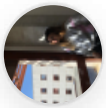


beacc
www.wuolah.com/student/beacc

1551

Tema 3 - ABD.pdf

Apuntes Tema 3

3° Administración de Bases de Datos

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Tema 3 - Organización de los datos en un SGDB

ESTRUCTURA INTERNA DE UN SGDB RELACIONAL

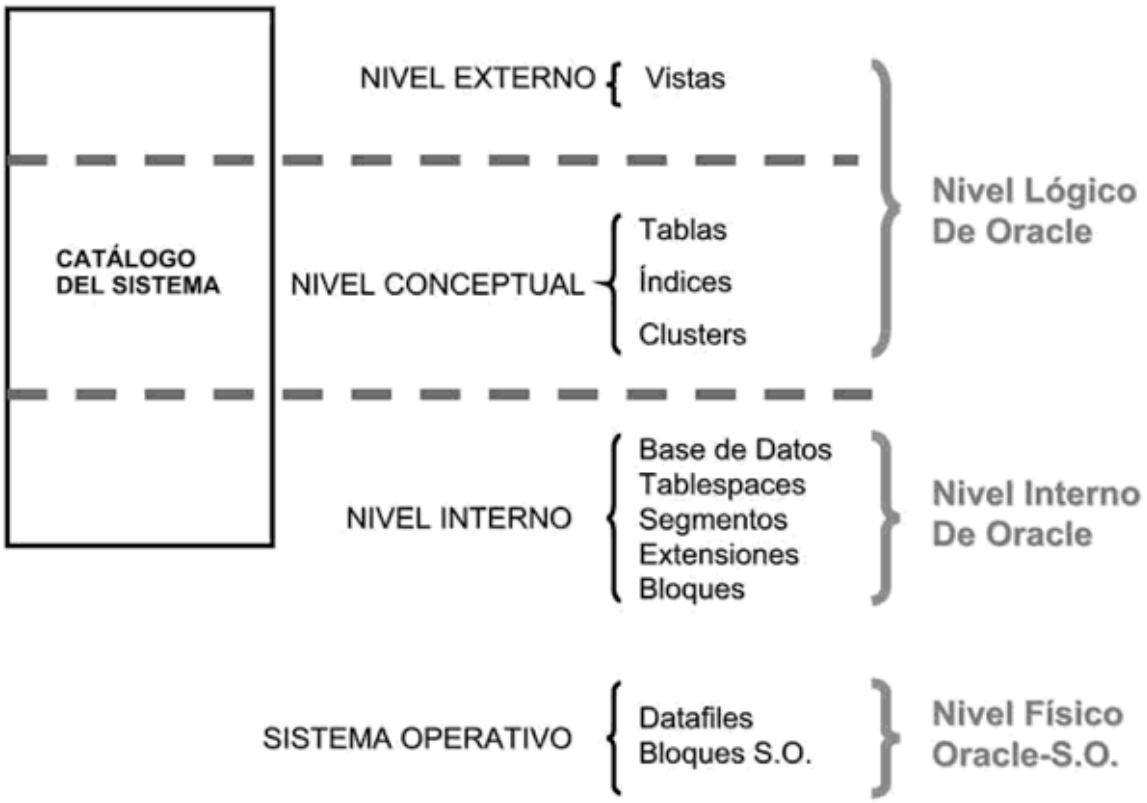


FIGURA 3.2 Estructura por niveles en Oracle.

El catálogo o diccionario de datos

Podemos definir el catálogo como un **repositorio** en el que se encuentra **toda la información** acerca de la **base de datos**, usuarios y su privilegios, tablas, correspondencia entre niveles… Estos datos se encuentran organizados y jerarquizados y su almacenamiento y mantenimiento depende del SGDB.


CUNEF
Lidera tu futuro.
Define tu éxito.
POSTGRADO EN
DATA SCIENCE
PARA FINANZAS
SÚMATE AL ÉXITO
Excelencia, futuro, éxito.
www.cunef.edu

El catálogo tiene 3 **utilidades básicas:**

- **Organizar** toda la **información** de los **usuarios**, **objetos** y **esquemas** de **almacenamiento** de la BD.
- **Mantener** siempre **actualizada** dicha información.
- **Permitir acceso de solo lectura** a cualquier usuario autorizado con el objetivo de obtener información acerca de la BD.

La organización interna de Oracle cuenta con las siguientes estructuras de almacenamiento, ordenadas según su nivel de abstracción:

- Tablespaces -> Datafile SO
- Segmentos
- Extensiones
- Bloques -> Bloques SO

Tablespaces o ficheros de datos

Los tablespaces, son estructuras lógicas que nos permiten organizar y almacenar objetos de la base de datos relacionados entre sí (por su contenido, su función, su propietario…)

Podríamos comparar un tablespace con un directorio.

