SCHEMA OBJECTS

ADMINISTRACIÓN DE BASES DE DATOS

JESUS CORREAS Y MERCEDES G. MERAYO

SCHEMA OBJECTS

- Cada usuario que se crea tiene asociado su propio schema y tiene el mismo nombre.
- Un **esquema** es una colección de los objectos (tablas, índices,vistas, constraints, sinónimos, procedimientos…) que pueden ser creados y manipulados mediante sentencias SQL.
- Cada objeto en un esquema tiene un **nombre único dentro de ese esquema**. El mismo nombre puede ser compartido por diferentes objetos en diferentes esquemas. Para referirse a ellos de forma unica se utiliza el nombre del esquema: **Esquema.Objeto**.
- Hay usuarios que no crearan nunca objetos, y por tanto su esquema siempre estará vacio.
- Durante la creación de la base de datos se crean los usuarios SYS y SYSTEM y los esquemas asociados. Se pueden crear otros dependiendo de las opciones seleccionadas.

SCHEMA OBJECTS

- Los nombres de los objectos no distinguen minusculas y mayusculas excepto si van entrecomillados.

  CREATE TABLE "Tabla2b" ("Col1" VARCHAR2(100))

  SELECT Col1 FROM tABLE2B falla, la correcta sería SELECT "Col1" FROM "Tabla2b"

- Aunque en general es cierto que los objetos se identifican con el usuario y el nombre, objetos en distintos **namespaces** pueden tener el mismo nombre.

- Un namespace es un grupo de tipos de objetos.

- 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

- El usuario SYS es propietario del diccionario de datos, constituido por un conjunto de tablas que definen la base de datos , y por tanto se encuentra en su esquema.

- El esquema del usuario SYS tambien almacena paquetes PL/SQL que son utilizados por los administradores y desarrolladores de la base datos.

- Los objetos del esquema SYS no deben ser modificados directamente mediante instrucciones DML, ya que se corre el riesgo de corromper la información que tendrian consecuencias no deseables.

- El esquema SYSTEM almacena objetos adicionales para la administración y monitoreo de la base de datos.

DICCIONARIO DE DATOS

- El diccionario de datos contiene “metadatos”, informacion sobre lo que hay en la base de datos.
- Los datos del diccionario pueden ser accedidos a través de vistas que se dividen en
    - Vistas **DBA, ALL or USER** para tratar con las estructuras de la base de datos.
        - Las que comienzan con USER solo muestran información de los objetos del usuario conectado.
        - Las que comienzan por ALL ven información de todos los objetos a los que tiene acceso el usuario.
        - Las vistas que comienzan por DBA acceden a todos los objetos. Solo accesibles por el usuario DBA.
    - **V$** Dynamic Performance Views para monitorizar estadísticas de la actividad de la base de datos en tiempo real

DICCIONARIO DE DATOS

USER_TABLES (TABS)	Descripción información tablas del usuario
USER_CATALOG (CAT)	Tablas, Vistas, Sinónimos y Secuencias del usuario
USER_COL_COMMENTS	Comentarios de columnas
USER_CONSTRAINTS	Definiciones de restricciones en tablas
USER_INDEXES (IND)	Información de los índices
USER_OBJECTS (OBJ)	Todos los objetos del usuario
USER TAB_COLUMNS (COLS)	Columnas de las tablas y vistas del usuario
USER_TAB_COMMENTS	Comentarios asociados a las tablas
USER_TRIGGERS	Triggers definidos por el usuario
USER_USERS	Información sobre el usuario
USER_VIEWS	Vistas del usuario

DICCIONARIO DE DATOS

DBA_TABLES	Descripción tablas de todos los usuarios
DBA_CATALOG	Tablas, Vistas, Sinónimos y Secuencias
DBA_OBJECTS	Todos los objetos de todos los usuarios
DBA_USERS	Información sobre todos los usuarios
DBA_DATA_FILES	Información sobre data files

SCHEMA OBJECTS - CONSTRAINTS

- Las **constraints** de una tabla 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 y CHECK

- Como cualquier objeto tiene asociado un nombre.

SCHEMA OBJECTS - CONSTRAINTS

- **Not Null Constraints**
- Exigen que se almacene informacion en la columna asociada
- Se pueden asignar valores por defecto (clausula DEFAULT)

- **Primary Key Constraints**
- Restriccion de unicidad de filas en una tabla.
- Todas las tablas requieren una primary key en el modelo relacional, pero Oracle permite definir tablas sin primary key.
- Solo hay una primary key por tabla.
- Consta de uno o varios atributos clave que contendrán valores únicos no pudiendo almacenar NULL

SCHEMA OBJECTS - CONSTRAINTS

- **Unique constraints**: Constan de uno o varios atributos. Pueden almacenar valores nulos, lo que no es posible en el caso de las primary keys.

