Tema 8

Particionado.

Gestión de contenidos.

Introducción al particionado

- Utilidad de las particiones:
  - Soluciona los problemas que surgen al manejar tablas muy grandes.
  - Especialmente útil en aplicaciones que manejan grandes volúmenes de datos.

- Forma de solucionarlo:
  - Las descompone en piezas más pequeñas y manejables denominadas particiones.

- Requisitos:
  - Cada partición de una tabla o índice debe tener los mismos atributos lógicos (nombre de las columnas, tipos de datos, etc).
  - Los atributos físicos de cada partición son separados (pctree, pctused, tablespaces, etc).
  - Todas las particiones de un objeto deben residir en tablespaces con el mismo tamaño de bloque.

Introducción al particionado

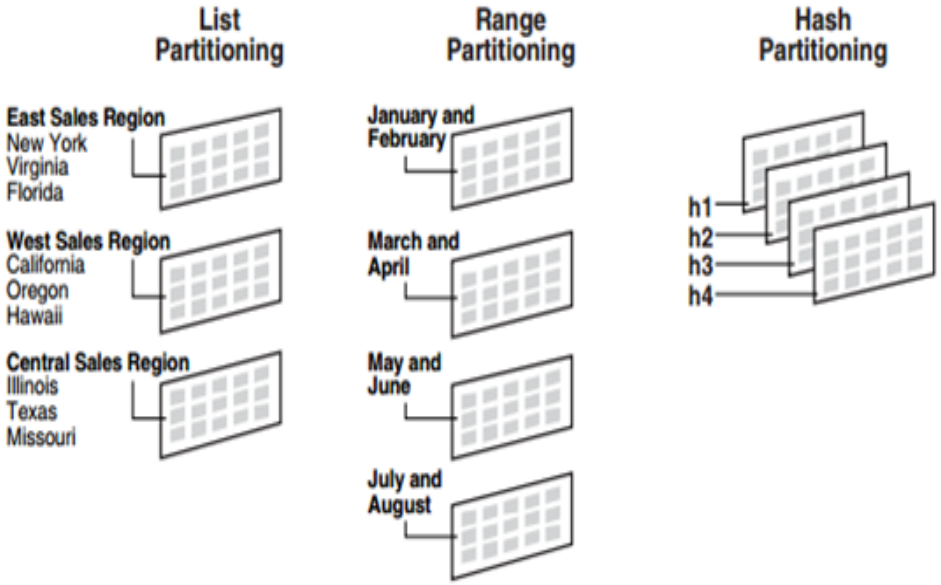
- Clave de partición:
  - Cada fila de una tabla es asignada unívocamente a una única partición.
  - La clave de partición es un conjunto de una o más columnas que determina la partición a la que pertenece cada fila.
  - Consiste en un listado ordenado de 1 a 16 columnas.
  - No puede contener las pseudocolumnas LEVEL, ROWID o MLSLABEL.
  - Pueden contener columnas que permitan nulos.

- Tablas particionadas
  - Las tablas pueden ser particionadas hasta en 1024k-1 particiones separadas.
  - No se puede particionar tablas con columnas de tipo LONG o LONG RAW.
  - Se puede particionar tablas con columnas de tipo BLOB o CLOB.

- Para reducir el espacio de disco y el uso de memoria se pueden almacenar en formato comprimido.
  - Esto puede acelerar la ejecución de consultas pero supone un ligero costo adicional en CPU.

