciencias y Sistemas, al trabajo de tesis titulado: **AUMENTO DEL RENDIMIENTO DEL *DATA WAREHOUSE* DE UNA COMPAÑÍA TELEFÓNICA POR MEDIO DEL PARTICIONAMIENTO DE TABLAS E ÍNDICES**, presentado por el estudiante universitario **Augusto Ignacio López de León**, procede a la autorización para la impresión de la misma.

IMPRÍMASE:

Ing. Sydney Alexander Samuels Milson
DECANO

Guatemala, octubre de 2003

/cdes

DEDICATORIA

De DIOS para mis padres, fruto de su esfuerzo.

ACTO QUE COMPARTO CON

MI FAMILIA

Mi papá, mi mamá, Claudia, Pepe, Gloria, Fernando, Luis, Alejandra, Mável, Chusy, Diana María José, Augusto Antonio, Patsy y los que vengan en camino.

Y

todas aquellas personas que me tendieron la mano mientras recorrí este camino, que el Creador de todas las cosas les bendiga.

...el cual quiere que todos los hombres sean salvos
y vengan al pleno
conocimiento de la verdad.

1 Timoteo 2:4

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Base de datos	Colección de datos relacionados, almacenados de tal manera que su contenido pueda ser fácilmente accedido, administrado y actualizado.
Base de datos relacional	Aquella en la que los datos se representan en forma de tablas.
Data warehouse	Repositorio central de datos provenientes de los distintos componentes de una organización, se caracteriza por almacenar grandes volúmenes de información.
DBMS	Programa de computadora que se encarga de administrar los datos almacenados en una base de datos que se encuentra dentro de una computadora.
DBMS relacional	DBMS en el que la base de datos administrada es relacional.
DML	Subconjunto de SQL que incluye las instrucciones para insertar, borrar y actualizar datos.
DSS	Sistema para toma de decisiones, se especializa en consultar información de una base de datos.
Gigabyte	1,073,741,824 *bytes*, también se referencia como Gb.
Llave foránea	Conjunto de columnas dentro de una tabla que identifican a una fila existente en otra tabla (o en la misma).
Llave primaria	Conjunto de columnas dentro de una tabla que identifican de forma única a cada una de las filas de la misma.
Megabyte	1,048,576 *bytes*. También se referencia como Mb.
OLTP	Procesamiento de transacciones en línea.
SELECT	Instrucción de SQL que se utiliza para consultar información

	que se encuentra en la base de datos.
SQL	Lenguaje que se utiliza para comunicarse con un DBMS relacional.
Tabla	Organización de datos en forma de filas y columnas.

RESUMEN

Una compañía telefónica cuenta con un *data warehouse* utilizado para la extracción de información estratégica acerca del comportamiento de sus clientes. La gerencia del *data warehouse* desea disminuir el tiempo de ejecución de las dos consultas principales que se hacen al mismo: el reporte semanal de egresos y el reporte mensual de egresos.

Dado el elevado tamaño de las tablas consultadas se considera la utilización de tablas e índices particionados. El particionamiento consiste en dividir al objeto (tabla o índice) en segmentos más pequeños y manejables sin que exista la necesidad de modificar las aplicaciones que hagan uso de él.

Las tablas consultadas se particionan por rango de fechas logrando disminuir el tiempo de ejecución del reporte semanal a un 2.85% del tiempo original y el tiempo de ejecución del reporte mensual a un 11.51% del tiempo original, con el agregado de que los tiempos de ejecución originales se obtienen usando índices mientras que los nuevos no hacen uso de ellos.

El ahorro de espacio unido al particionamiento permiten aumentar la historia almacenada por las tablas, es decir, se logra almacenar información de más tiempo. Además, cualquier consulta hecha a las tablas tenderá a tener los mismos tiempos de ejecución siempre que implique el recorrer un rango de fechas.

OBJETIVOS

- **General**

Aumentar el rendimiento del *data warehouse* de una compañía telefónica por medio del particionamiento de tablas e índices.

- **Específicos**

1. Definir el concepto de tabla particionada.

2. Definir el concepto de índice particionado.

3. Explicar las distintas operaciones de mantenimiento que pueden realizarse con las tablas e índices particionados.

4. Disminuir el tiempo de ejecución para el reporte de egresos semanales y para el reporte de egresos mensuales.

INTRODUCCIÓN

Imagínese terminar de elegir los abarrotes en el supermercado y ser atendido inmediatamente al llegar a la caja registradora, o bien, entrar al autobanco el último día del mes y ser atendido al llegar. La atención inmediata es un deseo ligado a cualquier ser humano, sobre todo en este tiempo en el que las personas corren de un lugar para otro y que todo precisa.

Así como una persona es cliente de un supermercado o una institución bancaria, los gerentes o tomadores de decisiones de una empresa son clientes de la base de datos de dicha empresa. La información de las operaciones de la compañía (ventas, salarios, etc.) constituyen la materia prima de la labor de los tomadores de decisiones, deseando entre otras cosas, que la base de datos responda a sus requerimientos en el menor tiempo posible (así como el cliente de un supermercado o institución bancaria desea atención inmediata).

El mayor proveedor de información para el tomador de decisiones es el *data warehouse* de la compañía. Un *data warehouse* es un base de datos grande (cientos de *gigabytes* de información) destinada principalmente a atender consultas, siendo la mayoría de ellas relacionales por lo que el lenguaje utilizado por el tomador de decisiones para comunicarse con la fuente de información es el SQL (Lenguaje Estructurado de Consultas, por sus siglas en inglés).

Existe en el mercado guatemalteco una compañía de telefonía celular interesada en disminuir los tiempos de ejecución (aumentar el rendimiento) de las dos principales consultas hechas a su *data warehouse*, estas son: el reporte semanal de egresos y el reporte mensual de egresos. Ambas consultas son hechas sobre las dos mayores tablas

de la base de datos (3,780 Mb. y 11,328 Mb.) existiendo un índice para cada una de ellas con el fin de minimizar el tiempo de ejecución. Actualmente los tiempos de ejecución para los reportes de egresos semanal y mensual son de 7 min. 01.07 seg. y 7 min. 06.01 seg., respectivamente. La compañía telefónica está interesada, principalmente, en disminuir estos tiempos de ejecución para satisfacción de los tomadores de decisiones.

En la actualidad la facilidad de manejo y el aumento del rendimiento de los *data warehouses* se logra por medio de la implementación de tablas e índices particionados, consistente en dividir tanto tablas como índices en segmentos más pequeños y manejables llamados particiones. La disminución de los tiempos de ejecución de los reportes de egresos semanal y mensual para la compañía telefónica por medio del uso de tablas e índices particionados es el tema del presente trabajo.

En el primer capítulo se da a conocer el concepto de tabla particionada incluyendo los distintos tipos de particionamiento de tablas que existen y ejemplos de creación de tablas para cada uno de ellos.

El segundo capítulo explora el concepto de índices particionados, los tipos de particionamiento existentes y ejemplos que ilustran cómo crear un índice particionado.

En el capítulo tres se muestran las distintas operaciones de mantenimiento que pueden realizarse sobre las tablas e índices particionados, incluyendo por ejemplo, cómo juntar dos particiones para formar una sola, cómo dividir una partición en dos y cómo eliminar una partición.

Por último, en el capítulo cuatro, se proponen cuatro opciones de particionamiento para disminuir el tiempo de ejecución de las consultas. En el mejor de los casos el tiempo de ejecución del reporte semanal de egresos se reduce a un 2.85% del tiempo original mientras que el reporte mensual se reduce a un 11.51% del tiempo original, todo

esto sin utilizar índices. La no utilización de índices conlleva la ventaja de aprovechar el espacio "ahorrado" para almacenar más historia.

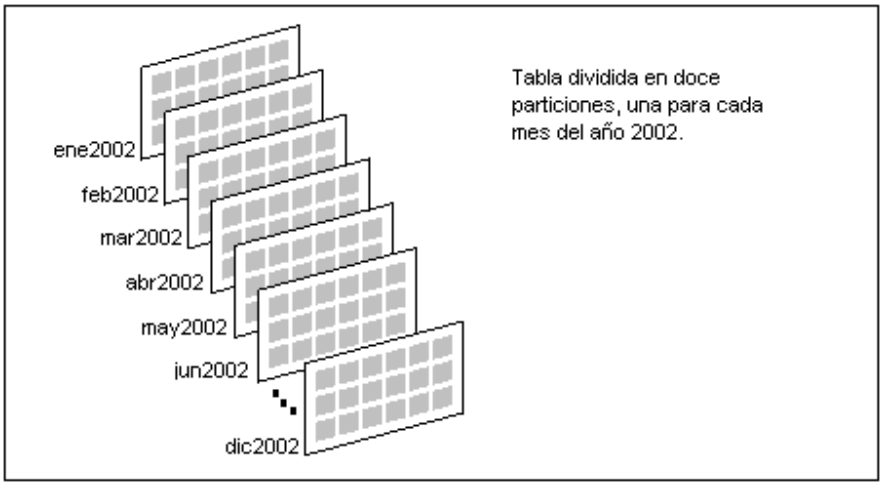
Este trabajo requiere del lector cierta familiaridad con los conceptos de bases de datos, bases de datos relacionales, SQL y arquitectura de computadores.

1. TABLAS PARTICIONADAS

1.1 Definición

Una tabla particionada es una tabla dividida en segmentos más pequeños y manejables llamados *particiones*. Está formada por atributos lógicos y físicos. Los atributos lógicos incluyen definiciones de columnas, de llave primaria, de llave foránea, etc., mientras que los atributos físicos definen aspectos relacionados con su almacenamiento en disco, por ejemplo, su tamaño inicial y su localización física. Los atributos lógicos aplican a toda la tabla mientras que los atributos físicos pueden definirse para toda la tabla y/o para cada una de las particiones. La figura 1 muestra una representación gráfica de una tabla particionada.

Figura 1. Representación gráfica de una tabla particionada



1.2 Características de las tablas particionadas

Cada partición contiene un subconjunto del total de registros de la tabla. Las particiones son independientes entre sí, por lo que las operaciones de carga, consulta, respaldo o recuperación de datos pueden realizarse en una partición dada sin afectar el acceso al resto de las particiones.

Para poder consultar, insertar, borrar o actualizar los registros de una tabla particionada no es necesario modificar la sintaxis de las ya conocidas sentencias `SELECT` y de *DML* (Lenguaje de Manipulación de Datos). Esto significa que las tablas ya existentes en la base de datos pueden ser particionadas sin necesidad de modificar la aplicación que hace uso de ellas. Por ejemplo, si se desea eliminar los registros de las ventas realizadas en la tercera semana de enero de 2002, únicamente se ejecuta la siguiente instrucción:

```
DELETE FROM ventas
WHERE fecha BETWEEN '13/01/2002' AND '19/01/2002';
```

No es necesario especificar la partición donde se encuentran los datos que se quieren eliminar, basta con decir en qué tabla están. Para la aplicación la tabla sigue siendo una sola.

Dado que los atributos físicos pueden definirse para cada partición, éstas pueden colocarse en discos distintos. Asignar las particiones a discos distintos aumenta la disponibilidad de la tabla.

Por ejemplo: la tabla *ventas* se particiona por mes de tal manera que existen las particiones *ene2002*, *feb2002*, *mar2002* y así sucesivamente hasta *dic2002*, cada una guardando las ventas realizadas en el mes correspondiente, es decir, *ene2002* guarda todas las ventas realizadas en el mes de enero del año 2002, *feb2002* guarda todas las

ventas realizadas en el mes de febrero de 2002 y así sucesivamente. La tabla es una, *ventas*, pero tiene doce particiones, una por cada mes del año 2002.

Cada partición se almacena en un disco distinto de tal manera que existen doce discos a través de los cuales está distribuida la tabla.

Supongamos ahora que se daña el disco en donde se encuentra la partición *nov2002*, qué sucede con la tabla? La respuesta es: nada, la tabla sigue disponible; qué sucede con las restantes once particiones? La respuesta es la misma: nada, todas las demás particiones siguen disponibles.

El particionamiento permite realizar operaciones en paralelo lo que conlleva una mejora en el rendimiento. Por ejemplo, si se desea saber qué cantidad de dinero se obtuvo por las ventas en los meses de enero y febrero del año 2002, la consulta puede leer al mismo tiempo las particiones *ene2002* y *feb2002* en lugar de hacerlo secuencialmente, esto si el servidor de datos cuenta con más de un procesador, claro está.

1.3 Columnas de particionamiento y llave de particionamiento

Cada registro de la tabla es asignado de forma no ambigua a una única partición, a esto se le llama *direccionamiento de registros*. Las columnas cuyos valores determinan la partición en la que debe ir el registro se conocen como columnas de particionamiento, los valores de estas columnas para un registro específico constituyen la llave de particionamiento para ese registro.

Se dice que la llave de particionamiento es simple cuando está formada por una sola columna y compuesta cuando está formada por más de una columna.

Los valores para las columnas de particionamiento que definen cada partición deben ser literales o cualquiera de las funciones `TO_DATE` o `RPAD`, tal como se muestra en el ejemplo de la figura 2.