- Oracle crea un indice asociado a los atributos tanto de las claves primarias como de las claves únicas, que permite chequear la existencia de los valores en caso de inserciones para rechazarlos en caso de duplicidad.

- El indice permite incrementar el rendimiento cuando las columnas se utilizan en la clausula WHERE de las sentencias SQL.

SCHEMA OBJECTS - CONSTRAINTS

- **Foreign Key Constraints**
- Una foreign key se define en la tabla hija de una relación padre-hija.
- Consta de uno o varios atributos que corresponden a los de la primary key de la tabla padre y deben coincidir en tipo de datos
- Tambien pueden usarse claves únicas para asociarlas a la foreign key. En este caso, se permite incluir valores NULL en los atributos de la foreign key.
- Se rechazarán inserciones en la tabla hija que tengan valores en los atributos de la foreign key que no existan en la tabla padre. Tampoco se puede borrar una fila en la tabla padre que esté asociada a alguna 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

SCHEMA OBJECTS - CONSTRAINTS

- **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.
- **Estado Constraints**
- Una constraint puede estar **enabled**/**disabled**, y **validated**/**notvalidated**.
  - **ENABLE VALIDATE** no se pueden dar de alta filas que no cumplan la constraint y todas las que están en la tabla la cumplen
  - **DISABLE NOVALIDATE** pueden insertarse filas que no cumplan la constraint y puede haber filas en la tabla que tampoco la cumplan.
  - **ENABLE NOVALIDATE** Las filas nuevas deben cumplir la constraint pero en la tabla puede haber filas que no la cumplan

SCHEMA OBJECTS - CONSTRAINTS

```
create table valores(
   id number,
   valor varchar2(50),
   descrip varchar2(300),
   CONSTRAINT PK_VALORES PRIMARY KEY (id) )
```

Enable Validate

```
alter table valores disable novalidate primary key;
```

Disable NoValidate

```
insert into valores values (1,'ELEMENTO UNO','Primer elemento, tabla vacía'
);

 insert into valores values (1,'ELEMENTO UNO','Primer elemento, repetido');

 insert into valores values (1,'ELEMENTO UNO','Primer elemento, otra vez re
petido');

alter table valores enable novalidate primary key;
```

Enable NoValidate

```
insert into valores values (1,'ELEMENTO UNO','Primer elemento, violando P
K, repetido');

ERROR en línea 1:
ORA-00001: restricción única (MGM.PK_VALORES) violada
```

```
select * from valores;

        ID VALOR                            DESCRIP
---------- ---------------------------------- ------------------------------------
         1 ELEMENTO UNO                       Primer elemento, tabla vacía
         1 ELEMENTO UNO                       Primer elemento, repetido
         1 ELEMENTO UNO                       Primer elemento, otra vez repetido
```

SCHEMA OBJECTS - CONSTRAINTS

```
truncate table valores;

alter table valores disable  novalidate primary key;

insert into valores values (1,'ELEMENTO UNO', 'Primer elemento, tabla vacía');
insert into valores values (2,'ELEMENTO DOS', 'Segundo elemento, ID distinto.');
insert into valores values (3,'ELEMENTO TRES', 'Tercer elemento, ID distinto.');

alter table valores disable validate primary key;

insert into valores values (3,'ELEMENTO TRES','Tercer elemento, ID existente.');
*
ERROR en línea 1:
ORA-25128:  No se puede insertar/actualizar/suprimir en la tabla con la restricción

(MGM.PK_VALORES) desactivada y validada

insert into valores values (4,'ELEMENTO CUATRO','Cuarto elemento, ID distinto.');

ERROR en línea 1:
ORA-25128: No se puede insertar/actualizar/suprimir en la tabla con la restricción (MGM.PK_VALORES) desactivada y validada
```

Disable NoValidate

Disable Validate

No chequea la constraint, no puede asegurar que se mantiene el estado VALIDATE

SCHEMA OBJECTS - CONSTRAINTS

- **Chequeo de Constraints**
- Las constraints se pueden chequear a nivel de sentencia (**IMMEDIATE** constraint) o cuando se confirma una transacción (**DEFERRED** constraint).
- Para que una constraint sea ***deferrable*** se debe indicar en su creación.
- Un uso común de constraints deferred es el caso de las foreign keys: si un proceso inserta o actualiza filas en ambas tablas y la foreign key constraint no es deferreable el proceso fallará si no se procesa en el orden correcto.
- Por defecto las constraints son enabled y validated, y no son deferrable.