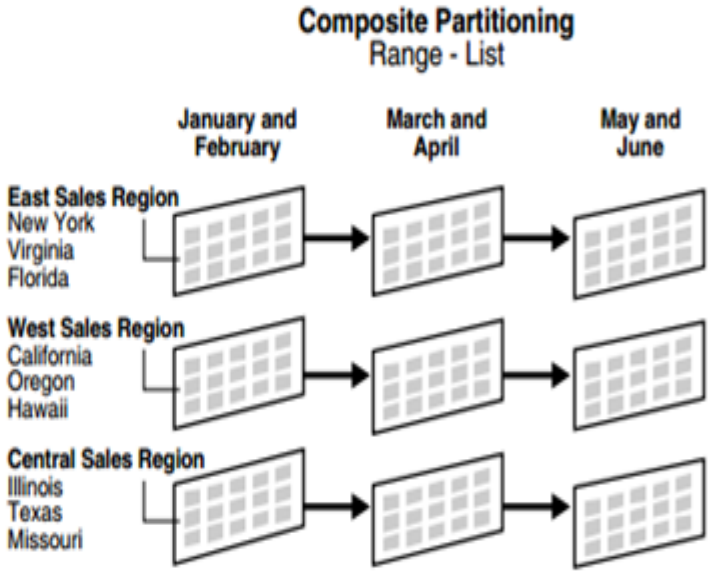
Métodos de particionado

- Oracle provee los siguientes métodos de particionado:
  - Particionado por rangos
  - Particionado por listas
  - Particionado por hash



Métodos de particionado

- También provee métodos compuestos a partir de dichos métodos:
  - Particionado compuesto rango-lista
  - Particionado compuesto rango-hash



Métodos de particionado

- En el siguiente ejemplo podemos ver una sentencia que crea la tabla sales_range particionada usando el método por rangos por el campo sales_date.

```
CREATE TABLE sales_range
(
        salesman_id NUMBER(5),
        salesman_name VARCHAR2(30),
        sales_amount NUMBER(10),
        sales_date (DATE)
)
PARTITION BY RANGE (sales_date)
(
        PARTITION sales_jan2000 VALUES LESS THAN (TO_DATE('02/01/2000', 'MM/DD/YYYY')),
        PARTITION sales_feb2000 VALUES LESS THAN (TO_DATE('03/01/2000', 'MM/DD/YYYY')),
        PARTITION sales_mar2000 VALUES LESS THAN (TO_DATE('04/01/2000', 'MM/DD/YYYY')),
        PARTITION sales_apr2000 VALUES LESS THAN (TO_DATE('05/01/2000', 'MM/DD/YYYY'))
);
```

Métodos de particionado

- En el siguiente ejemplo vamos a particionar la tabla sales por región, dependiendo de su localización geográfica:

```
CREATE TABLE sales_list
(
        salesman_id NUMBER(5),
        salesman_name VARCHAR2(30),
        sales_state VARCHAR2(20),
        sales_amount NUMBER(10),
        sales_date (DATE)
)
PARTITION BY LIST(sales_state)
(
        PARTITION sales_west VALUES ('California', 'Hawaii'),
        PARTITION sales_east VALUES ('New York', 'Virginia', 'Florida'),
        PARTITION sales_central VALUES ('Texas', 'Illinois'),
        PARTITION sales_other VALUES (DEFAULT)
);
```

Métodos de particionado

- En el siguiente ejemplo, se crea la tabla sales_hash que será particionada por hash sobre el campo salesman_id.
- Los nombres de los tablespaces son ts1, ts2, ts3 y ts4.
- Con esta sintaxis las particiones serán distribuidas por round robin entre los tablespaces especificados.

```
CREATE TABLE sales_hash
(
        salesman_id NUMBER(5),
        salesman_name VARCHAR2(30),
        sales_amount NUMBER(10),
        week_no NUMBER(2)
)
PARTITION BY HASH (salesman_id) PARTITIONS 4 STORE IN (ts1, ts2, ts3, ts4);
```

Métodos de particionado

- En el siguiente ejemplo de partición por el método compuesto rango-hash, se crea la tabla sales_composite y es particionada por rangos en el campo sales_data y subparticionada por hash en el campo salesman_id.

```
CREATE TABLE sales_range
(
        salesman_id NUMBER(5),
        salesman_name VARCHAR2(30),
        sales_amount NUMBER(10),
        sales_date DATE
)
PARTITION BY RANGE (sales_date)
        SUBPARTITION BY HASH (salesman_id)
                        SUBPARTITION TEMPLATE
                        (
                                        SUBPARTITION sp1 TABLESPACE ts1,
                                        SUBPARTITION sp2 TABLESPACE ts2,
                                        SUBPARTITION sp3 TABLESPACE ts3,
                                        SUBPARTITION sp4 TABLESPACE ts4
                        )
(
        PARTITION sales_jan2000 VALUES LESS THAN (TO_DATE('02/01/2000', 'MM/DD/YYYY')),
        PARTITION sales_feb2000 VALUES LESS THAN (TO_DATE('03/01/2000', 'MM/DD/YYYY')),
        PARTITION sales_mar2000 VALUES LESS THAN (TO_DATE('04/01/2000', 'MM/DD/YYYY')),
        PARTITION sales_apr2000 VALUES LESS THAN (TO_DATE('05/01/2000', 'MM/DD/YYYY')),
        PARTITION sales_may2000 VALUES LESS THAN (TO_DATE('06/01/2000', 'MM/DD/YYYY'))
);
```