1.4 Creación de una tabla particionada

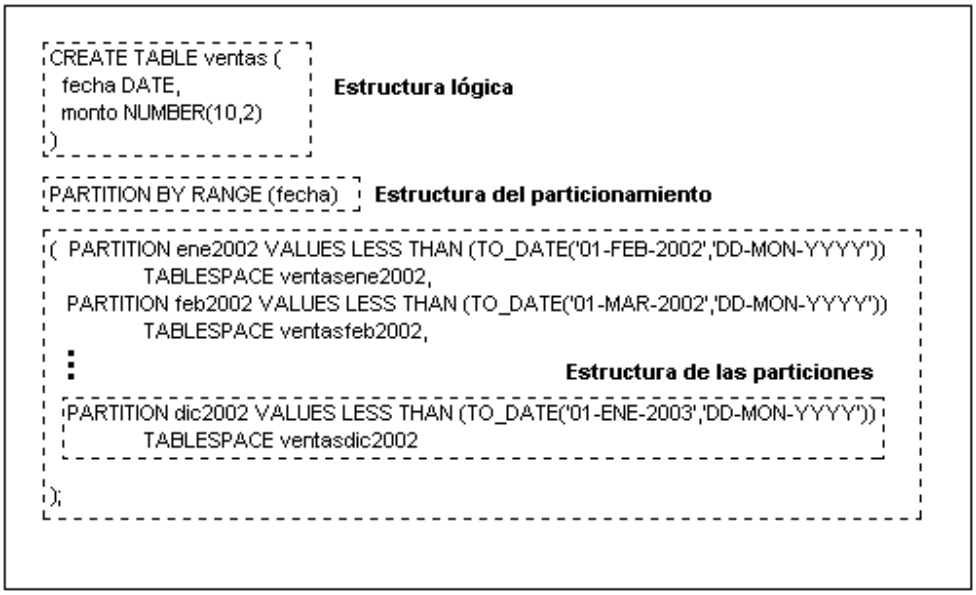
La creación de una tabla particionada incluye la definición de tres secciones (como se muestra en la figura 2):

1. La estructura lógica de la tabla.
2. La estructura del particionamiento, que incluye el método de particionamiento y la llave de particionamiento.
3. La estructura de cada una de las particiones, que tiene dos partes:

   Los atributos lógicos, como el nombre de la partición.

   Los atributos físicos, como la localización física.

Figura 2. Ejemplo de creación de una tabla particionada



1.5 Métodos de particionamiento

Existen cinco métodos de particionamiento:

1. Particionamiento por rango
2. Particionamiento por lista
3. Particionamiento por dispersión
4. Particionamiento rango-lista
5. Particionamiento rango-dispersión

Los primeros tres métodos de particionamiento se conocen como particionamiento simple mientras que los últimos dos se denominan métodos de particionamiento compuesto. En el particionamiento simple sólo se crea un nivel de particiones. El particionamiento compuesto implica la creación de dos niveles de particiones: las particiones y las subparticiones.

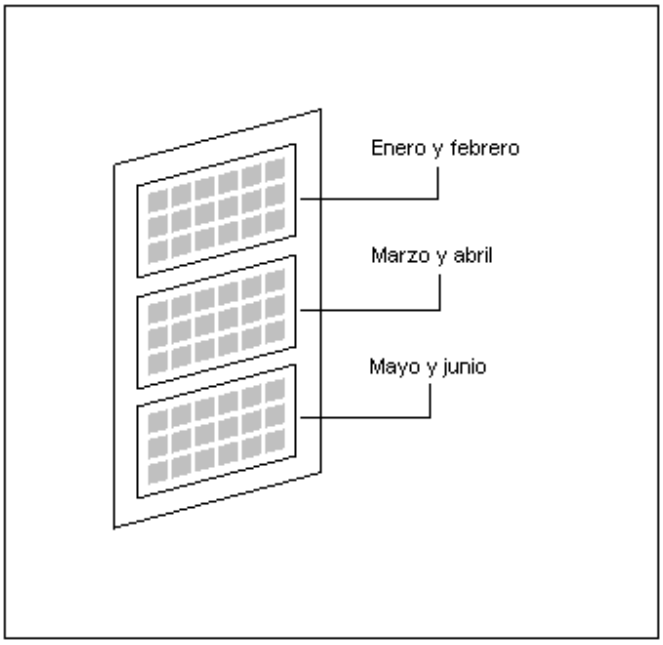
En las siguientes secciones se tratan los cinco métodos de particionamiento incluyendo ejemplos que muestran cómo crear una tabla utilizando cada uno de ellos.

1.5.1 Particionamiento por rango

El particionamiento por rango mapea registros a las particiones basándose en rangos de valores de la llave de particionamiento, definidos para cada partición. Es el método de particionamiento más común y se usa generalmente con fechas. Por ejemplo, se puede crear la tabla *ventas_r* particionada por mes.

La figura 3 muestra una representación gráfica de una tabla particionada por rango.

Figura 3. Tabla particionada por rango



Cuando se crea una tabla particionada por rango, se debe especificar:

1. Método de particionamiento: rango
2. Las columnas de particionamiento
3. Los límites de cada partición

El método de particionamiento y las columnas de particionamiento se definen por medio de la cláusula `PARTITION BY RANGE (column_list)` mientras que los límites de cada partición se definen a través de la cláusula `VALUES LESS THAN (value_list)`, donde `column_list` es una lista ordenada de las columnas que determinan la partición a la cual pertenece cada registro, es decir, las columnas de particionamiento, y `value_list` es una lista ordenada de valores para las columnas en `column_list`. Cada

valor debe ser una literal o cualquiera de las funciones `TO_DATE` o `RPAD` con argumentos constantes.

La cláusula `VALUES LESS THAN` especifica el límite superior no-inclusivo para cada partición. Todas las particiones, excepto la primera, tienen un valor menor especificado por `value_list` en la cláusula `VALUES LESS THAN` de la partición previa. Todas las llaves de particionamiento iguales a o mayores a este valor se asignan a la partición inmediatamente superior.

Si se define una partición con límite superior dado por la literal `MAXVALUE`, se dice que esta es la más grande. La literal `MAXVALUE` representa un valor virtual infinito superior a cualquier posible valor de la columna para la cual se define e incluye el valor nulo.

El siguiente ejemplo crea la tabla *ventas_r* particionada por rango a través de la columna *fecha*.

```
CREATE TABLE ventas_r (
   vendedor  NUMBER(5),
   fecha     DATE,
   monto     NUMBER(10,2)
)
PARTITION BY RANGE (fecha)
(
PARTITION ene2002 VALUES LESS THAN ('01/02/2002'),
PARTITION feb2002 VALUES LESS THAN ('01/03/2002'),
PARTITION mar2002 VALUES LESS THAN ('01/04/2002'),
PARTITION abr2002 VALUES LESS THAN ('01/05/2002'),
PARTITION may2002 VALUES LESS THAN ('01/06/2002'),
PARTITION jun2002 VALUES LESS THAN ('01/07/2002'),
PARTITION jul2002 VALUES LESS THAN ('01/08/2002'),
PARTITION ago2002 VALUES LESS THAN ('01/09/2002'),
PARTITION sep2002 VALUES LESS THAN ('01/10/2002'),
PARTITION oct2002 VALUES LESS THAN ('01/11/2002'),
PARTITION nov2002 VALUES LESS THAN ('01/12/2002'),
PARTITION dic2002 VALUES LESS THAN ('01/01/2003'),
PARTITION resto   VALUES LESS THAN (MAXVALUE)
);
```

Una venta realizada el 1 de marzo de 2002 tiene como llave de particionamiento '01/03/2002' y se mapea a la partición *mar2002*. Una venta efectuada el 15 de septiembre de 2003 tiene como llave de particionamiento '15/09/2003' y se mapea a la partición *resto*. Una venta para la cual no se haya registrado la fecha de venta tiene como llave de particionamiento '' (nulo) y se mapea a la partición *resto*.

El particionamiento por rango es un método conveniente para particionar tablas históricas. La mayoría de consultas que se hacen sobre tablas históricas extraen información por intervalos de tiempo. Al particionar la tabla por intervalos de tiempo las consultas se despacharán recorriendo una o más de las particiones, sin necesidad de barrer toda la tabla (a menos que el intervalo de tiempo abarque todos los datos almacenados). Al hecho de recorrer un subconjunto de las particiones para despachar una consulta se le denomina *recorte de particiones* y se considera como un método de optimización de consultas. Por ejemplo, para responder la pregunta "¿Cuánto se vendió en el primer bimestre de 2002?" únicamente se consultan las particiones *ene2002* y *feb2002*.

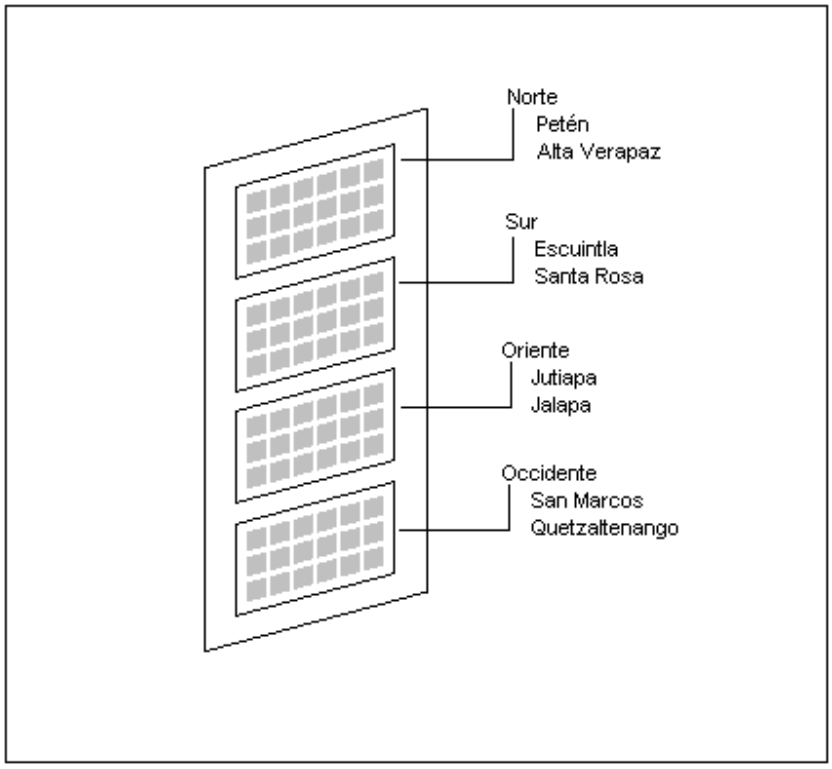
En la tabla *ventas_r*, tal como ocurre en las tablas históricas, los registros más recientes son los más utilizados, los registros más antiguos tienden a usarse sólo para consultas. Dado este comportamiento, no es necesario incluir en la estrategia de respaldo a aquellas particiones que únicamente se consulten, bastará con hacer respaldo de ellas una sola vez luego de que se hayan convertido en particiones de sólo lectura. Esto permite disminuir el tiempo y el espacio ocupado por los respaldos.

1.5.2 Particionamiento por lista

En el particionamiento por lista la tabla se divide por valores discretos de una sola de sus columnas. Los valores que definen cada partición se especifican por medio de literales, por ejemplo: 'NORTE', 'SUR', 'ESTE', 'OESTE'.

La figura 4 muestra una representación gráfica de una tabla particionada por lista.

Figura 4. Tabla particionada por lista



Distinto a como ocurre en el particionamiento por rango y dispersión, en el particionamiento por lista no se puede definir más de una columna de particionamiento, en otras palabras, la llave de particionamiento no puede ser compuesta.

Para crear una tabla particionada por lista se debe especificar:

1. Método de particionamiento: lista
2. La columna de particionamiento

3. La definición de cada partición, dada por una lista de valores discretos para la columna de particionamiento, estos valores califican a cada registro para ser incluido en la partición.

El siguiente ejemplo crea la tabla *ventas_l* particionada por el método de lista, cada partición agrupa un conjunto de departamentos geográficamente adyacentes. Note la existencia de la partición *otros* en la que se encuentran todos los departamentos que no se incluyan en alguna de las otras particiones.

```
CREATE TABLE ventas_l (
    vendedor     NUMBER(5),
    fecha        DATE,
    monto        NUMBER(10,2),
    departamento VARCHAR2(20)
)
PARTITION BY LIST (departamento)
(
PARTITION norte     VALUES ('Peten','Alta Verapaz'),
PARTITION sur       VALUES ('Escuintla','Santa Rosa'),
PARTITION centro    VALUES ('Guatemala','El Progreso'),
PARTITION oriente   VALUES ('Jutiapa','Jalapa'),
PARTITION occidente VALUES ('San Marcos','Quetzaltenango'),
PARTITION otros     VALUES (DEFAULT)
);
```

Una venta hecha en el departamento de San Marcos tiene como llave de particionamiento 'San Marcos' y se mapea a la partición *occidente*. Una venta hecha en el departamento de Santa Rosa tiene como llave de particionamiento 'Santa Rosa' y se mapea a la partición *sur*. Una venta hecha en el departamento de Izabal tiene como llave de particionamiento 'Izabal' y se mapea a la partición *otros*.

El método de particionamiento por lista está diseñado para modelar distribuciones de datos que siguen valores discretos. Por medio de este método de particionamiento los datos que no están relacionados ni ordenados pueden ser agrupados de manera natural.

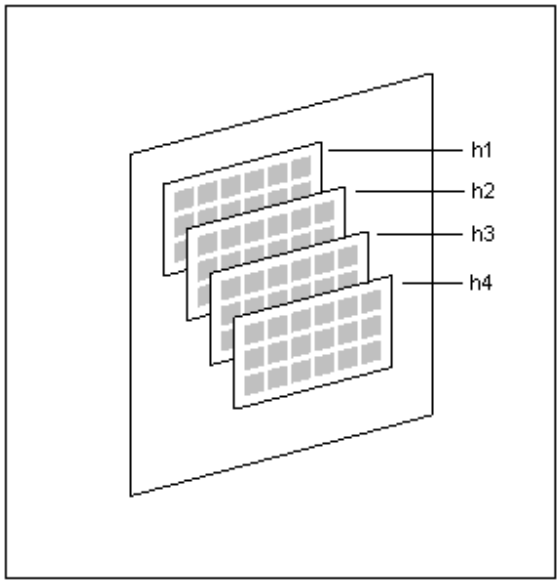
1.5.3 Particionamiento por dispersión

Al particionar una tabla por dispersión, los registros se mapean a una partición por medio de la aplicación de una función de dispersión a la llave de particionamiento. El algoritmo utilizado para calcular el valor de dispersión es propio del DBMS y no es modificable por lo que no se puede determinar a qué partición será asignado un registro dado.

El particionamiento por dispersión garantiza una distribución equitativa de los registros a través de todas las particiones pero, para lograr tal fin y debido a la naturaleza del mismo, es necesario que el número de particiones sea una potencia de dos, es decir: 2, 4, 8, 16, 32, 64, 128, etc.