SCHEMA OBJECTS - CONSTRAINTS

```
alter table valores enable validate primary key;

create table valoresHija(

idHija number,

idPadre number,

descrip varchar2(300),

CONSTRAINT PK_VALORESII PRIMARY KEY (idHija),

CONSTRAINT FK_VALORES  FOREIGN KEY (idHija) REFERENCES valores initially immediate);

insert into valoresHija values (1,3,'Hija de Valor 3');

insert into valoresHija values (2,4,'Hija de Valor 4');
```

IMMEDIATE

Error SQL: ORA-02291: restricción de integridad (SYSTEM.SYS_C0013519) violada - clave principal no encontrada

```
set constraint FK_VALORES deferred;
```

Error SQL: ORA-02447: no se puede diferir una restricción que no es diferible

SCHEMA OBJECTS - INDICES

- **Indices**
- Son estructuras asociadas a las tablas.
- Tienen dos funciones: **forzar constraints y mejorar el rendimiento**.
- Se generan de forma automática para las primary key y unique constraints sobre las columnas que las definen.
- En el caso de las foreign keys el índice existe en la tabla padre, no en la tabla hija, 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: si se borra una fila en la tabla padre será más rapido para Oracle usar un índice para determinar si existen filas en la tabla hija referenciando a la fila que se está borrando.

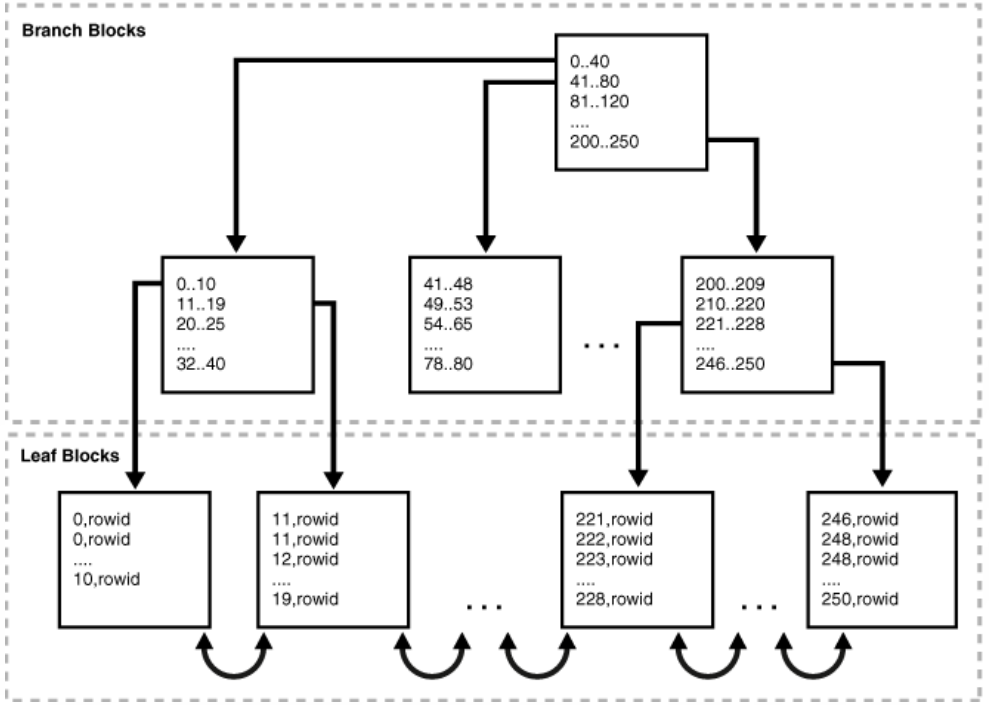
SCHEMA OBJECTS - INDICES

- **Indices**

- Un índice permite el **acceso inmediato a los valores de los atributos clave,** para chequear casi de inmediato su existencia.
- Son muy útiles con clausulas que requieren la **ordenación de los datos** (ORDER BY, GROUP BY…)
- El uso de índices, sin embargo, reduce el rendimiento cuando ejecutamos sentencias DML, ya que se deben actualizar los índices.
- Hay situaciones en las que un recorrido completo de la tabla (tablas pequeñas o consultas que recuperan una gran proporcion de los datos de una tabla) puede ser más rápido.
- Hay dos tipos de índices: **B*Tree index y bitmap.**

SCHEMA OBJECTS - INDICES

- **Indices 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 que contienen valores de las columnas del índice (en las filas de la tabla asociada) que están en los diferentes rangos junto con el **rowid** de las mismas.
- El ***rowid***, *una pseudocolumna* **,** que tienen todas las tablas y en la que se encripta la dirección física de la fila.
- Las filas con valores NULL en los atributos del índice no se almacenan.



