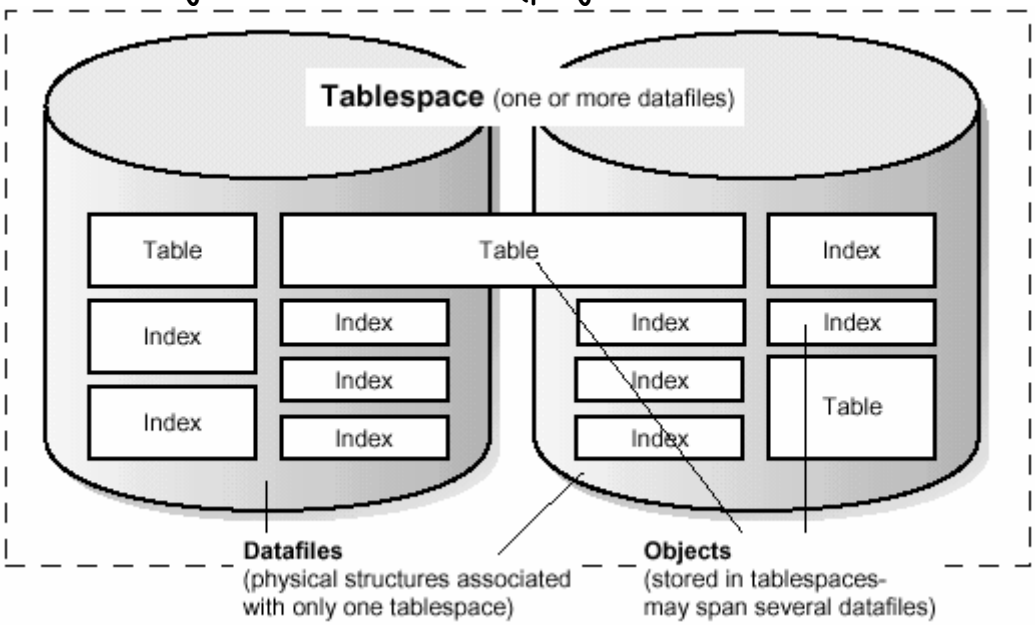
Estructura de una BD Oracle.

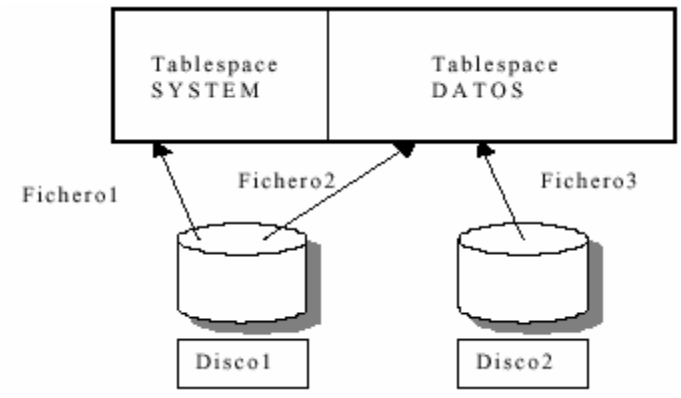
Una BD Oracle tiene una estructura física y una estructura lógica que se mantienen separadamente.

- La estructura física se corresponde a los ficheros del sistema operativo: de datos (**datafiles**), de **redo log** y de control (**controlfiles**).
- La estructura lógica está formada por los **tablespace** y los **objetos** de un esquema de BD (tablas, vistas, índices,...).

Estructura lógica.

Una BD se divide en unidades de almacenamiento lógicas: **Tablespaces**.
Contienen distintos objetos relacionados (p.ej. todas las tablas de una aplicación).

