Objetos del *schema* de Oracle

Administración de Bases de Datos

Curso 2016-2017

Mercedes G. Merayo, Yolanda García, Jesús Correas

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

.

Contenido

- Introducción. *Schema* de una BD Oracle.
- *Constraints.*
  - NOT NULL
  - PRIMARY KEY
  - UNIQUE
  - FOREIGN KEY
  - CHECK
- Índices. Configuración de índices.
  - Índices B-Tree.
  - Índices *bitmap*.
- Tablas temporales.
- Vistas materializadas.
- Secuencias.
- Sinónimos.

Introducción. *Schema* de una BD Oracle.

- Los componentes lógicos de una BD Oracle (tablas, vistas, etc.) se crean en el contexto de un esquema.
- Cada usuario que se crea tiene asociado su propio **esquema (schema)** y tiene el mismo nombre del usuario.
- Un esquema es una **colección de objectos:**
  - Asociados a segmentos (tablas, índices, etc.).
  - Y no asociados con segmentos (vistas, constraints, sinónimos, procedimientos, etc.).
- Cada objeto en un esquema **tiene un nombre único** dentro de ese esquema (*).
- Para referirse a ellos de forma única se utiliza el nombre del esquema: **`Esquema.Objeto.`**
- Un usuario puede acceder a un objeto de un esquema distinto del suyo si tiene privilegios sobre él y utilizando el nombre completo.

Denominación de objetos en el esquema

- Los nombres de los schema objects no distinguen minusculas y mayusculas
  - excepto si van con dobles comillas: `dual` y `system` lo mismo que `"DUAL"` Y `"SYSTEM"`.
  - Internamente todo está guardado en mayúsculas.
- (*) Dentro de un esquema se puede utilizar el mismo nombre para objetos en **distintos namespaces:**
  - Se encuentran **en el mismo namespace:** Tablas, vistas, secuencias, sinónimos privados, procedimientos, funciones, paquetes, vistas materializadas y tipos definidos por el usuario.
  - Cada uno de los siguientes objetos tienen **su propio namespace:** Indices, restricciones, clusters, triggers, database links, dimensions.

Esquemas SYS y SYSTEM

- Durante la creación de la base de datos se crean los usuarios **SYS** y **SYSTEM** y sus esquemas.
- El diccionario de datos está en el esquema **SYS**, que es su propietario.
- El esquema del usuario **SYS** tambien almacena paquetes PL/SQL que son utilizados por los administradores y desarrolladores.
- Los objetos del esquema SYS no se modifican con instrucciones DML: se utilizan instrucciones específicas DDL.
- El esquema **SYSTEM** almacena objetos adicionales de administración y monitorización.

Constraints

- Las **restricciones (constraints)** de una tabla son objetos que permiten que la base de datos mantenga **la consistencia de la información.**
  - validando el cumplimiento de las reglas de negocio planteadas por el modelo de datos.
- Tipos de constraints:
  - **UNIQUE**
  - **NOT NULL**
  - **PRIMARY KEY**
  - **FOREIGN KEY**
  - **CHECK**
- Como cualquier objeto tienen asociado un nombre.

Constraints - NOT NULL, PRIMARY KEY, UNIQUE

- **NOT NULL constraints**
  - Exigen que se almacene informacion en la columna asociada
  - Se pueden asignar valores por defecto (clausula DEFAULT)
- **PRIMARY KEY Constraints**
  - Implementan el concepto de **clave primaria** del modelo relacional.
  - Además garantiza la unicidad de filas en una tabla.
  - Todas las tablas requieren una primary key en el modelo relacional, pero Oracle permite definer tablas sin primary key.
  - Solo hay una primary key por tabla.
  - Consta de uno o varios atributos que no pueden almacenar NULL.
- **UNIQUE constraints**
  - Constan de uno o varios atributos.
  - Pueden almacenar valores nulos.
  - Oracle crea un indice asociado a los atributos que permite chequear la existencia de valores repetidos en caso de inserciones.
  - Además el índice incrementa el rendimiento cuando las columnas se utilizan en la clausula WHERE.