SCHEMA OBJECTS - INDICES

- **Indices B*Tree**
- Los índices B*Tree se deben considerar si el volumen de valores diferentes en el atributo es alto, si el volumen de filas de la tabla es elevado y si la columna es utilizada en las consultas.
- **Unique/non-unique:** Un índice único no permitirá que dos filas contengan los mismos valores en la columnas clave del índice.
- **Compressed**: almacenan los valores repetidos solo una vez, seguido de la secuencia de rowids de las filas que contienen dicho valor.
- **Reverse key** invierte los valores de las columnas del índice
  - Permite resolver problemas de contención en las hojas de la parte derecha del índice cuando se accede repetidamente para realizar actualizaciones del mismo bloque.

SCHEMA OBJECTS - INDICES

- **Indices B*Tree**

- Por ejemplo, supongamos que el valor de la columna indexada de una fila es 123456, Oracle lo invertirá 654321 antes de insertarlo en el índice. Al invertir el siguiente valor 123457 insertará 754321 en el índice y así sucesivamente. Este hace que los insert se repartan uniformemente en la estructura del indice.

- No obstante conllevan otros problemas:
  - Si se busca un rango (BETWEEN <, >, <=, >=, LIKE) los valores no estarán contiguos. Oracle ignora el índice.

SCHEMA OBJECTS - INDICES

- **Ascending/Descending:** Por defecto son ascendentes pero puede interesar que alguna columna se ordene descendentemente.

- **Composite:** Índices sobre más de una columna. El orden en el que se indican en la definición dependerá del uso en las consultas y es importante.

    - **Si tenemos un índice sobre las columnas A y B de una tabla y en una consulta no se incluye la columna A Oracle no utilizará el índice.**

- **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'). Para que se use el índice la consulta debera aplicar la misma función.

SCHEMA OBJECTS - INDICES

- **Indices Bitmap**

- Un indice bitmap almacena los rowids asociados con cada valor clave como un bitmap.

- Una ventaja de este tipo de índices es que al contrario que los índices B*Tree si incluye los valores NULL.

- En general, los indices bitmap se deben usar cuando el ratio de valores diferentes en el atributo clave y el número de filas es bajo.

SCHEMA OBJECTS - INDICES

- **Indices Bitmap**
  - TIENDA, PRODUCTO, FECHA y RECOGIDA almacenan información de las compras en una cadena de supermercados
  - Las consultas habituales comparan ventas de diferentes tiendas con un tipo especifico de reparto.
  - Un índice bitmap almacena los rowids associados con cada valor como un bitmap.
    - RECOGIDA= Tienda 11010111000101011101011101.....
    - RECOGIDA= Domicilio 00101000111010100010100010.....
    - TIENDA= Madrid 11001001001001101001010000.....
    - TIENDA= Cuenca 00100010011000010001001000.....
    - TIENDA= Zamora 00010001000100000100100010.....
    - TIENDA= Lugo 00000100100010000010000101.....

SCHEMA OBJECTS - INDICES

- **Indices Bitmap**

Si se hace una consulta como

```
select count(*) from ventas where recogida= ' Tienda' and tienda='Madrid';
```

Oracle puede recuperar los dos bitmaps involucrados y operar con ellos

- Tienda 11010111000101011101011101
- Madrid 11001001001001101001010000
- Tienda y Madrid 11000001000000001001010000

SCHEMA OBJECTS - INDICES

- El tamaño del índice depende de la cardinalidad de la columna así como de la distribución de los datos.
  - Si una columna toma pocos valores (por ejemplo, genero) un bitmap ocupará mucho menos que un índice B*tree sobre dicha columna.
  - Si por el contrario toma valores únicos (como DNI) será conveniente usar un índice B*tree.
- 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 lo que no los hace recomendables si **el nivel de actualizacion** de la columna correspondiente es alto ya que provoca problemas de bloqueos de rangos de filas.

SCHEMA OBJECTS – TABLAS TEMPORALES

- **Tablas Temporales**
- Una tabla temporal tiene una **estructura común para todas las sesiones**, pero las filas son accesibles sólo por la sesión 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 la transitoriedad de los datos, lo que hace que los commandos SQL que se ejecutan sobre ellas sean mucho más rápidos. Están en la PGA (Program global area), no en un segmento de un tablespace, por lo que no se accede a disco.
- Se utilizan para operaciones que requieren manipular grandes cantidades de datos. Durante la operación los datos se almacenan en una tabla temporal.

SCHEMA OBJECTS

- **Vistas Materializadas** es el resultado de una consulta que se almacena en una tabla real, que se actualiza periódicamente.
  - Acceso más eficiente con un incremento en 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).