La siguiente figura muestra una representación gráfica de una tabla particionada por dispersión.

Figura 5. Tabla particionada por dispersión



Para crear una tabla particionada por dispersión se debe especificar:

1. Método de particionamiento: dispersión
2. La(s) columna(s) de particionamiento
3. El número de particiones o la descripción individual para cada partición

En los siguientes ejemplos se crea la tabla *ventas_d* particionada por dispersión a través de la columna *vendedor*. En el primero de ellos las particiones se especifican por número (el nombre para cada una de ellas es asignado por el DBMS), mientras que en el segundo se especifican por nombre.

```
CREATE TABLE ventas_d (
   vendedor     NUMBER(5),
   fecha        DATE,
   monto        NUMBER(10,2),
   departamento VARCHAR2(20)
)
PARTITION BY HASH (vendedor)
PARTITIONS 4;
```

```
CREATE TABLE ventas_d (
   vendedor     NUMBER(5),
   fecha        DATE,
   monto        NUMBER(10,2),
   departamento VARCHAR2(20)
)
PARTITION BY HASH (vendedor)
(
PARTITION h1,
PARTITION h2,
PARTITION h3,
PARTITION h4
);
```

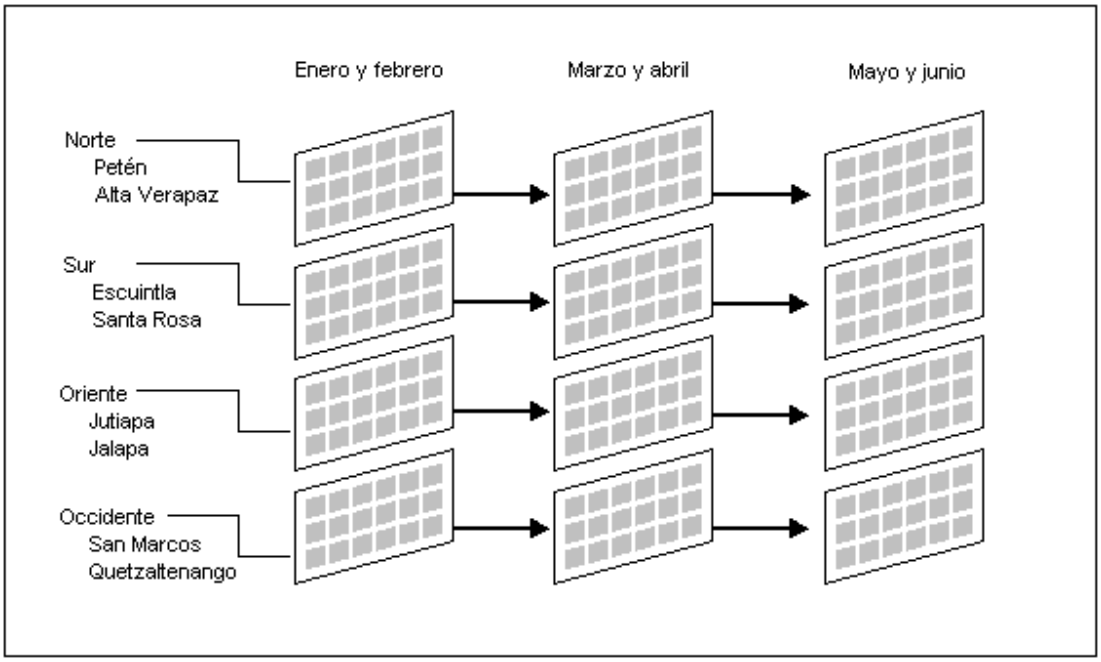
El particionamiento por dispersión se utiliza comúnmente para dividir tablas grandes no históricas en segmentos más pequeños y aumentar con ello el rendimiento y la disponibilidad de las mismas. Esta división puede hacerse de forma equitativa a través de las particiones las que a su vez pueden ser distribuidas en discos distintos para aprovechar el procesamiento paralelo de las mismas.

1.5.4 Particionamiento por rango-lista

En el particionamiento por rango-lista se crean dos niveles de particionamiento. El primer nivel de particionamiento se basa en rangos de valores, como ocurre en el particionamiento por rango; el segundo nivel se basa en valores discretos, como ocurre en el particionamiento por lista. Esta forma de particionamiento compuesto resulta muy útil para datos históricos, pero permite además agrupar registros basados en valores de columnas no relacionados o no ordenados.

La siguiente figura muestra una representación gráfica de una tabla particionada por rango-lista.

Figura 6. Tabla particionada por rango-lista



Para crear una tabla particionada por rango-lista se debe especificar:

1. Método de particionamiento: rango
2. La(s) columna(s) de particionamiento
3. Método de subparticionamiento: lista
4. La columna de subparticionamiento
5. La descripción de cada partición, indicando los límites de las mismas
6. La definición de cada subpartición, dada por una lista de valores discretos para la columna de particionamiento, estos valores califican a cada registro para ser incluido en la partición. El nombre de la subpartición debe ser único a nivel de tabla, es decir, en una misma tabla no pueden existir dos subparticiones con el mismo nombre.

En el siguiente ejemplo se crea la tabla *ventas_rl* particionada por rango-lista: para cada trimestre del año 2002 (y los años restantes) se crean cinco regiones de venta y una extra para los departamentos no listados. Esto permite agrupar las ventas por trimestre y dentro de cada trimestre agruparlas por región.

```
CREATE TABLE ventas_rl (
   vendedor     NUMBER(5),
   fecha        DATE,
   monto        NUMBER(10,2),
   departamento  VARCHAR2(20)
)
PARTITION BY RANGE (fecha)
SUBPARTITION BY LIST (departamento)
(
PARTITION primero2002 VALUES LESS THAN ('01/04/2002')
(
  SUBPARTITION p2002norte     VALUES ('Peten','Alta Verapaz'),
  SUBPARTITION p2002sur       VALUES ('Escuintla','Santa Rosa'),
  SUBPARTITION p2002centro    VALUES ('Guatemala','El Progreso'),
  SUBPARTITION p2002oriente   VALUES ('Jutiapa','Jalapa'),
  SUBPARTITION p2002occidente VALUES ('San Marcos','Quetzaltenango'),
  SUBPARTITION p2002otros     VALUES (DEFAULT)
),
```

```
PARTITION segundo2002 VALUES LESS THAN ('01/07/2002')
(
  SUBPARTITION s2002norte     VALUES ('Peten','Alta Verapaz'),
  SUBPARTITION s2002sur       VALUES ('Escuintla','Santa Rosa'),
  SUBPARTITION s2002centro    VALUES ('Guatemala','El Progreso'),
  SUBPARTITION s2002oriente   VALUES ('Jutiapa','Jalapa'),
  SUBPARTITION s2002occidente VALUES ('San Marcos','Quetzaltenango'),
  SUBPARTITION s2002otros     VALUES (DEFAULT)
),
PARTITION tercero2002 VALUES LESS THAN ('01/10/2002')
(
  SUBPARTITION t2002norte     VALUES ('Peten','Alta Verapaz'),
  SUBPARTITION t2002sur       VALUES ('Escuintla','Santa Rosa'),
  SUBPARTITION t2002centro    VALUES ('Guatemala','El Progreso'),
  SUBPARTITION t2002oriente   VALUES ('Jutiapa','Jalapa'),
  SUBPARTITION t2002occidente VALUES ('San Marcos','Quetzaltenango'),
  SUBPARTITION t2002otros     VALUES (DEFAULT)
),
PARTITION cuarto2002 VALUES LESS THAN ('01/01/2003')
(
  SUBPARTITION c2002norte     VALUES ('Peten','Alta Verapaz'),
  SUBPARTITION c2002sur       VALUES ('Escuintla','Santa Rosa'),
  SUBPARTITION c2002centro    VALUES ('Guatemala','El Progreso'),
  SUBPARTITION c2002oriente   VALUES ('Jutiapa','Jalapa'),
  SUBPARTITION c2002occidente VALUES ('San Marcos','Quetzaltenango'),
  SUBPARTITION c2002otros     VALUES (DEFAULT)
),
PARTITION resto VALUES LESS THAN (MAXVALUE)
(
  SUBPARTITION rnorte      VALUES ('Peten','Alta Verapaz'),
  SUBPARTITION rsur        VALUES ('Escuintla','Santa Rosa'),
  SUBPARTITION rcentro     VALUES ('Guatemala','El Progreso'),
  SUBPARTITION roriente    VALUES ('Jutiapa','Jalapa'),
  SUBPARTITION roccidente  VALUES ('San Marcos','Quetzaltenango'),
  SUBPARTITION rotros      VALUES (DEFAULT)
)
);
```

Para mapear un registro hacia una subpartición primero se establece a cuál de las particiones debe ir el registro (mapeo por rango), una vez identificada la partición de primer nivel, se establece a cuál de las subparticiones dentro de esa partición debe ir el registro (mapeo por lista).

Por ejemplo, una venta realizada el 1 de abril de 2002 en el departamento de Alta Verapaz tiene como llave de particionamiento ('01/04/2002','Alta Verapaz') por lo que se asigna a la subpartición *s2002norte* de la partición *segundo2002*.

1.5.5 Particionamiento por rango-dispersión

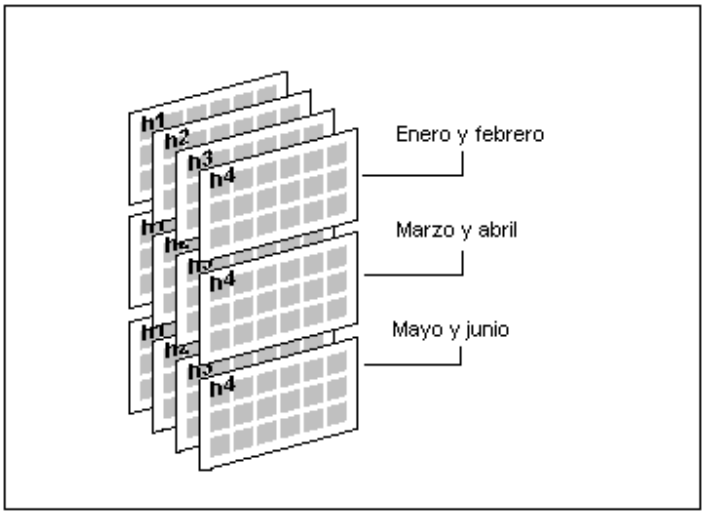
Este método de particionamiento combina las ventajas del crecimiento fijo de las particiones por rango con la distribución equitativa y el acceso paralelo del particionamiento por dispersión.

El crecimiento fijo de las particiones por rango se refiere al hecho de que una partición que almacena datos históricos no seguirá aumentando su tamaño una vez se haya alcanzado el límite superior de tiempo definido para la partición, por ejemplo, si se define una partición que almacene las ventas de enero de 2002, ésta dejará de recibir registros una vez alcanzado el 1 de febrero de 2002, por supuesto, esto es válido solo para límites definidos por fechas.

En el particionamiento rango-dispersión, el nivel de particionamiento es de tipo rango (por lo que se definen límites para cada partición) mientras que el nivel de subparticionamiento es de tipo dispersión (por lo que se define un número puntual de particiones).

La siguiente figura muestra una representación gráfica de una tabla particionada por rango-dispersión.

Figura 7. Tabla particionada por rango-dispersión



Para crear una tabla particionada por rango-dispersión se debe especificar:

1. Método de particionamiento: rango
2. La(s) columna(s) de particionamiento
3. Método de subparticionamiento: dispersión
4. La(s) columna(s) de subparticionamiento
5. La descripción de cada partición, indicando los límites de las mismas
6. El número de subparticiones o la descripción individual para cada partición. Al igual que sucede en el particionamiento rango-lista, las subparticiones deben tener un nombre único a nivel de la tabla.

En el siguiente ejemplo se crea la tabla *ventas_rh* particionada por rango-dispersión: para cada trimestre del año 2002 (y los años restantes) se crean cuatro particiones para distribuir entre ellas a los vendedores.

```
CREATE TABLE ventas_rh (
    vendedor      NUMBER(5),
    fecha         DATE,
    monto         NUMBER(10,2),
   departamento  VARCHAR2(20)
)
PARTITION BY RANGE (fecha)
SUBPARTITION BY HASH (vendedor)
(
PARTITION primero2002 VALUES LESS THAN ('01/04/2002')
(
  SUBPARTITION p2002h1, SUBPARTITION p2002h2,
  SUBPARTITION p2002h3, SUBPARTITION p2002h4
),
PARTITION segundo2002 VALUES LESS THAN ('01/07/2002')
(
  SUBPARTITION s2002h1, SUBPARTITION s2002h2,
  SUBPARTITION s2002h3, SUBPARTITION s2002h4
),
PARTITION tercero2002 VALUES LESS THAN ('01/10/2002')
(
  SUBPARTITION t2002h1, SUBPARTITION t2002h2,
  SUBPARTITION t2002h3, SUBPARTITION t2002h4
),
PARTITION cuarto2002 VALUES LESS THAN ('01/01/2003')
(
  SUBPARTITION c2002h1, SUBPARTITION c2002h2,
  SUBPARTITION c2002h3, SUBPARTITION c2002h4
),
PARTITION resto VALUES LESS THAN (MAXVALUE)
(
  SUBPARTITION r2002h1, SUBPARTITION r2002h2,
  SUBPARTITION r2002h3, SUBPARTITION r2002h4
)
);
```

Para mapear un registro hacia una subpartición primero se establece a cuál de las particiones debe ir el registro (mapeo por rango), una vez identificada la partición de primer nivel, la elección de la subpartición dependerá del DBMS a través de la utilización de la función de dispersión.

Por ejemplo, una venta realizada el 1 de abril de 2002 por el vendedor cuyo código sea el número cinco, tiene como llave de particionamiento ('01/04/2002', 5) por lo que se asigna a la partición *segundo2002* y dentro de ella a la subpartición elegida por el DBMS.