Constraints - FOREIGN KEY, CHECK

- **FOREIGN KEY constraints**
  - Define **claves externas** en relaciones padre-hija.
  - Se definen en la tabla hija, con atributos que referencian a la clave primaria de la tabla padre.
  - Tambien pueden usarse claves únicas. En este caso, se permite incluir valores NULL.
  - **Se comprueba:**
    - ★ Las filas de la tabla hija deben referirse a filas existentes en el padre.
    - ★ Al borrar una fila en la tabla padre no debe estar asociada a ninguna fila de la tabla hija.
  - Se puede evitar con `ON DELETE CASCADE` y `ON DELETE SET NULL`.
  - No se puede eliminar la tabla padre mientras exista la tabla hija.
- **CHECK constraints**
  - Permite indicar condiciones que deben ser cumplirse cuando se le da valor a un atributo.
  - Pueden hacer referencia a otros atributos de la tabla.
  - No se puede usar SYSDATE.

Constraints - Estado y Comprobación de constraints

- Una constraint puede estar en distintos **estados:** enabled/disabled, y validated/notvalidated.
  - **ENABLE VALIDATE**
    - ★ Las filas nuevas deben cumplir la constraint.
    - ★ Todas las filas que están en la tabla la cumplen.
  - **DISABLE NOVALIDATE**
    - ★ Pueden insertarse filas que no cumplan la constraint.
    - ★ Puede haber filas en la tabla que no la cumplan.
  - **ENABLE NOVALIDATE**
    - ★ Las filas nuevas deben cumplir la constraint.
    - ★ Puede haber filas en la tabla que no la cumplan.
- Las constraints se pueden comprobar a nivel de sentencia (**IMMEDIATE**) o cuando se confirma la transacción (**DEFERRED**).
- **Por ejemplo:** si un proceso inserta o actualiza filas en ambas tablas y la clave externa no es diferible, el proceso fallará si no se procesa en el orden correcto.
- Por defecto: **enabled y validated, y no son deferrable.**

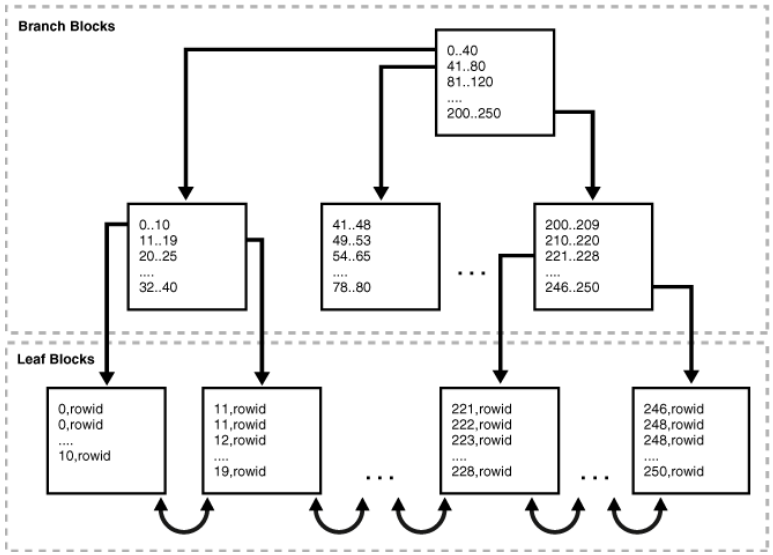
Índices

- Tienen dos funciones: **forzar constraints y mejorar el rendimiento.**
- Un índice permite el acceso casi inmediato a los valores de los atributos para comprobar su existencia.
- Pero **reduce el rendimiento** cuando ejecutamos sentencias DML: se deben actualizar los índices.
- Los índices son **independientes de las tablas:** se pueden crear nuevos índices o destruir los antiguos sobre tablas existentes.
- **Índices y constraints:**
  - Se generan índices de forma automática para las **primary key** y **unique** constraints.
  - En el caso de las **foreign keys** el índice existe en la tabla padre para chequear la existencia de los valores.
  - Sin embargo, es conveniente crear un índice en los atributos de la foreign key de la tabla hija **por motivos de eficiencia.**
- Hay dos tipos de índices (entre otros): B-Tree y bitmap.

Índices B-Tree

- Un **índice B-Tree** es una estructura de árbol.
- El tipo de índice por defecto.
- Tiene dos tipos de bloques: **ramas y hojas.**
  - El nodo raíz tiene rangos de valores que apuntan a bloques del nivel inferior.
  - Cada uno de estos bloques apunta a hojas con valores que están en los diferentes rangos con el rowid de la fila correspondiente.

- El `rowid` es una pseudocolumna que tienen todas las tablas y en la que se encripta la **dirección física** de la fila.