Segmentos , Extensiones y Bloques

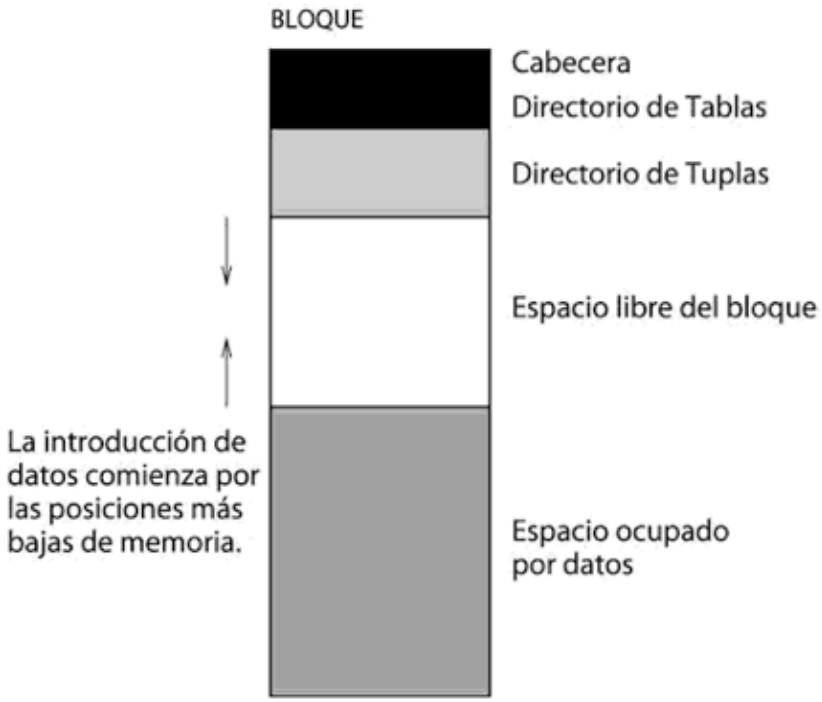
Los **segmentos** almacenan datos de un mismo objeto ayudando así a clasificar y ordenar la información de un tablespace.

Cada vez que se crea una tabla, se crea un **segmento** que contiene una extensión (fragmento de memoria contigua). Un **segmento** esta formado por una o más **extensiones** que se van enlazando mediante punteros.

Una **extensión** esta compuesta por un conjunto de **bloques**. Definimos un **bloque** como la unidad lógica de almacenamiento más pequeña y por lo tanto, el nivel más bajo al que el SGDB puede administrar sus datos. Su tamaño debe de ser un múltiplo del tamaño del bloque del SO para optimizar las operaciones de lectura/escritura en disco.

En Oracle podemos distiguir 4 **tipos de segmentos** diferentes:

- **Segmento de datos** (tablas)
- **Segmento de índices** (índices)
- **Segmentos temporales**. Se generan durante operaciones como Order by, Group by, Union… en las que se necesita espacio para almacenar datos intermedios. Estos segmentos se destruyen cuando el resultado ya ha sido calculado
- **Segmentos de rollback**. Estos segmentos sirven para almacenar los valores antiguos de los datos antes de ser modificados, con el fin de poder recuperar la base de datos en caso de fallo de una transacción (rollback).



Estructura de un bloque

Cabecera: La cabecera contiene información general del bloque como su dirección y el tipo de segmento al que pertenece.

Directorio de Tablas: Esta zona guarda información de las tablas que tienen filas en este bloque.

Directorio de Tuplas: Esta zona guarda información acerca de las filas almacenas en el bloque, incluyendo su dirección.

Zona de datos: Es la zona donde se almacena la información de la BD, los registros.

ESTRUCTURA LÓGICA DE UN SGDB RELACIONAL

El conjunto de objetos de un usuario se denomina **esquema** y está compuesto por :

-tablas, vistas, índices, clústers…

-procedimientos, funciones, paquetes

-disparadores…

TABLA

Una tabla es una estructura lógica formada por **columnas** (atributos) y **filas** (tuplas).

Al crear una tabla se crea un **segmento** con una **extensión** y se adjudica a un **tablespace** por defecto.

Si una fila no cabe en un bloque se genera otro bloque en la extensión, enlazado al anterior, quedando así la tupla dividida.

Cada fila tiene un **rowid único** e invariable (rowid es una cadena de texto alfanumérica que identifica de forma única cada fila de cada tabla).

Una tabla se crea con la sentencia CREATE TABLE nombre_tabla (atributos)

VISTA

Una vista es una **presentación de datos** proveniente de una o más **tablas/ vistas**, hecha a medida del usuario. Se le asigna un nombre a el **resultado** de una **consulta** para poder ser utilizado dicho resultado como si se tratase de una tabla.

Normalmente las vistas no están almacenadas físicamente, si no que se encuentran en el catálogo y se reconstruyen cada vez que sea necesario.