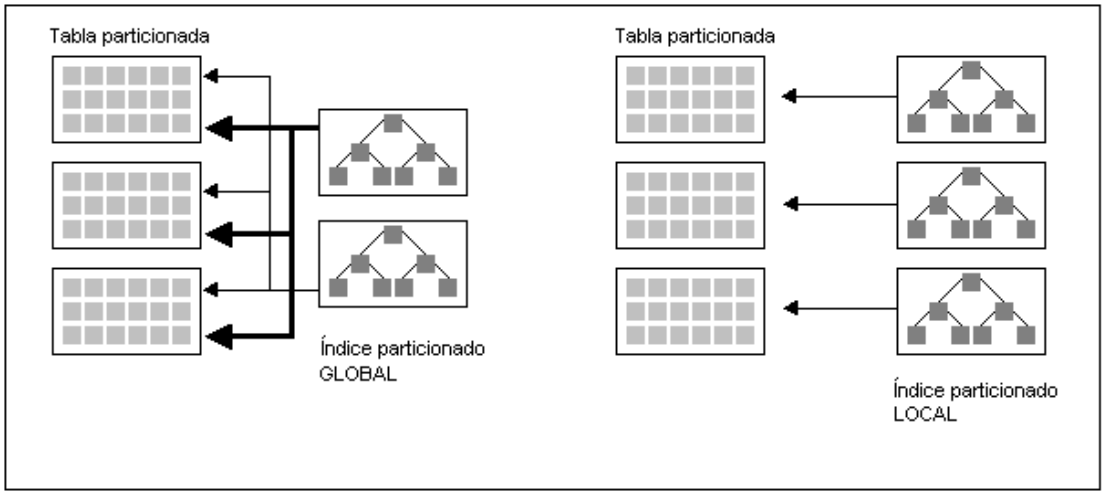
2. ÍNDICES PARTICIONADOS

2.1 Definición

Al igual que una tabla particionada, un índice particionado es un índice dividido en segmentos más pequeños y manejables llamados particiones.

Cuando el particionamiento del índice es independiente del particionamiento de la tabla se le denomina índice particionado global (o índice global). Cuando el índice está ligado al método de particionamiento de la tabla se denomina índice particionado local (o índice local). La siguiente figura muestra una representación gráfica de esta clasificación.

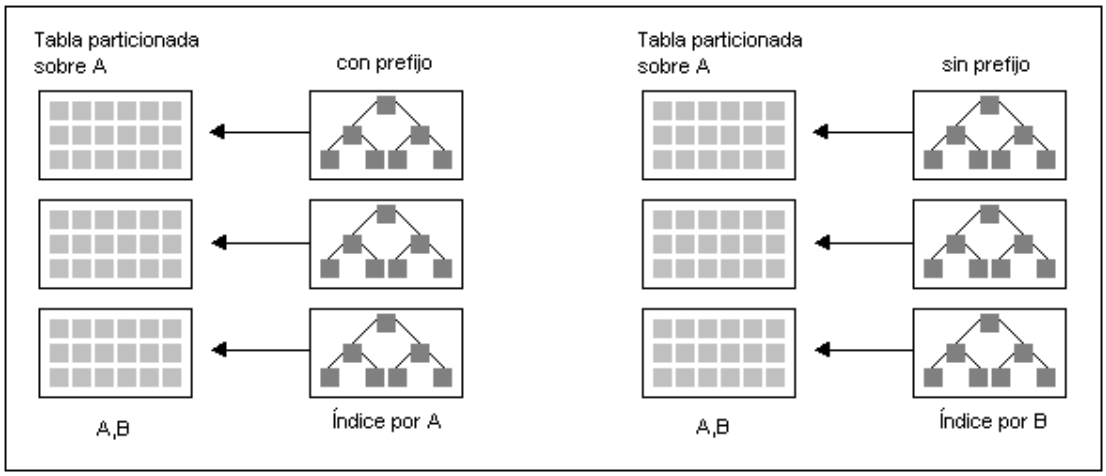
Figura 8. Representación gráfica de un índice global y un índice local



Tanto los índices globales como los locales pueden ser "con prefijo" o "sin prefijo". Si la llave de particionamiento del índice representa un prefijo de la llave del mismo, se dice que el índice es "con prefijo". Cuando la llave de particionamiento del índice no representa un prefijo de la llave, se dice que el índice es "sin prefijo".

La clasificación del índice como "con prefijo" o "sin prefijo" no se especifica al momento de crear el índice, viene como consecuencia de la relación existente entre la llave de particionamiento y la llave del índice. La siguiente figura muestra esta clasificación.

Figura 9. Representación gráfica de un índice con prefijo y uno sin prefijo



En las siguientes secciones se tratan los índices globales y los locales incluyendo ejemplos de creación de cada uno de ellos.

2.2 Índices particionados globales

Un índice particionado de tipo global es aquel cuyas particiones no están asociadas a una partición correspondiente dentro de la tabla para la cual se crea. Este tipo de índice puede crearse tanto para tablas particionadas como para tablas no particionadas.

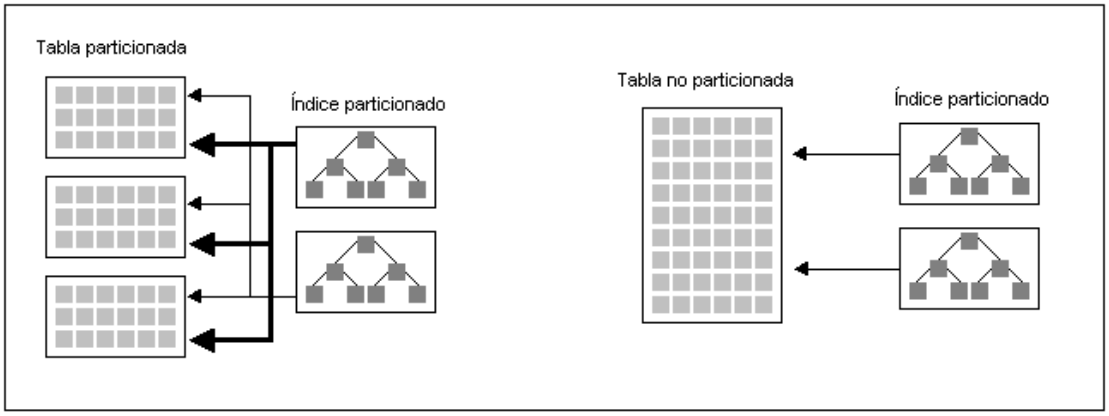
2.2.1 Índices particionados globales con prefijo

Las características de este tipo de índices globales incluyen:

1. Sólo pueden ser particionados por rango.
2. La partición mayor debe incluir la cláusula `MAXVALUE` para contener cualquier valor posible de la llave del índice.
3. Pueden crearse tanto para tablas particionadas como para tablas no particionadas.
4. No se requiere relación entre el particionamiento del índice y el particionamiento de la tabla.
5. Pueden ser únicos (llaves no repetidas) o no únicos (llaves repetidas).

La figura 10 muestra una representación gráfica de un índice particionado de tipo global con prefijo definido tanto para una tabla particionada como para una no particionada. Como puede apreciarse, en el caso de la tabla particionada, cada partición del índice referencia a más de una partición de la tabla.

Figura 10. Representación gráfica de un índice particionado de tipo global con prefijo



A continuación se muestra un ejemplo de creación de un índice particionado de tipo global con prefijo.

Si se tiene la tabla *empleados* (no particionada), creada de la siguiente manera:

```
CREATE TABLE empleados (
  id          NUMBER(5),
  nombres     VARCHAR2(60),
  apellidos   VARCHAR2(60),
  direccion   VARCHAR2(100),
  telefono    VARCHAR2(10)
);
```

Entonces, se puede crear un índice particionado global sobre la columna nombres de la siguiente manera:

```
CREATE INDEX indice_global ON empleados (nombres) GLOBAL
PARTITION BY RANGE (nombres)
(   PARTITION part1 VALUES LESS THAN ('H'),
    PARTITION part2 VALUES LESS THAN (MAXVALUE)
);
```

2.2.2 Índices particionados globales sin prefijo

Un índice global sin prefijo no tiene beneficios en el manejo y el rendimiento comparado con un índice no particionado por lo que su implementación carece de sentido.

2.3 Índices particionados locales

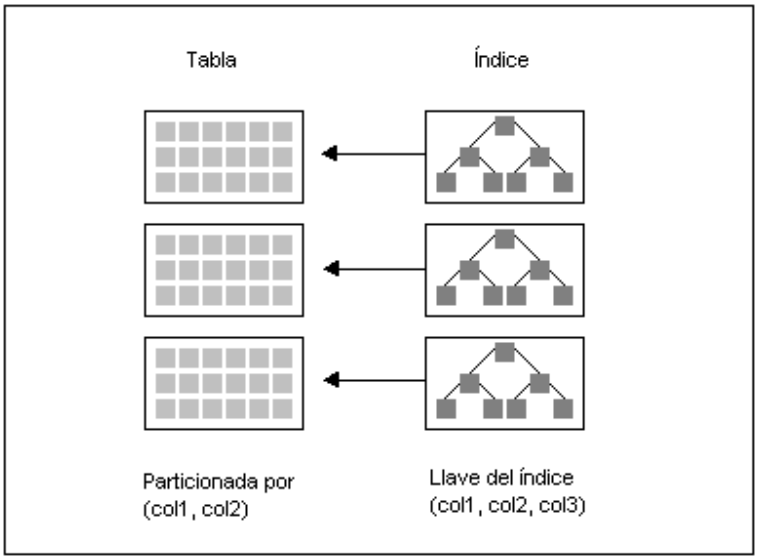
Un índice particionado de tipo local sólo puede crearse para tablas particionadas. Cada partición del índice está asociada a una partición correspondiente dentro de la tabla, es decir, cada partición del índice se trata como un índice de su partición de tabla correspondiente.

El método de particionamiento del índice es el mismo que el definido para la tabla por lo que no se incluye en la cláusula de creación. Cuando se crean para una tabla subparticionada, cada partición del índice corresponde a una subpartición, no existe una partición de índice para las particiones de tabla de primer nivel.

2.3.1 Índices particionados locales con prefijo

En este tipo de índice la llave de particionamiento (que es la misma que la de la tabla) es un prefijo de la llave del índice tal como lo muestra la siguiente figura.

Figura 11. Índice particionado de tipo local con prefijo



Un índice local con prefijo puede crearse para cualquiera de los cinco tipos de tablas particionadas además, pueden ser únicos y no únicos.

Para la tabla *empleados* (particionada) creada de la siguiente manera:

```
CREATE TABLE empleados (
  id          NUMBER(5),
  nombres     VARCHAR2(60),
  apellidos   VARCHAR2(60),
  direccion   VARCHAR2(100),
  telefono    VARCHAR2(10)
)
PARTITION BY RANGE (nombres)
(   PARTITION pa VALUES LESS THAN ('B'),
    PARTITION pb VALUES LESS THAN ('C'),
    ...
    PARTITION pz VALUES LESS THAN (MAXVALUE)
);
```

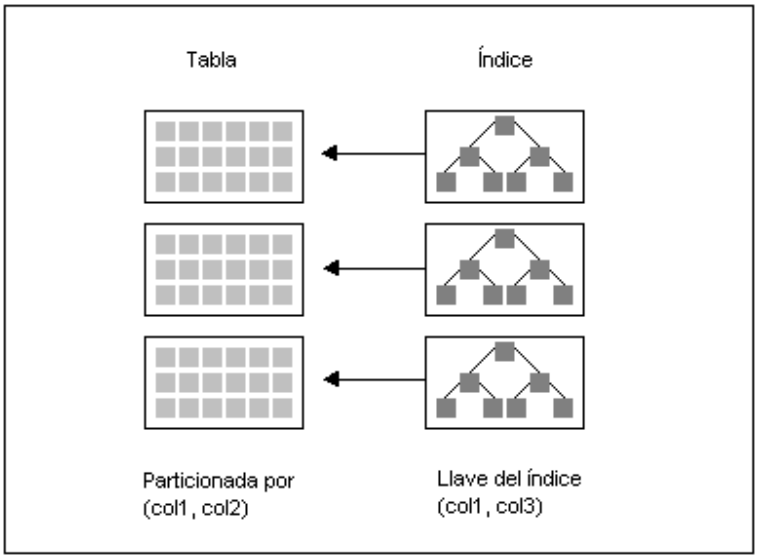
Se puede definir un índice local con prefijo de la siguiente forma:

```
CREATE INDEX indice_local1 ON empleados (nombres, apellidos) LOCAL
(   PARTITION part1,
    PARTITION part2,
    PARTITION part3
);
```

2.3.2 Índices particionados locales sin prefijo

En este tipo de índice la llave de particionamiento no es un prefijo de la llave del índice, tal como lo muestra la figura 12.

Figura 12. Índice particionado de tipo local con prefijo



Para la tabla empleados (particionada) creada de la siguiente manera:

```
CREATE TABLE empleados (
  id          NUMBER(5),
  nombres     VARCHAR2(60),
  apellidos   VARCHAR2(60),
  direccion   VARCHAR2(100),
```

```
    telefono    VARCHAR2(10)
  )
  PARTITION BY RANGE (nombres)
  (   PARTITION pa VALUES LESS THAN ('B'),
      PARTITION pb VALUES LESS THAN ('C'),
      ...
      PARTITION pz VALUES LESS THAN (MAXVALUE)
  );
```

se puede crear el siguiente índice como particionado local sin prefijo:

```
CREATE INDEX indice_local2 ON empleados (apellidos) LOCAL
(   PARTITION part1,
    PARTITION part2,
    PARTITION part3
);
```

Los índices locales sin prefijo pueden ser no únicos. Si la llave de particionamiento es un subconjunto de la llave del índice, entonces el índice local sin prefijo puede ser único.

Cuando una consulta hecha sobre la tabla involucra a su llave de particionamiento, entonces el *recorte de particiones* puede ser utilizado para despachar la consulta. Si la consulta involucra las mismas columnas que la llave del índice, entonces todas las particiones del índice deben ser utilizadas, porque las particiones del índice pueden contener valores iguales de llave (debe recordarse que en este tipo de índice cada partición de él se trata como un índice propio para la partición de tabla correspondiente).