Métodos de particionado

- En el siguiente ejemplo de partición por el método compuesto rango-lista, se crea la tabla bimonthly_regional_sales y es particionada por rangos en el campo txn_date y subparticionada por lista en el campo state

```
CREATE TABLE bimontly_regional_sales
(
        deptno NUMBER,
        Item_no VARCHAR2(20),
        Txn_date DATE,
        txn_amount NUMBER,
        state VARCHAR2(2)
)
PARTITION BY RANGE (txn_date)
        SUBPARTITION BY LIST (state)
                        SUBPARTITION TEMPLATE
                        (
                                        SUBPARTITION east VALUES ('NY', 'VA', 'FL') TABLESPACE ts1,
                                        SUBPARTITION west VALUES ('CA', 'OR', 'HI') TABLESPACE ts2,
                                        SUBPARTITION central VALUES ('IL', 'TX', 'MO') TABLESPACE ts3,
                        )
(
        PARTITION janfeb_2000 VALUES LESS  THAN (TO_DATE ('1-MAR-2000', 'DD-MON-YYYY')),
        PARTITION marapr_2000 VALUES LESS  THAN (TO_DATE ('1-MAY-2000', 'DD-MON-YYYY')),
        PARTITION mayjun_2000 VALUES LESS  THAN (TO_DATE ('1-JUL-2000', 'DD-MON-YYYY')),
);
```

Cuando particionar una Tabla

- A continuación proponemos una serie de sugerencias sobre cuando particionar una tabla:
  - Las tablas superiores a 2GB deberían ser siempre consideradas para el particionado.
  - Las tablas que contengan valores históricos, en los cuales se añaden datos a la partición más reciente también deben ser consideradas para el particionado.
    - Un ejemplo típico es una tabla histórica en la que los datos del mes actual son actualizables, pero los del resto del año son de sólo lectura.

Particionado de índices

Particionado de índices

- Conceptos:
  - Clave de indización (CI).
  - Clave de particionado (CP).
  - Tabla particionada asociada (TP).

- Particionado local:
  - En la definición del índice sólo hay que indicar que es local.
  - El índice hereda el particionado de la tabla asociada.
  - Su gestión la lleva a cabo Oracle de manera automática.

- Particionado global:
  - Cuando no es local.

Particionado de índices

- Prefijado.
  - Si la CP es prefijo de la CI se dice que el particionado del índice es prefijado.
  - Posibilita la poda de particiones en el índice.

- Unicidad del índice.
  - Oracle permite la unicidad en un índice si al buscar una ocurrencia lleva a una única partición.
  - Cuando la CP es un subconjunto de la CI, se cumple la regla anterior.