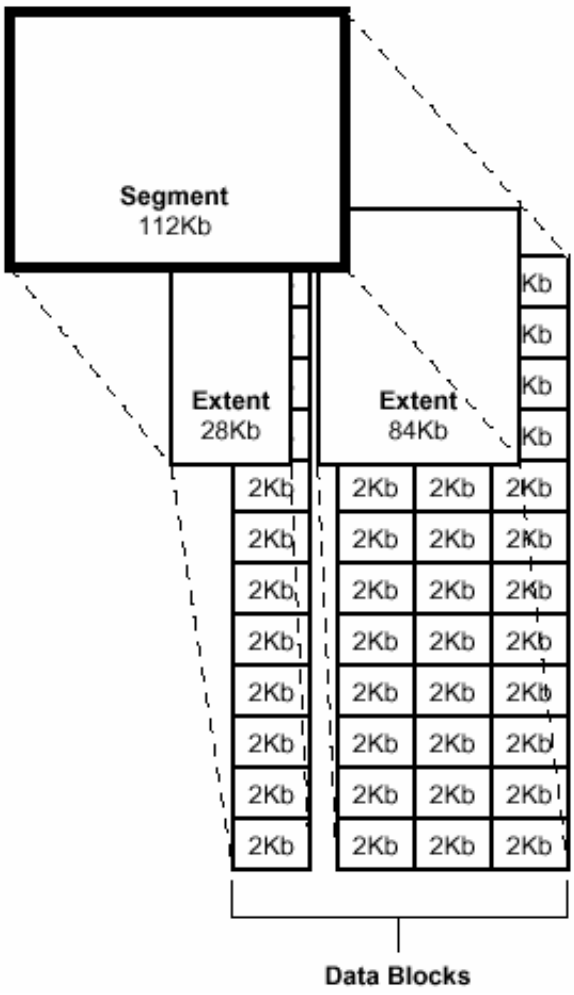




Cada BD estará formada por uno o mas tablespaces (al menos existe el **tablespace SYSTEM → catálogo del sistema**).
Cada tablespace se corresponde con uno o más **ficheros de datos**.
Objetos: tablas, vistas, índices asociados a una tabla, clusters, ...

Oracle define **esquema** como la *colección de objetos o estructuras lógicas* que corresponden directamente a los datos almacenados, y crea un **nuevo esquema** por cada usuario que crea objetos en la base de datos.

No hay ninguna **relación directa entre tablespace y esquema**, objetos del mismo esquema pueden estar en diferentes tablespaces y un mismo tablespace puede almacenar distintos esquemas.



El control del uso del espacio del disco se obtiene mediante las **estructuras lógicas de almacenamiento**: bloque de datos, extensión y segmento.

El nivel más pequeño de granularidad es el del **bloque de datos:** *número específico de bytes contiguos de espacio físico en el disco.*

(tamaño mínimo de **2K**, el bloque físico del disco y que depende el sistema operativo no tiene por que coincidir con éste).

El siguiente nivel es el de **extensión**, que es un *número específico de bloques de datos **contiguos** en el disco*.

Por último el **segmento** es un *conjunto de extensiones utilizadas para almacenar alguna estructura lógica*.

Tendremos **segmentos de datos** para tablas o clusters, **segmentos de índices** para índices, **segmentos de rollback** para poder deshacer / rehacer cambios por transacciones y **segmentos temporales**.

Hay varios tipos de sentencias en las que Oracle se ve en la obligación de utilizar los segmentos temporales: **ordenaciones**.

SELECT ... ORDER BY...
CREATE INDEX.
SELECT ... GROUP BY...
SELECT ... UNION ...
SELECT DISTINCT ...
SELECT … INSERSEC ...
SELECT ... MINUS ...

Oracle va incrementando el espacio para los segmentos mediante extensiones. Cuando una extensión está llena y necesita más espacio el sistema busca otra extensión, que podrá estar o no estar contigua a la anterior en el disco (dependerá simplemente del estado de éste).