Sin embargo, si la consulta involucra la llave de particionamiento y la llave del índice, entonces solamente las particiones de índice correspondientes a la llave de particionamiento son utilizadas para despachar la consulta.

3. MANTENIMIENTO DE TABLAS E ÍNDICES PARTICIONADOS

Una vez creadas las tablas e índices particionados, surge la necesidad de realizar operaciones de mantenimiento sobre los mismos. Las operaciones de mantenimiento incluyen:

1. Renombrar particiones
2. Agregar particiones
3. Borrar particiones
4. Dividir particiones
5. Combinar particiones
6. Mezclar particiones
7. La reconstrucción de índices

Muchas de estas operaciones realizadas sobre las tablas invalidan los índices no particionados y los índices globales asociados a ellas, esto produce la necesidad de reconstruir el índice luego de aplicar la operación. Para evitar esto y dejar el índice como válido puede agregarse la cláusula `UPDATE GLOBAL INDEXES` a la sentencia que ejecuta la operación en la tabla.

Esto acarrea los siguientes beneficios:

1. El índice se actualiza al mismo tiempo que la operación de mantenimiento se ejecuta por lo que se hace innecesario reconstruir el índice al terminar dicha operación.
2. Existe mayor disponibilidad de índices (no particionados y globales) ya que no se marcan como inutilizables. El índice permanece disponible aún cuando se está

ejecutando la operación por lo que puede utilizarse para acceder a otras particiones en la tabla.

3. Ya no es necesario buscar los nombres de los índices globales como inutilizables para reconstruirlos.

Sin embargo, es necesario tomar en cuenta que la operación de mantenimiento tomará más tiempo que si no se incluyera la cláusula. Como regla general, es más rápido actualizar el índice que reconstruirlo cuando el tamaño de la partición afectada del índice es menor al 5% del tamaño de la tabla.

Las siguientes secciones tratan las operaciones de mantenimiento que pueden realizarse sobre cada tipo de tabla e índice así como las consecuencias que el aplicar la operación tiene sobre los índices asociados a la tabla, sean estos globales, locales o no particionados.

3.1 Mantenimiento de tablas particionadas por rango

3.1.1 Cómo renombrar una partición

Cambiar el nombre de una partición en una tabla particionada por rango se hace de la siguiente manera:

```
ALTER TABLE ventas_r RENAME PARTITION resto TO ene2003;
```

donde `resto` es el nombre actual de la partición y `ene2003` es el nuevo nombre para la partición.

Los índices locales, globales y no particionados no se ven afectados por el cambio de nombre.

3.1.2 Cómo agregar una nueva partición

A una tabla particionada por rango se le puede agregar una partición al final del conjunto de particiones siempre que el límite superior de la última partición no sea `MAXVALUE`. Por ejemplo, si la tabla *ventas_r* tiene definidas doce particiones, una para cada mes del año 2002, siendo el límite superior de la última partición la fecha '01/01/2003' entonces se puede agregar una partición que contenga los registros de ventas para el mes de enero del año 2003 de la siguiente manera:

```
ALTER TABLE ventas_r
ADD PARTITION ene2003 VALUES LESS THAN ('01/02/2003');
```

Si la última partición tiene como límite superior `MAXVALUE`, entonces no se puede agregar una nueva partición, en su lugar, la última partición debe ser dividida en dos particiones tal como se muestra en la sección 3.1.4.

No se pueden agregar particiones al inicio o en medio del conjunto de particiones para la tabla, únicamente se pueden agregar al final. Si se desea colocar una nueva partición al inicio o en medio del conjunto de particiones, entonces debe utilizarse la división de particiones tal como se explica en la sección 3.1.4.

Los índices locales, globales y no particionados asociados a la tabla permanecen utilizables.

3.1.3 Cómo borrar una partición

La tabla de la cual se desea eliminar la partición puede tener asociados índices no particionados, índices globales o índices locales. Los índices no particionados y todas las particiones de los índices globales se marcan como no utilizables y deben ser

reconstruidos. Los índices locales son automáticamente actualizados. Existen tres formas de eliminar una partición, a continuación se explica cada uno de ellos:

Método 1

Se borra la partición y se actualizan los índices, por ejemplo:

```
ALTER TABLE ventas_r DROP PARTITION ene2002;
```

Un índice no particionado se actualiza de la siguiente manera:

```
ALTER INDEX ventas_r_vendedor_ix REBUILD;
```

Un índice global no puede ser actualizado por completo, debe actualizarse partición por partición de la siguiente manera:

```
ALTER INDEX ventas_r_vendedor_ix REBUILD PARTITION ene2002;
ALTER INDEX ventas_r_vendedor_ix REBUILD PARTITION feb2002;
...
ALTER INDEX ventas_r_vendedor_ix REBUILD PARTITION dic2002;
```

Este método es el más apropiado para tablas grandes en las que el tamaño de la partición a borrar representa un alto porcentaje del total de registros de la tabla.

Método 2

Se borran los registros de la partición y luego se elimina la partición, este método deja utilizables tanto los índices no particionados como los índices globales por lo que se hace innecesaria su recreación:

```
DELETE FROM ventas_r WHERE fecha BETWEEN '01/01/2002' AND '31/01/2002';
ALTER TABLE ventas_r DROP PARTITION ene2002;
```

Este método es más apropiado para tablas pequeñas, o para tablas grandes en las que el tamaño de la partición a eliminar representa un porcentaje pequeño del total de datos de la tabla.

Método 3

Incluir la cláusula `UPDATE GLOBAL INDEXES` para que los índices globales y los no particionados se actualicen conforme se borra la partición:

```
ALTER TABLE ventas_r DROP PARTITION ene2002 UPDATE GLOBAL INDEXES;
```

3.1.4 Cómo dividir una partición

Una partición de una tabla particionada por rango puede ser dividida en dos particiones. Al dividir una partición en dos, se crean dos nuevas particiones y la original es eliminada.

Las particiones afectadas de los índices locales se marcan como no utilizables y deben ser reconstruidas. Los índices no particionados y todas las particiones de los índices globales se marcan como no utilizables y deben ser reconstruidos, a menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES` o que la partición dividida no tenga datos. En el siguiente ejemplo la partición *ene2002* se divide en dos particiones, la primera almacena las ventas anteriores al 1 de enero del 2002 y la segunda las ventas durante enero de 2002.

```
ALTER TABLE ventas_r
SPLIT PARTITION ene2002 AT ('01/01/2002') INTO
(
  PARTITION dic2001,
  PARTITION ene2002
);
```

El valor definido como límite (1 de enero de 2002 en el ejemplo) se incluye en la segunda partición.

3.1.5 Cómo combinar dos particiones

Se pueden juntar dos particiones adyacentes de una tabla particionada por rango, si las particiones no son adyacentes esta operación de mantenimiento no puede realizarse. La nueva partición toma como límite superior el definido para la segunda partición.

Las particiones afectadas de los índices locales se marcan como no utilizables y deben ser reconstruidas, los índices no particionados y todas las particiones de los índices globales también se marcan como no utilizables y deben ser reconstruidos a menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES`.

En el siguiente ejemplo las particiones de enero y febrero de 2002 se combinan para formar una nueva partición que representa el primer bimestre de ventas del año 2002:

```
ALTER TABLE ventas_r
MERGE PARTITIONS ene2002, feb2002 INTO PARTITION bimestre1_2002;
```

3.2 Mantenimiento de tablas particionadas por lista

3.2.1 Cómo renombrar una partición

Una partición de una tabla particionada por lista puede renombrarse de la siguiente manera:

```
ALTER TABLE ventas_l RENAME PARTITION oriente TO este;
```

Los índices no particionados, globales y locales no se ven afectados al renombrar la partición.

3.2.2 Cómo agregar una nueva partición

Puede agregarse una partición a una tabla particionada por lista siempre que su valor de particionamiento no exista ya en alguna de las particiones, además, la tabla no debe tener una partición cuyo valor definido sea `DEFAULT`, en cuyo caso debe usarse la división de particiones (sección 3.2.4). En el siguiente ejemplo se agrega una partición a la tabla de ventas (que no incluye una partición `DEFAULT`) que contendrá el registro de las ventas hechas a nivel centroamericano sin incluir a Guatemala:

```
ALTER TABLE ventas_l
ADD PARTITION centroamerica
VALUES ('El Salvador','Honduras','Nicaragua','Costa Rica',
        'Panama','Belice');
```

Los índices no particionados, globales y locales asociados a la tabla permanecen utilizables.

3.2.3 Cómo borrar una partición

Una partición de una tabla particionada por lista puede borrarse tal como se indica en la sección 3.1.3.

3.2.4 Cómo dividir una partición

Al dividir una partición de una tabla particionada por lista, se crean dos particiones: la primera incluye los valores especificados en la operación de división y la segunda conserva los valores restantes de la partición original. A continuación se muestra un ejemplo en el que la partición *norte* se divide en dos nuevas particiones, la partición

norte_calor que incluye al departamento de Petén y la partición *norte_templado* que incluye al departamento de Alta Verapaz.

```
ALTER TABLE ventas_l
SPLIT PARTITION norte VALUES ('Peten') INTO
(
  PARTITION norte_calor,
  PARTITION norte_templado
);
```

Cuando la partición dividida es la `DEFAULT`, la primera partición incluye los valores especificados en la operación de división y la segunda se convierte en la partición `DEFAULT`. En el siguiente ejemplo la partición `DEFAULT` de la tabla *ventas_l* se divide en una nueva partición llamada *centroamerica* y otra que es la nueva partición `DEFAULT`.

```
ALTER TABLE ventas_l
SPLIT PARTITION otros VALUES
('El Salvador','Honduras','Nicaragua','Costa Rica','Panama','Belice')
INTO
(
  PARTITION centroamerica,
  PARTITION otros
);
```

Las particiones correspondientes a los índices locales se marcan como no utilizables y deben ser reconstruidas. Los índices no particionados y todas las particiones de los índices globales se marcan como no utilizables y deben ser reconstruidos a menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES`.

3.2.5 Cómo combinar dos particiones

Dos particiones de una tabla particionada por lista pueden ser combinadas para formar una sola sin que sean adyacentes, contrario a como ocurre en el caso del particionamiento por rango, esto debido a que el particionamiento por lista no guarda las

particiones por orden. Cuando una de las particiones combinadas es la `DEFAULT`, la nueva partición se convierte en la `DEFAULT`. A continuación se muestra un ejemplo:

```
ALTER TABLE ventas_l
MERGE PARTITIONS sur, occidente
INTO PARTITION sur_occidente;
```

Las particiones *sur* y *occidente* dejan de existir y surge una nueva llamada *sur_occidente*.

Las particiones correspondientes a los índices locales se marcan como no utilizables y deben ser reconstruidas. Los índices no particionados y todas las particiones de los índices globales se marcan como no utilizables y deben ser reconstruidos a menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES`.

3.3 Mantenimiento de tablas particionadas por dispersión

3.3.1 Cómo renombrar una partición

Cambiar el nombre de una partición en una tabla particionada por dispersión se hace de la siguiente manera:

```
ALTER TABLE ventas_h RENAME PARTITION h1 TO dispersion1;
```

donde *h1* es el nombre actual de la partición y *dispersion1* es el nuevo nombre para la partición.

Los índices locales, globales y no particionados no se ven afectados por el cambio de nombre.

3.3.2 Cómo agregar una nueva partición

Cuando se agrega una nueva partición a una tabla particionada por dispersión, la nueva partición es llenada con registros provenientes de otras particiones sin que estas últimas se queden sin registros. El siguiente ejemplo agrega la partición *h5* a la tabla *ventas_h*:

```
ALTER TABLE ventas_h
ADD PARTITION h5;
```

Las particiones correspondientes a los índices locales se marcan como no utilizables y deben ser reconstruidas. Los índices no particionados y todas las particiones de los índices globales se marcan como no utilizables y deben ser reconstruidos a menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES`.

3.3.3 Cómo borrar una partición

A diferencia de una partición por rango o por lista, una partición tipo dispersión no puede ser borrada, en su lugar debe utilizarse la mezcla de particiones explicada en la sección 3.3.4.

3.3.4 Cómo mezclar una partición

La mezcla de una partición es una manera de reducir el número de particiones en una tabla particionada por dispersión. Cuando una partición es mezclada, su contenido es redistribuido en las particiones restantes (particiones absorbentes) y la partición es eliminada.

Cada partición de índice local correspondiente a la partición mezclada es borrada. Las particiones de índice local correspondientes a una o más de las particiones absorbentes se marcan como no utilizables y deben ser reconstruidas.

A menos que se especifique `UPDATE GLOBAL INDEXES`, todos los índices no particionados, o todas las particiones de los índices globales particionados son marcados como no utilizables y deben ser reconstruidos.

A continuación se muestra un ejemplo:

```
ALTER TABLE ventas_h COALESCE PARTITION;
```

Tanto la partición a mezclar como las particiones absorbentes son elegidas por el DBMS.

3.4 Mantenimiento de tablas particionadas por rango-lista

3.4.1 Cómo renombrar una partición o subpartición

Una partición dentro de una tabla particionada por rango-lista puede ser renombrada de la siguiente manera:

```
ALTER TABLE ventas_rl
RENAME PARTITION primero2002 TO trimestre1_2002;
```

En este ejemplo se cambia el nombre de la partición *primero2002* a *trimestre1_2002*.

A su vez, una subpartición puede ser renombrada de la siguiente forma:

```
ALTER TABLE ventas_rl
RENAME SUBPARTITION p2002otros TO p2002resto_del_pais;
```

Aquí, el nombre de la subpartición *p2002otros* es cambiado a *p2002resto_del_pais*.

Los índices asociados a la tabla o sus subparticiones siguen siendo utilizables.