Se crean mediante el comando CREATE VIEW

Ejemplo:

CREATE VIEW Alumnos_gr AS SELECT nombre, apellidos, dni FROM alumnos WHERE provincia='Granada' ;

Obteniendo así de la tabla alumnos una vista de los alumnos que son de Granada.
Las vistas nos permiten tener **más seguridad** en nuestro sistema, ya que nos permiten ocultar información, hacer visible al usuario solo la parte de la BD que necesite.

Tras la ejecución de la consulta anterior se insertará una fila nueva en el catálogo de la base de datos, que se podrá consultar a través de DBA_VIEWS:

> **dba_views** (**owner, name ,text**)
>
> *donde owner es el propietario de la vista, name es el nombre que se le asignó a dicha vista y text es la sentencia select que permite reconstruirla.*

Una vista se elimina mediante la sentencia DROP VIEW

Generalmente las vistas no son actualizables, salvo que cumplan algunos requisitos:

- No puede incluir **cláusulas de agregamiento** (GROUP BY) o **funciones de agregación** (MAX, COUNT, AVG…)
- No puede incluir la cláusula DISTINCT
- No puede incluir la reunión ni operadores de conjuntos.
- Todos los atributos NOT NULL deben estar en la vista

Además de por sus ventajas en seguridad las vistas se utilizan para abstraer la complejidad de la estructura de los datos, simplificar comandos, consultas complejas , aislar aplicaciones de los cambios…

Las vistas son el principal mecanismo para implementar el **nivel externo** de la arquitectura ANSI/SPARC.

INDICE

Un **índice** es una estructura de datos definida sobre una columna de tabla (o varias) y que permite localizar de forma rápida las filas de la tabla en base a su contenido en la columna indexada además de permitir recuperar las filas de la tabla ordenadas por esa misma columna.

Oracle crea por defecto un índice para la clave primaria de cada tabla.

Los índices agilizan el acceso a los datos pero ocupan espacio y a su vez hacen que las inserciones y modificaciones sean más lentas (al tener que actualizar el índice con ellas).

Para crear un índice se usa la sentencia CREATE INDEX

CREATE INDEX *index_name*
ON *table_name* (*column1*, *column2*, …);

Y para borrarlos se usa la sentencia DROP INDEX.

¿Cuándo se utilizan los índices?
- Buscar registros por valores específicos
- Recorrer una tabla en un orden distinto al físico (order by)
- Buscar registros en un rango

Un índice con más de un atributo solo es útil cuando se consulta de forma ordenada por dichos atributos.

Es importante que si no se utilizan, o no mejoran la eficiencia se borren.

Para consultar los índices en el catálogo de la base de datos se hará mediante: DBA_INDEXES y DBA_IND_COLUMNS

CLÚSTER

Es una **estructura de almacenamiento** que sirve para guardar tablas que comparten campos y a las que se accede de forma conjunta (reunión natural). Almacena fisicamente la **reunión natural** (álgebra relacional) de dos tablas.

Un clúster deja de ser eficiente si se accede a cada tabla de forma individual , también cuando se modifican frecuentemente las tablas que forman el clúster.

Sus ventajas son que mejoran la reunión natural, al reducir los accesos a discos y también ahorran espacio pues los datos compartidos sólo se almacenan una vez.

Un clúster se crea con la sentencia CREATE CLUSTER. Después se crea cada tabla del clúster con su correspondiente sentencia CREATE TABLE añadiendo al final CLUSTER <nombre-clúster> (campos de reunión) ; Antes de insertar los datos hay que crear un índice sobre el clúster con la sentencia: CREATE INDEX <nombre> ON CLUSTER <nombre>;

El clúster se borra con la sentencia DROP CLUSTER, borrando en primer lugar las tablas o utilizando la sentencia INCLUDING TABLES. También es necesario borrar las claves externas o incluir la sentencia INCLUDE CONSTRAINTS.

Por último se pude borrar su índice con DROP INDEX pero no se podrá acceder al contenido del clúster si no se reconstruye dicho índice.

Ejemplo de creación de un clúster:

Creamos el cluster:

```
CREATE CLUSTER prov_ventas (codpro char(3) );
```

Creamos las tablas que estarán en el cluster:

TABLA PROVEEDOR

```
CREATE TABLE proveedor2 ( codpro char(3) ,nompro varchar2(30),
status number(2) ,ciudad varchar2(15) ) CLUSTER prov_ventas (codpro);
```

TABLA VENTAS

```
CREATE TABLE ventas2 ( codpro char(3), codpie char(3), codpj char(3),
cantidad number(4)) CLUSTER prov_ventas (codpro);
```

Creamos el índice sobre el cluster :

```
CREATE INDEX  indx_prov_ventas   ON CLUSTER  prov_ventas ;
```