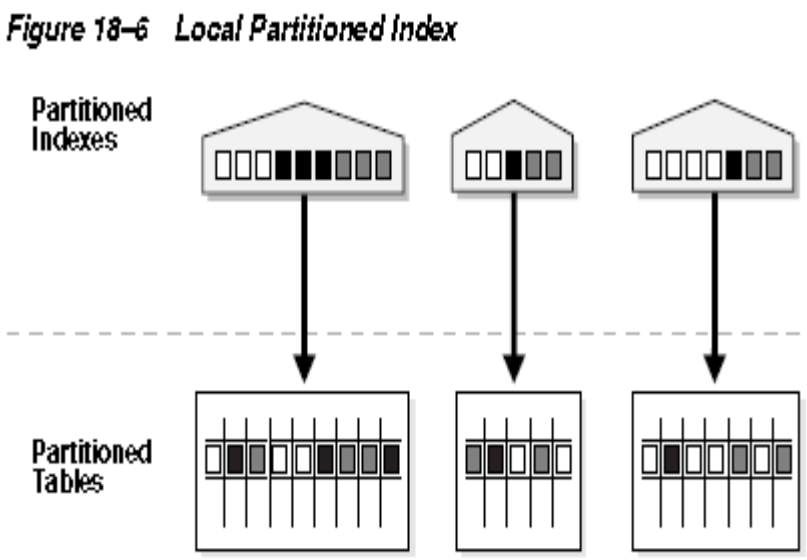
Particionado de índices

- El índice local está recomendado:
  - Si la CP de la tabla es subconjunto de CI.
  - Si prioriza la administración.
  - Si estamos en un entorno de DSS y se prioriza el volumen de información que fluye.

- El índice global está recomendado:
  - Si el índice es único.
  - Si estamos en un entorno de OLTP y se prioriza el tiempo de respuesta.

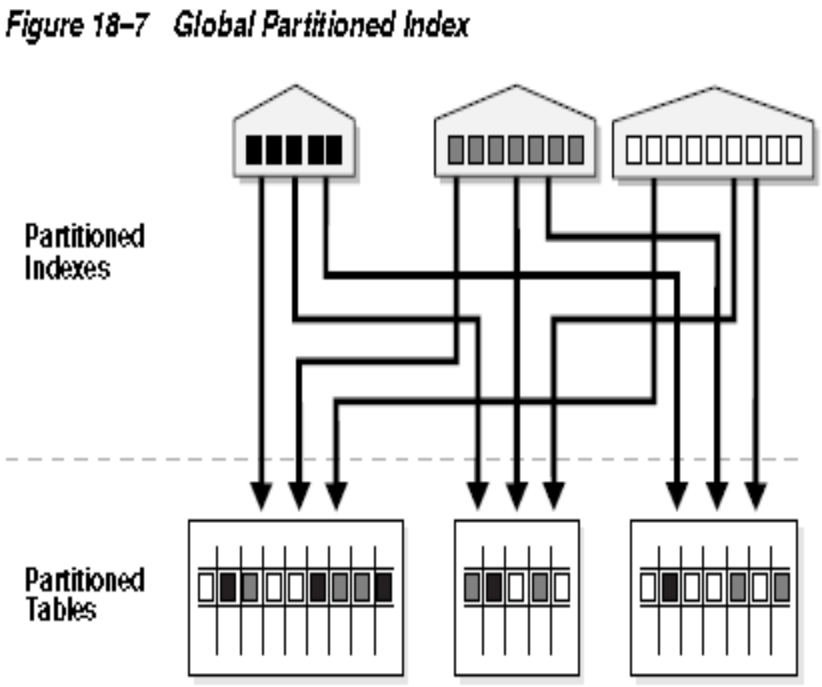
Índices locales particionados

- No es posible añadir una partición a un índice local, las particiones de índice se añaden junto con las de la tabla.
- Un índice local puede ser único (CP es clave), esto es útil en entornos OLTP.



Índices globales particionados

- Oracle sólo ofrece dos modelos:
  - Por rango.
  - Por hash.



Índices globales particionados por rango

- Flexibles, independientes del particionado de la tabla.
- Usados en entornos OLTP, acceso eficiente a cualquier registro individual.
- Mayor partición limitada (todos los valores a MAXVALUE).
- Asegura que todas las filas de la tabla puedan ser representadas en el índice.
- El índice global puede ser único o no.

Índices globales particionados por rango

- No es posible añadir una partición (límite de la partición superior con MAXVALUE).
- Se puede añadir una nueva partición superior dividiendola con ALTER INDEX SPLIT PARTITION.
- Se pueden eliminar particiones usando ALTER INDEX DROP PARTITION
- Al eliminar una partición con datos, la siguiente mayor partición se marca como no usable.
- En un índice global no se puede eliminar la partición superior.