3.4.2 Cómo agregar una partición o subpartición

Una partición (tipo rango) puede ser agregada a una tabla particionada por rango-lista de la siguiente manera:

```
ALTER TABLE ventas_rl
ADD PARTITION primero2003 VALUES LESS THAN ('01/04/2003')
(
  SUBPARTITION p2003norte     VALUES ('Peten','Alta Verapaz'),
  SUBPARTITION p2003sur       VALUES ('Escuintla','Santa Rosa'),
  SUBPARTITION p2003centro    VALUES ('Guatemala','El Progreso'),
  SUBPARTITION p2003oriente   VALUES ('Jutiapa','Jalapa'),
  SUBPARTITION p2003occidente VALUES ('San Marcos','Quetzaltenango'),
);
```

En este ejemplo, la tabla *ventas* no cuenta con una partición con valor definido como `MAXVALUE` por lo que se puede agregar una partición al final del conjunto. La partición nueva almacena los registros de ventas realizadas en el primer trimestre del año 2003 y dicha partición se subparticiona por lista utilizando los mismos criterios de agrupamiento que las subparticiones ya existentes, es decir, por región de venta.

En el siguiente ejemplo se agrega una subpartición a la partición *primero2003* para que se incluya a El Salvador como región de venta:

```
ALTER TABLE ventas_rl
MODIFY PARTITION primero2003
ADD SUBPARTITION p2003elsalvador VALUES ('El Salvador');
```

Los índices locales y globales asociados a la tabla se mantienen utilizables.

3.4.3 Cómo borrar una partición o subpartición

Una partición puede eliminarse de la siguiente forma:

```
ALTER TABLE ventas_rl
DROP PARTITION primero2003;
```

De esta manera se elimina la partición *primero2003* de la tabla *ventas_rl* y todas sus subparticiones. Una subpartición puede eliminarse de la manera siguiente:

```
ALTER TABLE ventas_rl
DROP SUBPARTITION p2003elsalvador;
```

Si existen índices locales para la partición o subpartición eliminada, las particiones correspondientes también son eliminadas. Los índices no particionados, o todas las particiones de los índices globales son marcadas como no utilizables y deben ser reconstruidas a menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES` al momento de borrar la partición o subpartición.

3.4.4 Cómo dividir una partición o subpartición

Cuando se divide una partición o subpartición en dos, la partición o subpartición dividida es borrada y se crean dos nuevas particiones o subparticiones. Cuando la división se hace en el primer nivel de particionamiento, las nuevas particiones heredan la descripción de subparticionamiento de la partición de la cual fueron generadas.

En el siguiente ejemplo se muestra cómo dividir la partición por rango en dos particiones. Recuérdese que una tabla particionada por rango-lista tiene particiones por rango en el primer nivel de particionamiento y particiones por lista en el segundo. En el ejemplo, la partición *primero2002* que almacena los registros de ventas del primer trimestre del año 2002 se divide en dos particiones, estas son, *primero1_2002* que

almacena las ventas realizadas del 1 de enero de 2002 al 14 de febrero del mismo año y *primero2_2002* que almacena las ventas realizadas del 15 de febrero de 2002 al 31 de marzo del mismo año. Cada una de las particiones resultantes esta subparticionada por región de venta.

```
ALTER TABLE ventas_rl
SPLIT PARTITION primero2002 AT ('15/02/2002') INTO
(
  PARTITION primero1_2002,
  PARTITION primero2_2002
);
```

Dividir una subpartición por lista en dos nuevas particiones puede realizarse de la manera siguiente:

```
ALTER TABLE ventas_rl
SPLIT SUBPARTITION p2002norte VALUES ('Peten') INTO
(
  SUBPARTITION p2002norte_calor,
  SUBPARTITION p2002norte_templado
);
```

En este ejemplo las ventas del primer trimestre del año 2002 de la región norte se divide en ventas hechas en departamentos cálidos del norte del país (partición *p2002norte_calor*) y departamentos de clima templado del norte del país (partición *p2002norte_templado*).

Las particiones de índice local correspondientes a las nuevas particiones se marcan como no utilizables y deben ser reconstruidas. A menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES`, todo índice no particionado, o todas las particiones de todos los índices globales son marcadas como no utilizables y deben ser reconstruidas.

3.4.5 Cómo combinar una partición o subpartición

Dentro de una tabla particionada por rango-lista pueden combinarse dos particiones adyacentes de primer nivel (por rango), a continuación se muestra un ejemplo:

```
ALTER TABLE ventas_rl
MERGE PARTITIONS primero1_2002, primero2_2002
INTO PARTITION primero_2002;
```

Dos subparticiones también pueden ser combinadas para formar una nueva partición, a continuación un ejemplo:

```
ALTER TABLE ventas_rl
MERGE SUBPARTITIONS p2002norte_calor, p2002norte_templado
INTO SUBPARTITION p2002norte;
```

Cuando dos particiones o subparticiones son combinadas, las particiones o subparticiones originales son borradas y en su lugar se crea una nueva partición o subpartición.

Las particiones de índice local correspondientes a las nuevas particiones se marcan como no utilizable y deben ser reconstruidas. A menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES`, todo índice no particionado, o todas las particiones de todos los índices globales son marcadas como no utilizables y deben ser reconstruidas.

3.5 Mantenimiento de tablas particionadas por rango-dispersión

3.5.1 Cómo renombrar una partición o subpartición

Una partición dentro de una tabla particionada por rango-dispersión se puede renombrar de la misma manera que se renombra en el particionamiento rango-lista. A

continuación un ejemplo en el que el nombre de la partición *primero2002* se cambia a *trimestre1_2002*:

```
ALTER TABLE ventas_rd
RENAME PARTITION primero2002 TO trimestre1_2002;
```

De igual manera, una subpartición puede cambiar de nombre de la siguiente forma:

```
ALTER TABLE ventas_rd
RENAME SUBPARTITION p2002h1 TO trim1_2002_h1;
```

aquí, la partición *p2002h1* es renombrada a *trim1_2002_h1*.

Cuando se cambia el nombre a una partición o subpartición de una tabla particionada por rango-dispersión, los índices locales, los índices no particionados y los índices globales asociados a la tabla no se ven afectados, todos siguen siendo utilizables.

3.5.2 Cómo agregar una partición o subpartición

A una tabla particionada por rango-dispersión se puede agregar una partición (por rango) o una subpartición (por dispersión).

A continuación se muestra un ejemplo que indica cómo puede agregarse una partición a la tabla, en él la tabla *ventas_rd* no tiene definida una partición `MAXVALUE` por lo que es posible agregar una nueva partición por rango al final del conjunto; la nueva partición, llamada *primero2003*, se crea para almacenar los registros de ventas de los primeros tres meses del año 2003.

```
ALTER TABLE ventas_rd
ADD PARTITION primero2003 VALUES LESS THAN ('01/04/2003')
SUBPARTITIONS 4;
```

Las subparticiones que contendrá la partición pueden especificarse por número o por nombre. A continuación se muestra un ejemplo en el que las subparticiones para la partición primero2003 se especifican por nombre:

```
ALTER TABLE ventas_rd
ADD PARTITION primero2003 VALUES LESS THAN ('01/04/2003')
(
  SUBPARTITION p2003h1,
  SUBPARTITION p2003h2,
  SUBPARTITION p2003h3,
  SUBPARTITION p2003h4
);
```

También es posible agregar una subpartición por dispersión a una partición por rango. Cuando se hace esto, la nueva subpartición es alimentada con registros provenientes de las subparticiones hermanas. A continuación se muestra un ejemplo:

```
ALTER TABLE ventas_rd
MODIFY PARTITION primero2003
ADD SUBPARTITION p2003h5;
```

aquí, se agrega una quinta subpartición, llamada *p2003h5*, a la partición *primero2003* de la tabla *ventas_rd*.

Cuando se agrega una partición, todos los índices asociados a la tabla permanecen utilizables. Sin embargo, cuando se agrega una subpartición sucede lo siguiente:

1. Las particiones de los índices locales correspondientes a la nueva subpartición y a las subparticiones de las cuales se extrajeron registros para alimentar a la nueva subpartición, se marcan como no utilizables y deben ser reconstruidos.
2. A menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES`, todos los índices no particionados o todas las particiones de los índices globales asociados a la tabla se marcan como no utilizables y deben ser reconstruidos.

3.5.3 Cómo borrar una partición o subpartición

En una tabla particionada por rango-dispersión sólo pueden borrarse las particiones de primer nivel (particiones por rango), para eliminar una subpartición debe emplearse la mezcla de particiones explicada en la sección 3.5.6.

En el siguiente ejemplo se muestra cómo borrar una partición, la partición *primero2003* y todas sus subparticiones son eliminadas:

```
ALTER TABLE ventas_rd
DROP PARTITION primero2003;
```

Las particiones de índices locales asociadas a la partición eliminada también son borradas. Los índices no particionados o todas las particiones de los índices globales se marcan como no utilizables y deben ser reconstruidos a menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES` o que la partición borrada haya estado vacía, es decir, sin registros.

3.5.4 Cómo dividir en dos una partición o subpartición

Una partición por rango dentro de una tabla particionada por rango-dispersión puede ser dividida en dos nuevas particiones. La partición original es borrada y en su lugar se crean dos nuevas particiones. Las descripción de las subparticiones que contendrán las nuevas particiones son heredadas de la partición dividida.

En el siguiente ejemplo la partición *primero2002* (que tiene cuatro subparticiones: p2002h1, p2002h2, p2002h3 y p2002h4) se divide en dos nuevas particiones, estas son, *primero1_2002* (que contiene los registros de ventas del 1 de enero de 2002 al 14 de febrero de 2002) y *primero2_2002* (que contiene los registros de ventas del 15 de febrero de 2002 al 31 de marzo de 2002).

```
ALTER TABLE ventas_rd
SPLIT PARTITION primero2002 AT ('15/02/2002') INTO
(
  PARTITION primero1_2002
  PARTITION primero2_2002
);
```

Las particiones de índice local asociadas a las nuevas particiones se marcan como no utilizables y deben ser reconstruidas. A menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES`, los índices no particionados o todas las particiones de los índices globales asociados a la tabla se marcan como no utilizables y deben ser reconstruidos.

Una subpartición por dispersión dentro de una tabla particionada por rango-dispersión no puede ser dividida.

3.5.5 Cómo combinar una partición o subpartición

Dentro de una tabla particionada por rango-dispersión sólo pueden combinarse particiones de primer nivel (por rango). Las subparticiones que contendrá la nueva partición pueden ser especificadas por número o por nombre. Las particiones que se combinan son borrados y en su lugar se crea una nueva.

Las subparticiones no pueden ser combinadas, en su lugar se utiliza la mezcla de subparticiones explicada en la sección 3.5.6.

A continuación se muestran dos ejemplos de combinación de particiones, en el primero las subparticiones de la nueva partición se especifican por número mientras que en el segundo se especifican por nombre. En el ejemplo se combinan las particiones *primero1_2002* y *primero2_2002* para formar la partición *primero2002*.

```
ALTER TABLE ventas_rd
MERGE PARTITIONS primero1_2002, primero2_2002
INTO PARTITION primero2002
SUBPARTITIONS 4;


ALTER TABLE ventas_rd
MERGE PARTITIONS primero1_2002, primero2_2002
INTO PARTITION primero2002
( SUBPARTITION h1,
  SUBPARTITION h2,
  SUBPARTITION h3,
  SUBPARTITION h4
);
```

Las particiones de índice local asociadas a las nuevas subparticiones se marcan como no utilizables y deben ser reconstruidas. A menos que se haya especificado la cláusula `UPDATE GLOBAL INDEXES`, los índices no particionados o todas las particiones de los índices globales asociados a la tabla se marcan como no utilizables y deben ser reconstruidos.

3.5.6 Cómo mezclar subparticiones

La mezcla de subparticiones dentro de una tabla particionada por rango-dispersión consiste en distribuir los registros de la subpartición a mezclar entre las subparticiones hermanas (subparticiones absorbentes) y luego eliminar la subpartición mezclada.

En el siguiente ejemplo se mezcla una subpartición de la partición *primero2003* de la tabla *ventas_rd*:

```
ALTER TABLE ventas_rd
MODIFY PARTITION primero2003
COALESCE SUBPARTITION;
```

Las particiones de índice local de la subpartición mezclada son eliminados y las particiones de índice local correspondientes a las subparticiones absorbentes se marcan como no utilizables y deben ser reconstruidas. A menos que se haya especificado la

cláusula `UPDATE GLOBAL INDEXES`, los índices no particionados o todas las particiones de los índices globales asociados a la tabla se marcan como no utilizables y deben ser reconstruidos.

3.6 Mantenimiento de índices globales

Las operaciones de mantenimiento de un índice global incluyen: renombrar una partición, reconstruir una partición, dividir una partición y borrar una partición. En las siguientes secciones se trata cada una de estas cuatro operaciones.

3.6.1 Cómo renombrar una partición

El nombre de una partición puede ser cambiado de la siguiente manera:

```
ALTER INDEX indice_global
RENAME PARTITION part1 TO particion1;
```

en este ejemplo la partición *part1* del índice llamado *indice_global* se renombra a *particion1*.

3.6.2 Cómo reconstruir una partición

Muchas de las operaciones de mantenimiento realizadas en las tablas marcan como no utilizables las particiones de los índices globales asociados a ellas. Para que las particiones puedan volver a ser utilizadas es necesario reconstruirlas.

La reconstrucción del índice puede hacerse de dos maneras:

1. Reconstruyendo las particiones
2. Borrando el índice y volviéndolo a crear.

El segundo método es más eficiente ya que la tabla se recorre solamente una vez.

En el siguiente ejemplo se reconstruye el índice llamado *indice_global* el cual cuenta con dos particiones: *part1* y *part2*.

```
ALTER INDEX indice_global REBUILD PARTITION part1;
ALTER INDEX indice_global REBUILD PARTITION part2;
```

3.6.3 Cómo dividir una partición

Una partición dentro de un índice global puede ser dividida en dos nuevas particiones, la reconstrucción de las nuevas particiones se hace necesaria para mantener utilizable el índice.