Índices B-Tree

- Los índices B-Tree se deben utilizar si el número de valores diferentes en el atributo es alto.
- Otros parámetros de configuración de un índice B-Tree son:
  - **Reverse key:** invierte los bytes de los valores de las columnas del índice. Para resolver problemas de contención en la actualización del índice.
  - **Unique/non-unique:** para impedir valores repetidos del índice.
  - **Compressed:** almacenan los valores repetidos solo una vez, seguido de la secuencia de rowids de las filas.
  - **Ascending/Descending:** para que alguna columna se ordene descendentemente.
  - **Composite:** Índices sobre más de una columna. El **orden de las columnas** es relevante.
    Primero la columna que más se utilice (y que discrimine más).
  - Function-based: se construye en base a la aplicación de una función a uno o mas de los atributos del índice, por ejemplo `upper(apellido)` o `to_char(fecha_fin,'yy-m-dd')`.

Índices *bitmap*

- Almacena los rowids asociados con cada valor como un bitmap.
- Incluye los valores NULL.
- **Ejemplo:** TIENDA, PRODUCTO, FECHA y RECOGIDA almacenan información de las compras en una cadena de supermercados

```
RECOGIDA=EnTienda   11010111000101011101011101.....
RECOGIDA=Domicilio  00101000111010100010100010.....
TIENDA=Madrid       11001001001001101001010000.....
TIENDA=Cuenca       00100010011000010001001000.....
TIENDA=Zamora       00010001000100000100100010.....
TIENDA=Lugo         00000100100010000010000101.....
```

- En una consulta como: `select count(*) from ventas where RECOGIDA = 'EnTienda' and TIENDA = 'Madrid'`:

```
'EnTienda&Madrid'   11000001000000001001010000...
```

Índices *bitmap*

- El tamaño del índice depende de la **cardinalidad de la columna** y de la **distribución de los datos.**
  - Si una columna toma pocos valores (género) un *bitmap* ocupa mucho menos que un índice B-tree.
  - Si toma valores únicos (como DNI) será conveniente usar un índice B-tree.
- En general, se deben usar índices *bitmap* cuando el ratio de valores diferentes en el atributo clave y el número de filas es bajo.
- Cuando el número de valores aumenta el tamaño del bitmap lo hace exponencialmente y su rendimiento decrece en el mismo orden.
- Oracle almacena rangos de rowids en los indices bitmap.
  - No es recomendable si se modifica la columna frecuentemente ya que provoca problemas de bloqueos de rangos de filas.

Índices

- Durante la creación de las tablas e índices se pueden indicar diferentes parámetros asociados al almacenamiento.
  - **TABLESPACE:** Se puede determinar el tablespace en el que se va a crear el objeto de BD.
  - **PCTFREE:** Espacio disponible en cada bloque para modificaciones de filas existentes (valor mayor supone mayor espacio consumido, pero más eficiencia en determinados UPDATEs).
  - **PCTUSED:** Espacio ocupado al que debe bajar el bloque para permitir nuevas inserciones (valor mayor supone menor espacio consumido pero menor eficiencia en INSERT y UPDATE).
  - **INITRANS:** Espacio disponible en el bloque para mantener la información de transacciones que lo modifican (concurrentes).
  - **STORAGE** (INITIAL NEXT PCTINCREASE MINEXTENTS MAXEXTENTS BUFFER_POOL...): Información de almacenamiento: Tamaños y número de extents del segmento donde se almacena el índice, etc.

Tablas Temporales

- Una tabla temporal tiene una estructura común para todas las sesiones, pero las filas son accesibles solo por la session que las ha insertado. `CREATE GLOBAL TEMPORARY TABLE temp_tab_name`

```
(column datatype [,column datatype...] )
[ON COMMIT DELETE | PRESERVE ROWS] ;
```

- Difieren de las tablas regulares en que son transitorias:
  - Están en la PGA del proceso, no en un segmento de un tablespace, por lo que no se accede a disco.
  - Los commandos SQL que se ejecutan sobre ellas son mucho más rápidos.

Otros objetos del esquema

- **Vistas Materializadas:** es el resultado de una consulta que se almacena en una tabla real, que se actualiza periódicamente.
  - Acceso más eficiente, pero se incrementa el tamaño de la base de datos.
  - Posible falta de sincronización
- **Secuencias:** proporcionan una serie numérica secuencial.
- **Sinónimos:** son alias de un objeto de un esquema, como una vista o una tabla.
  - Pueden ser publicos (no es necesario hacer referencia al esquema propietario) o privados (necesario referenciar al esquema si no eres el propietario).