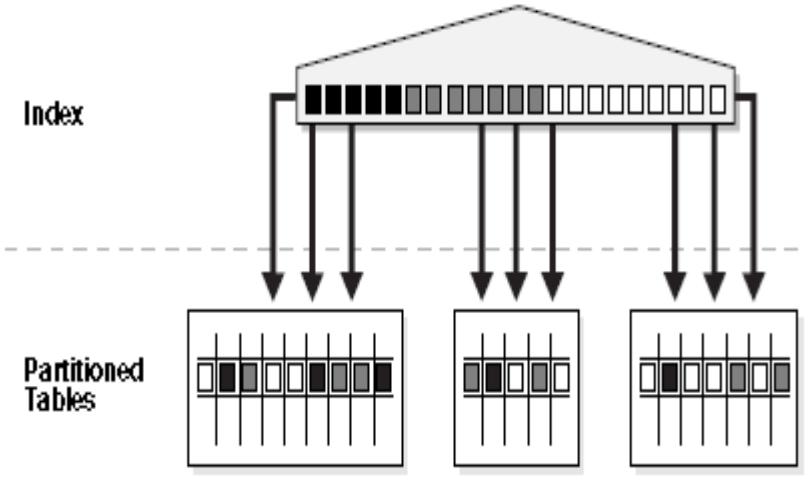
Índices globales particionados por hash

- Mejoran el rendimiento dispersando el contenido cuando la mayoría de inserciones sólo ocurren en el extremo derecho de un índice.
- Ciertas operaciones sobre una tabla organizada como montículo marcan todos los índices globales como no usables.

Índices globales no particionados

- Tienen el mismo comportamiento que un índice no particionado.
- Se emplea en entornos OLTP
  - Acceso eficiente a cualquier registro individual.

Figure 18–8 Global Nonpartitioned Index



Mejoras de funcionamiento usando el particionado

- Poda de particiones.
- Joins de particiones.
- DML en paralelo.

Poda de particiones

- Oracle reconoce las particiones y poda aquellas que no requieran ser accedidas (las omite).
  - No necesitamos acceder a los datos de todo el año si nuestra consulta sólo afecta a los datos del mes de marzo.

- Las particiones no puede podarse si la instrucción SQL aplica una función a la columna particionada (salvo la función TO_DATE), tanto en tablas como en índices.

Administración de contenidos

Administración de contenidos

- Introducción a la administración de contenidos.
- Vista general de XML en Oracle.
- Vista general de LOBs.
- Vista general de Oracle Text.
- Vista general de Oracle Ultra Search.
- Vista general de Oracle interMedia.
- Vista general de Oracle Spatial.

Administración de contenidos

- Oracle Database incluye tipos de datos para manejar todos los tipos de contenidos de Internet tal como datos relacionales, datos objeto-relacionales, texto, XML, audio, video, imagen, y espaciales.
  - Esos tipos de datos aparecen como tipos nativos en la base de datos.

- Dado que las aplicaciones evolucionan para incluir semánticas cada vez más ricas, se encuentran con la necesidad de hacer frente a los siguientes tipos de datos:
  - Datos estructurados simples.
  - Datos estructurados complejos
  - Datos semi-estructurados.
  - Datos no estructurados.

Administración de contenidos

- Tradicionalmente, el modelo relacional ha tenido mucho éxito en el tratamiento de datos estructurados simples.
  - El tipo que puede caber en tablas simples.
- Oracle añade características relacionales de objetos para que las aplicaciones puedan hacer frente a los datos estructurados complejos.
  - Colecciones, referencias, tipos definidos por el usuario, etc.
- Las tecnologías de cola, como Oracle Streams Advanced Queuing, se ocupan de mensajes y otros datos semi-estructurados.
- Introduciremos las tecnologías de Oracle para manejar los datos no estructurados.

Administración de contenidos

- Los datos no estructurados no pueden descomponerse en componentes estándar.
  - Los datos acerca de un empleado pueden ser "estructurado" en un nombre (probablemente una cadena de caracteres), una identificación (probablemente un número), un sueldo, y así sucesivamente.
  - Pero si se le da una foto, los datos realmente consisten en una larga secuencia de 0 y 1.
    - Estos 0s y 1s se utilizan para cambiar los píxeles encendido o apagado, para que veas la foto en una pantalla, pero no se puede dividir en una estructura más fina en términos de almacenamiento de base de datos.