En el siguiente ejemplo la partición *part1* del índice global llamado *indice_global* es dividida en dos nuevas particiones: *part1_a* y *part1_b*.

```
ALTER INDEX indice_global
SPLIT PARTITION part1 AT ('E') INTO
PARTITION part1_a, part1_b;
```

3.6.4 Cómo borrar una partición

Se puede borrar una partición dentro de un índice global. Si la partición contiene datos entonces la siguiente partición es marcada como no utilizable y debe ser reconstruida, además, la partición mayor (la que incluye el parámetro `MAXVALUE`) no puede ser borrada.

A continuación se muestra un ejemplo en el que se borra la partición *part1_a* del índice *indice_global*, la partición *part1_b* queda marcada como no utilizable por lo que se hace necesaria su reconstrucción:

```
ALTER INDEX indice_global
DROP PARTITION part1_a;

ALTER INDEX indice_global REBUILD PARTITION part1_b;
```

3.7 Mantenimiento de índices locales

Las operaciones de mantenimiento que puede realizarse con los índices locales son: renombrar una partición o subpartición y reconstruir una partición o subpartición.

Una partición o subpartición dentro de un índice local no puede ser dividida, la división de la partición o subpartición del índice ocurre cuando se divide la partición o subpartición de tabla correspondiente.

Una partición o subpartición dentro de un índice local no puede ser borrada, la eliminación de la partición o subpartición del índice ocurre cuando se borra la partición o subpartición de tabla correspondiente.

3.7.1 Cómo renombrar una partición o subpartición

Una partición dentro de un índice local puede ser renombrada de la siguiente manera:

```
ALTER INDEX indice_local1 RENAME PARTITION part1 TO particion1;
```

en este ejemplo la partición *part1* del índice *indice_local1* es renombrada a *particion1*.

De igual manera, una subpartición dentro de un índice local puede ser renombrada de la siguiente manera:

```
ALTER INDEX indice_local RENAME SUBPARTITION part1a TO particion1a;
```

3.7.2 Cómo reconstruir una partición o subpartición

Una partición dentro de un índice local puede ser reconstruida de la siguiente manera:

```
ALTER INDEX indice_local1 REBUILD PARTITION part1;
```

en este ejemplo se reconstruye la partición *part1* del índice *indice_local1*. Si la tabla está particionada por rango-dispersión, entonces no puede reconstruirse la partición, en su lugar deben reconstruirse todas las subparticiones.

Una subpartición dentro de un índice local puede ser reconstruida de la siguiente manera:

```
ALTER INDEX indice_local REBUILD SUBPARTITION part1a;
```

en este ejemplo se reconstruye la partición *part1a* del índice *indice_local*.

4. UTILIZACIÓN DE TABLAS E ÍNDICES PARTICIONADOS EN UNA COMPAÑÍA TELEFÓNICA

En este capítulo se expone el caso práctico de utilización de tablas e índices particionados en una compañía telefónica. Dicha compañía está interesada en disminuir el tiempo que toma ejecutar dos consultas que extraen información de las dos tablas más grandes que existen en su *data warehouse*.

Las consultas en cuestión son:

1. El reporte semanal de egresos
2. El reporte mensual de egresos

El código de ética de la compañía impone las siguientes restricciones en la realización del trabajo:

1. No es posible dar el nombre de la empresa.
2. No es permitido mostrar las sentencias `SELECT` que extraen la información proporcionada por los reportes.
3. No es permitido detallar la información proporcionada por los reportes.
4. No es permitido mostrar toda la estructura de las tablas ni el contenido de las mismas.

Estas restricciones no influyen en la parte medular del objetivo de disminución de tiempos de ejecución ya que bastará con saber cuánto tiempo le toma a cada reporte extraer la información requerida para saber si dicha meta está siendo lograda o no. Asimismo, es posible mostrar los nombres y parte de la estructura de las tablas (e

índices) involucradas en los reportes para entender las sentencias que crean las tablas y los índices particionados.

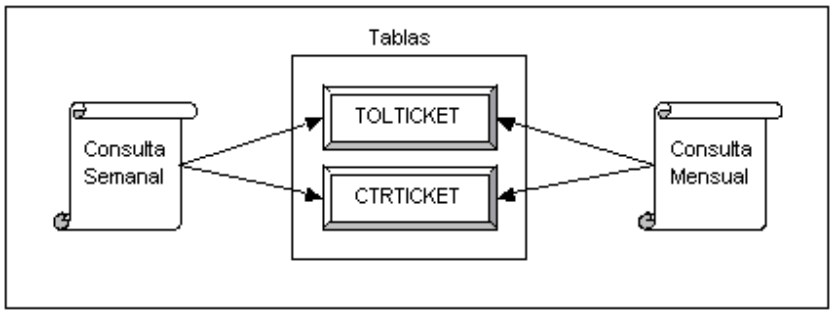
La información proporcionada por los reportes es extraída de dos tablas, estas son:

1. La tabla *tolticket*
2. La tabla *ctrticket*

cada una de estas tablas contiene una columna llamada *tickethour*, dicha columna almacena la fecha y hora en que se realizó una llamada telefónica. La columna *tickethour* es determinante para establecer qué registros se incluyen en el reporte ya que los mismos extraen información (como su nombre lo indica) por intervalos de fecha.

Como se denotó anteriormente, por razones de seguridad no es posible detallar el nombre de la compañía, la información brindada por las consultas y los datos almacenados por las tablas, bastará con conocer los tiempos de ejecución de las consultas. En la siguiente figura se muestra una representación gráfica de la relación existente entre las consultas y las tablas.

Figura 13. Relación existente entre las consultas y las tablas



Como puede observarse, tanto el reporte semanal de egresos como el reporte mensual de egresos extraen la información de ambas tablas.

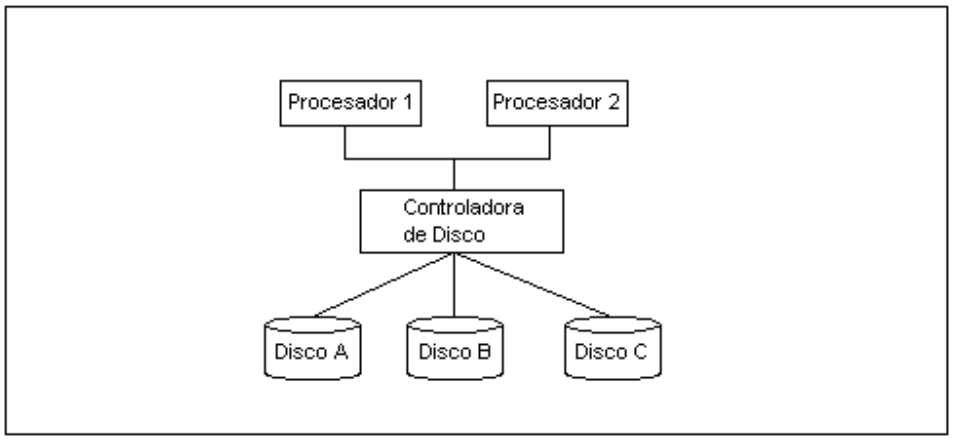
Las características de las tablas consultadas se muestran en la siguiente tabla:

Tabla I. Características de las tablas consultadas

NOMBRE DE LA TABLA	TAMAÑO DE LA TABLA	NÚM. DE REGISTROS	HISTORIA ALMACENADA
TOLTICKET	3,780 Mb.	50,135,090	de mayo/2001 a octubre/2002
CTRTICKET	11,328 Mb.	93,580,388	de marzo/2002 a octubre/2002

Las tablas se encuentran almacenadas en un servidor de datos organizado como se muestra en la siguiente figura:

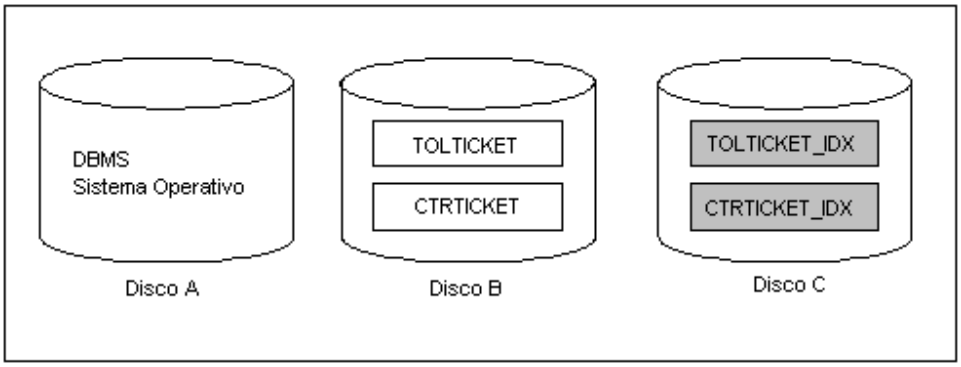
Figura 14. Organización del servidor de datos que contiene las tablas



4.1 Solución actual

Actualmente las tablas *tolticket* y *ctrticket* se encuentran almacenadas en el mismo disco (disco B). Cada una de las tablas cuenta con un índice definido sobre la columna *tickethour*. El objetivo de estos índices es optimizar las consultas. El índice de la tabla *tolticket* se llama *tolticket_idx* mientras que el índice de la tabla *ctrticket* se llama *ctrticket_idx*. Ambos índices se encuentran almacenados en un disco distinto a aquel en donde se encuentran las tablas, el disco C. En la siguiente figura se muestra la organización de las tablas y los índices para la solución actual.

Figura 15. Organización de tablas e índices para la solución actual



Los tiempos de ejecución de las consultas para esta solución se muestran en la siguiente tabla:

Tabla II. Tiempos de ejecución para la solución actual

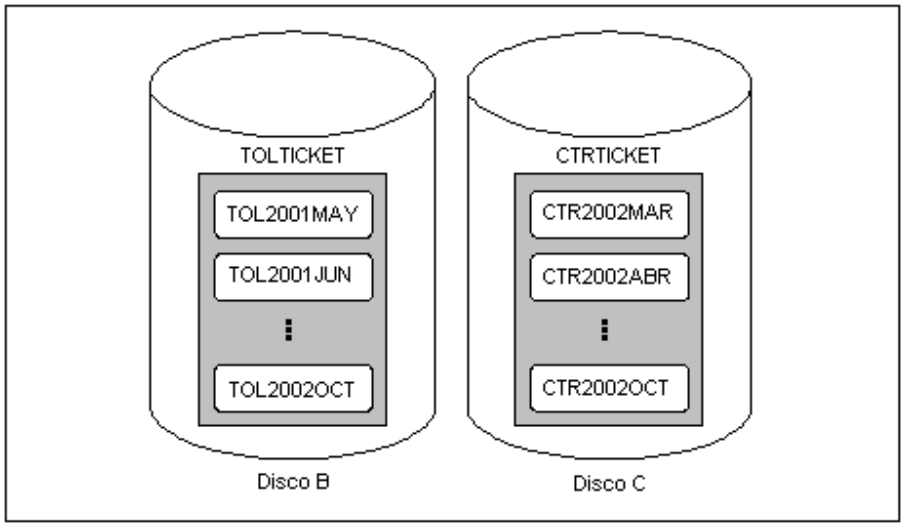
CONSULTA	TIEMPO DE EJECUCIÓN
Semanal	7 min. 01.07 seg.
Mensual	7 min. 06.01 seg.

En las siguientes cuatro secciones (4.2, 4.3, 4.4 y 4.5) se muestran cuatro opciones de organización de las tablas e índices para disminuir los tiempos de ejecución de las consultas. Dada la naturaleza histórica de las tablas, los particionamientos se hacen por rango de fecha. El propósito de crear cuatro tipos de organización de las tablas e índices es encontrar el que más se adecue a la disminución de tiempo que se busca. Una vez establecidas las opciones de particionamiento y los tiempos ofrecidos por cada una, corresponde a la compañía telefónica elegir la opción que más le convenga.

4.2 Opción de particionamiento mensual

En esta opción las tablas *tolticket* y *ctrticket* se particionan por mes a través de la columna *tickethour*. Las particiones de la tabla *tolticket* se almacenan en el disco B mientras que las particiones de la tabla *ctrticket* se almacenan en el disco C. La siguiente figura muestra esta organización:

Figura 16. Organización de las tablas para la opción mensual



Las tablas particionadas se crean de la siguiente forma:

```
create table tolticket (
  tickethour date,
  ...
)
partition by range (tickethour)
(
  partition tol2001may values less than ('01-06-2001'),
  partition tol2001jun values less than ('01-07-2001'),
  partition tol2001jul values less than ('01-08-2001'),
  ...
  partition tol2002oct values less than ('01-11-2002')
);

create table ctrticket (
  tickethour date,
  ...
)
partition by range (tickethour)
(
  partition ctr2002mar values less than ('01-04-2002'),
  partition ctr2002abr values less than ('01-05-2002'),
  partition ctr2002may values less than ('01-06-2002'),
  ...
  partition ctr2002oct values less than ('01-11-2002')
);
```

Al crear las tablas particionadas de esta forma, se obtienen los tiempos de ejecución que se muestran en la siguiente tabla:

Tabla III. Tiempos de ejecución para la opción mensual

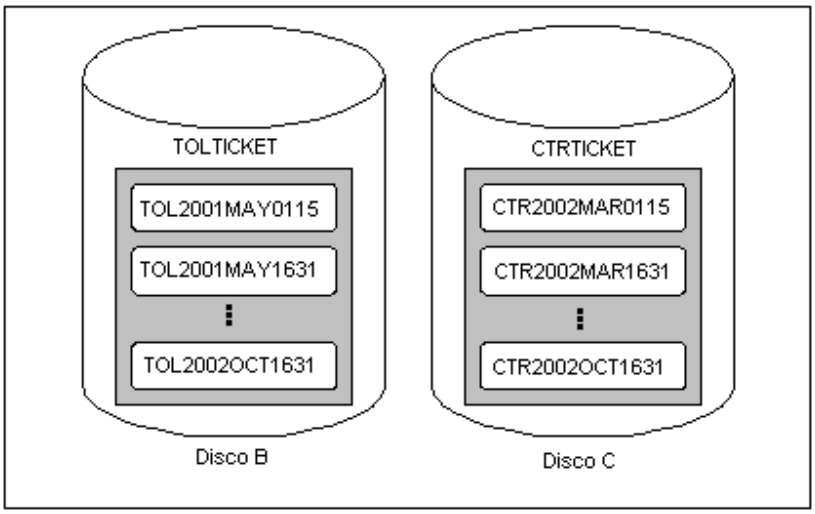
CONSULTA	TIEMPO DE EJECUCIÓN
Semanal	48.04 seg.
Mensual	49.03 seg.

4.3 Opción de particionamiento quincenal

En esta opción las tablas *tolticket* y *ctrticket* se particionan por quincena. Las particiones de la tabla *tolticket* se almacenan en el disco B mientras que las particiones

de la tabla *ctrticket* se almacenan en el disco C. La siguiente figura muestra esta organización:

Figura 17. Organización de las tablas para la opción quincenal



Las tablas particionadas se crean de la siguiente forma:

```
create table tolticket (
  tickethour date,
  ...
)
partition by range (tickethour)
(
  partition tol2001may0115 values less than ('16-05-2001'),
  partition tol2001may1631 values less than ('01-06-2001'),
  ...
  partition tol2002oct1631 values less than ('01-11-2002')
);


create table ctrticket (
  tickethour date,
  ...
)
partition by range (tickethour)
(
  partition ctr2002mar0115 values less than ('16-03-2002'),
```

```
  partition ctr2002mar1631 values less than ('01-04-2002'),
  ...
  partition ctr2002oct1631 values less than ('01-11-2002')
);
```

Al crear las tablas particionadas de esta forma, se obtienen los tiempos de ejecución que se muestran en la siguiente tabla:

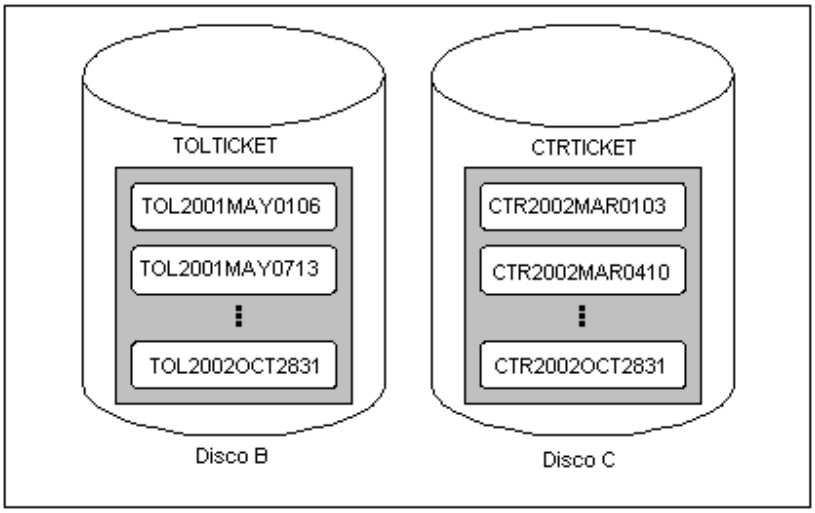
Tabla IV. Tiempos de ejecución para la opción quincenal

CONSULTA	TIEMPO DE EJECUCIÓN
Semanal	23.04 seg.
Mensual	1 min. 09.05 seg.

4.4 Opción de particionamiento semanal

En esta opción las tablas *tolticket* y *ctrticket* se particionan por semana. Las particiones de la tabla *tolticket* se almacenan en el disco B mientras que las particiones de la tabla *ctrticket* se almacenan en el disco C. La siguiente figura muestra esta organización:

Figura 18. Organización de las tablas para la opción semanal



Las tablas particionadas se crean de la siguiente forma:

```
create table tolticket (
  tickethour date,
  ...
)
partition by range (tickethour)
(
  partition tol2001may0106 values less than ('07-05-2001'),
  partition tol2001may0713 values less than ('14-05-2001'),
  partition tol2001may1420 values less than ('21-05-2001'),
  partition tol2001may2127 values less than ('28-05-2001'),
  partition tol2001may2831 values less than ('01-06-2001'),
  ...
  partition tol2002oct2831 values less than ('01-11-2002')
);


create table ctrticket (
  tickethour date,
  ...
)
partition by range (tickethour)
(
  partition ctr2002mar0103 values less than ('04-03-2002'),
  partition ctr2002mar0410 values less than ('11-03-2002'),
  partition ctr2002mar1117 values less than ('18-03-2002'),
  partition ctr2002mar1824 values less than ('25-03-2002'),
```

```
  partition ctr2002mar2531 values less than ('01-04-2002'),
  ...
  partition tol2002oct2831 values less than ('01-11-2002')
);
```

Al crear las tablas particionadas de esta forma, se obtienen los tiempos de ejecución que se muestran en la siguiente tabla:

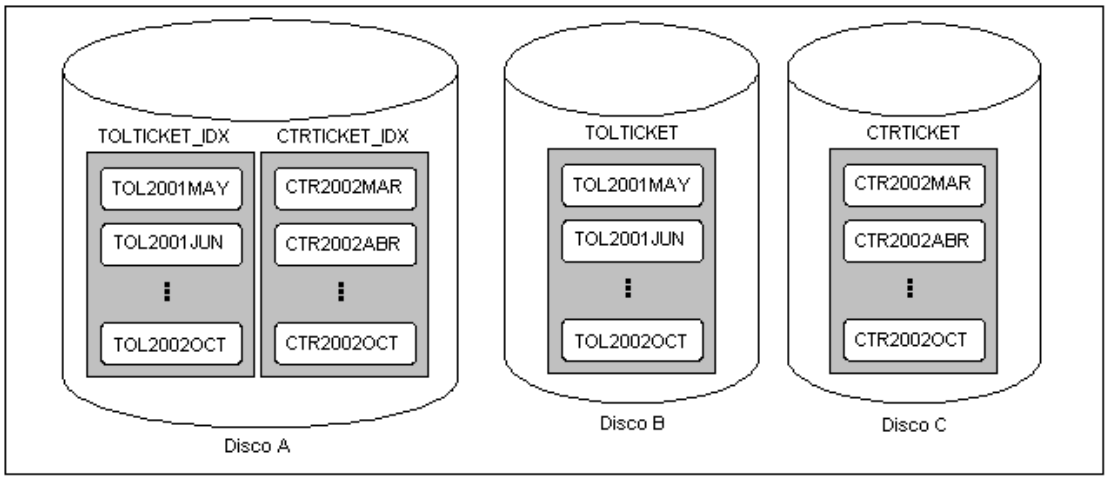
Tabla V. Tiempos de ejecución para la opción semanal

CONSULTA	TIEMPO DE EJECUCIÓN
Semanal	12.00 seg.
Mensual	57.08 seg.

4.5 Opción de particionamiento mensual con índices particionados

En esta opción las tablas *tolticket* y *ctrticket* se particionan por mes y cada uno de los índices asociados a ellas (*tolticket_idx* y *ctrticket_idx*) se particionan localmente asignándolos al disco A. La siguiente figura muestra esta organización:

Figura 19. Organización de las tablas para la opción mensual con índices particionados



Las tablas particionadas se crean de la siguiente forma:

```
create table tolticket (
  tickethour date,
  ...
)
partition by range (tickethour)
(
  partition tol2001may values less than ('01-06-2001'),
  partition tol2001jun values less than ('01-07-2001'),
  partition tol2001jul values less than ('01-08-2001'),
  ...
  partition tol2002oct values less than ('01-11-2002')
);

create table ctrticket (
  tickethour date,
  ...
)
partition by range (tickethour)
(
  partition ctr2002mar values less than ('01-04-2002'),
  partition ctr2002abr values less than ('01-05-2002'),
  partition ctr2002may values less than ('01-06-2002'),
  ...
  partition ctr2002oct values less than ('01-11-2002')
);
```

Los índices particionados localmente se crean de la siguiente manera:

```
create index tolticket_idx on tolticket(tickethour) local
(
  partition tol2001may,
  partition tol2001jun,
  partition tol2001jul,
  ...
  partition tol2002oct
);

create index ctrticket_idx on ctrticket(tickethour) local
(
  partition ctr2002mar,
  partition ctr2002abr,
  partition ctr2002may,
  ...
  partition ctr2002oct
);
```

Al crear las tablas y los índices particionados de esta forma, se obtienen los tiempos de ejecución que se muestran en la siguiente tabla:

Tabla VI. Tiempos de ejecución para la opción mensual con índices particionados

CONSULTA	TIEMPO DE EJECUCIÓN
Semanal	45.00 seg.
Mensual	49.03 seg.

Al comparar los datos de la tabla VI con los datos de la tabla III puede observarse que no existe una diferencia significativa que justifique el espacio extra ocupado por los índices. Esta relación se mantuvo para el particionamiento quincenal con índices y para el semanal con índices. Debido a la no justificación de empleo de los índices, sólo se incluyen los tiempos del particionamiento mensual para mostrar que la inclusión de índices no resulta beneficiosa para la disminución de tiempos de ejecución de las consultas involucradas.

4.6 Comparación de los tiempos de ejecución

A continuación se comparan los tiempos ofrecidos por la solución actual y por cada una de las opciones analizadas. La siguiente tabla muestra los resultados:

Tabla VII. Comparación de tiempos de ejecución para las consultas

	CONSULTA	TIEMPO DE EJECUCIÓN
SOLUCIÓN ACTUAL	Semanal	7 min. 01.07 seg.
	Mensual	7 min. 06.01 seg.
OPCIÓN MENSUAL	Semanal	48.04 seg.
	Mensual	**49.03 seg.**
OPCIÓN QUINCENAL	Semanal	23.04 seg.
	Mensual	1 min. 09.05 seg.
OPCIÓN SEMANAL	Semanal	**12.00 seg.**
	Mensual	57.08 seg.
MENSUAL CON ÍNDICES	Semanal	48.04 seg.
	Mensual	49.03 seg.

De la anterior tabla puede observarse lo siguiente:

1. El menor tiempo de ejecución para el reporte semanal de egresos lo ofrece el particionamiento semanal.

2. El menor tiempo de ejecución para el reporte mensual de egresos lo ofrece el particionamiento mensual.

3. El uso de índices no disminuye el tiempo de ejecución para las consultas en cuestión por lo que su uso no es necesario.

Que los índices sean innecesarios para los reportes de egresos mensual y semanal, no implica que siempre que se particione una tabla no se necesiten índices para despachar las consultas que se hacen a ella.

En el caso de la telefónica existe un reporte (el reporte de llamadas realizadas) que muestra el detalle de llamadas realizadas por un número telefónico determinado. Para

minimizar el tiempo de ejecución de esta consulta, se creó un índice particionado globalmente sobre la columna numero_a (el número de telefóno que realizó la llamada) de la tabla *ctrticket*, con ello se logró disminuir el tiempo de 4 minutos a 30 segundos. El detalle de esta tercera consulta escapa de los límites del presente trabajo, pero se incluyó para mostrar a la telefónica el beneficio del particionamiento de índices.

CONCLUSIONES

1. Una tabla particionada es una tabla dividida en segmentos más pequeños y manejables llamados *particiones*. El particionamiento puede realizarse por rango, lista, dispersión, rango-lista o rango-dispersión.

2. El particionamiento por rango es conveniente para tablas históricas.

3. El particionamiento por lista es conveniente para agrupar registros relacionados de forma lógica de acuerdo con la perspectiva humana, por ejemplo, geográficamente.

4. El particionamiento por dispersión resulta útil para tablas no históricas en sistemas con paralelismo.

5. El particionamiento compuesto (rango-lista o rango-dispersión) aprovecha las ventajas de cada uno de los dos métodos de particionamiento que incluye.

6. Un índice particionado es un índice dividido en segmentos más pequeños y manejables llamados *particiones*. Los índices pueden particionarse de forma global o local.

7. Los índices globales no están asociados al método de particionamiento de la tabla y son preferibles para ambientes OLTP.

8. Los índices locales están asociados al método de particionamiento de la tabla y son preferibles para ambientes DSS.

9. El particionamiento aumenta la disponibilidad y disminuye los tiempos asociados a la administración y acceso a las tablas e índices.

10. En el caso de la compañía telefónica, el particionamiento semanal brinda el menor tiempo de ejecución para el reporte semanal de egresos mientras que el particionamiento mensual hace lo propio para el reporte mensual de egresos.

RECOMENDACIONES

1. Particionar las tablas *tolticket* y *ctrticket* por semana. Al particionar por semana el reporte mensual de egresos se tarda 8.05 seg. más que si se particionara por mes, sin embargo, al particionar por mes el reporte semanal de egresos se tarda 36.04 seg. más que si se particionara por semana.

2. No utilizar índices para los reportes de egresos.

3. Considerar la creación de índices sobre las tablas *tolticket* y *ctrticket* sólo si la consulta asociada no extrae datos por intervalos de fecha.

BIBLIOGRAFÍA

1. Baylis, Ruth y Kathy Rich. ***Oracle9i Database Administrator's Guide, Release 2 (9.2)***. s.l.: s.e., 2002. 1038pp.

2. Cyran, Michele. ***Oracle9i Database Concepts, Release 2 (9.2).*** s.l.: s.e., 2002. 734pp.

3. Date, C. J. **Introducción a los sistemas de bases de datos volumen 1 5ª. ed.** Trad.: Roberto Escalona García. E.U.A.: Addison-Wesley Iberoamericana, 1993.

4. ***Definitions for thousands of the most current IT-related words***. *http://whatis.techtarget.com,* mayo 2002.

5. Kimball, Ralph y otros. ***The Data Warehouse Lifecycle Toolkit***. E.U.A.: John Wiley & Sons, Inc., 1998.

6. Lane, Paul y Viv Schupmann. ***Oracle9i Data Warehousing Guide, Release 2 (9.2).*** s.l.: s.e., 2002. 666pp.

7. Möller, Michael y Jim Womack. ***Implement Partitioning***. s.l.: s.e., 2002. 180pp.