Administración de contenidos

- Los datos no estructurados, tales como texto, imágenes gráficas, videoclips, video de movimiento completo, y onda de sonido tienden a ser grandes.
  - Un registro típico empleado puede ser unos pocos cientos de bytes, pero incluso pequeñas cantidades de datos multimedia pueden ser miles de veces más grande.
- Algunos datos multimedia pueden residir en archivos del sistema operativo siendo accesibles desde la base de datos.

XML de Oracle

- Extensible Markup Language (XML) es:
  - Un lenguaje de marcado basado en etiquetas que permite a los desarrolladores crear sus propias etiquetas para describir los datos que se intercambian entre aplicaciones y sistemas.
  - Ampliamente adoptado como el lenguaje común de intercambio de información entre las empresas.
  - Legible por humanos, es decir, que es texto sin formato.
    - Debido a que es texto sin formato, los documentos XML y mensajes basados en XML se pueden enviar fácilmente usando protocolos comunes, tales como HTTP o FTP.
  - Oracle XML DB trata XML como un tipo de datos nativo en la base de datos.
    - Oracle XML DB no es un servidor independiente.

LOBs

- El tipos de datos de objetos grande (LOB) BLOB, CLOB, NCLOB y BFILE permiten almacenar y manipular grandes bloques de datos no estructurados (tales como texto, imágenes gráficas, clips de vídeo y sonido) en formato binario o de caracteres.
  - Ofrecen acceso aleatorio y por piezas a los datos de forma eficiente.

- Con el crecimiento de internet y el rico contenido de las aplicaciones, se ha hecho imprescindible que las bases de datos soporten un tipo de datos que cumpla lo siguiente:
  - Que pueda almacenar datos no estructurados.
  - Que esté optimizado para grandes cantidades de estos datos.
  - Que proporcione una manera uniforme de acceder a grandes cantidades de datos no estructurados dentro de la base de datos o fuera.

Oracle Text

- Oracle Text indexa cualquier documento o contenido textual para añadir rápido, recuperación precisa de información para aplicaciones de gestión de contenidos de Internet, catálogos de comercio electrónico, servicios de noticias ofertas de trabajo, etc.
  - Estos índices pueden almacenarse en archivos de sistema, bases de datos o en la Web.
- Oracle Text permite búsquedas de texto que se combinan con búsquedas regulares de la base de datos en una sentencia simple SQL.
- Puede encontrar documentos basados en su contenido textual, metadatos o atributos.
- El Oracle Text SQL API hace que sea sencillo e intuitivo crear y mantener índices Text y ejecutar búsquedas Text.

Oracle Ultra Search

- Oracle Ultra Search esta construido en el servidor de la base de datos Oracle y la tecnología Oracle Text que provee uniformemente capacidades de buscar-y-localizar sobre múltiples repositorios:
  - Base de datos Oracle, otras bases de datos ODBC compatibles, servidores mail IMAP, servidores de documentos HTML a través de un servidor Web, ficheros en disco, etc.

- Ultra Search usa un ‘rastreador’ para indexar documentos.
  - El documento permanece en su propio repositorio, y el rastreador de información es usado para construir un índice que permanece dentro de su firewall en una base de datos Oracle designada.

Oracle interMedia

- Oracle interMedia es una función que permite a la Oracle Database almacenar, gestionar y recuperar datos de imágenes, audio y video de manera integrada con otra información de la empresa.

- Oracle interMedia extiende la fiabilidad, habilidad y gestión de datos de Oracle Database los contenidos de medios tradicionales, internet, comercio electrónico y aplicaciones ricas en medios.

- interMedia administra el contenido multimedia al proporcionar lo siguiente:
  - Almacenamiento y recuperación de datos de los medios en la base de datos para sincronizar los datos de los medios con los datos empresariales asociados.
  - Soporte para imagen, audio y vídeo de formatos populares.