Estructura física

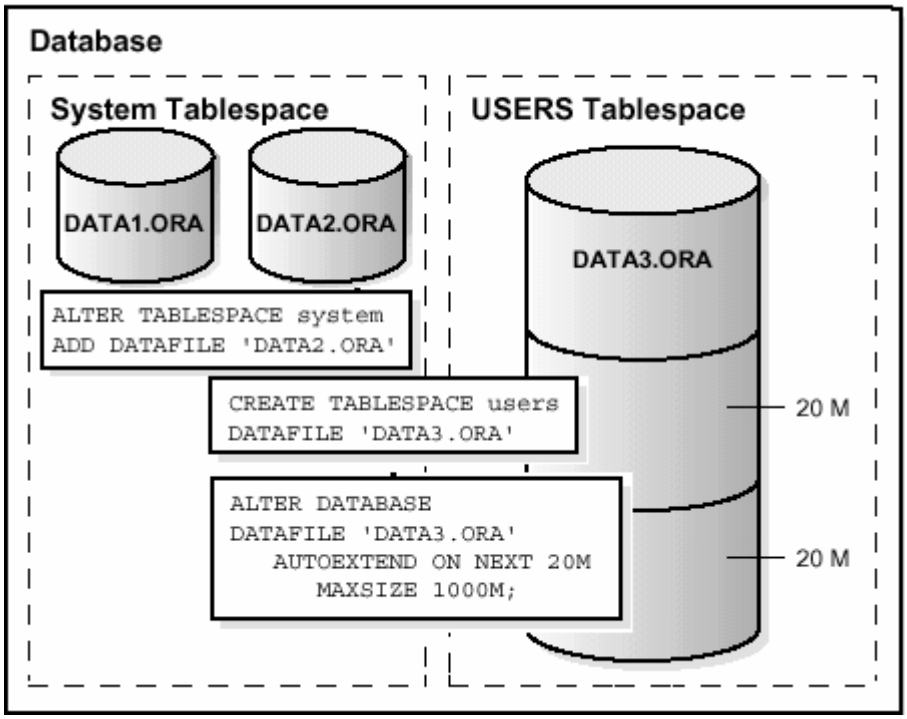
Una B.D. tiene **uno o más ficheros de datos**. Estos ficheros son de **tamaño fijo** y se establecen en el momento en que se crea la base de datos o en el momento en el que se crean tablespaces.

Los datos del fichero de datos son leídos cuando se necesitan y situados en una caché de memoria compartida (llamada ***SGA***, System Global Area: db_block_buffers; recomendable tamaño SGA = 50% de la memoria principal) para que el próximo acceso a los mismos sea más rápido.

Las modificaciones en los datos se guardan ante una petición o cuando los datos son eliminados de la SGA por falta de memoria libre para atender más peticiones.

El conjunto de **ficheros redo log** sirven para registrar todos los cambios (insert, update, delete, create, alter o drop) sobre la BD y poder recuperarla ante un error.

Los **ficheros de control** almacenan información de la estructura física de la BD.



Los distintos elementos que forman parte del **entorno de memoria de Oracle** :

SGA: System Global Area o Shared Global Area.

Zona principal de la memoria de Oracle. Está dividida en varias subareas desempeñando cada una de estas una tarea totalmente distinta: la **Shared Pool**, la **Database Buffer Cache** (parámetro DB_BLOCK_BUFFERS) y el **Redo Log Buffer**.

Shared pool:

Library Cache: se encuentra a su vez dividida en varios apartados: zona compartida de sql, zona privada de sql, procedimientos y paquetes pl/sql y, por último, la zona de control y bloqueos propios de la library cache.

***Shared sql area* o área de sql compartido**, se guardan los árboles sintácticos de las sentencias analizadas así como los planes de ejecución elegidos para cada una.

Private sql area o zona privada de sql: por cada sesión diferente que hay en la base de datos se crea una zona de sql privado. Se mantiene información de las sentencias que se están tratando en ese momento.

Procedimientos y Paquetes PL/SQL: existe un área diferenciada para el tratamiento de los procedimientos, funciones y paquetes pl/sql. Se tratan en esta zona igual que si fueran sentencias sql en la zona de sql compartido.

Dictionary cache: mantiene datos de sus propias tablas y vistas ya que accede constantemente a ellas al ejecutar cualquier sentencia.

PGA: Program Global Area.

Destinada a guardar información de los procesos de usuario y procesos de background que corren en una instancia de la base de datos y que a través de distintos procesos intercambian la información con la SGA.

Sort Areas (parámetro SORT_AREA_SIZE.)

Son las zonas de memoria que Oracle reserva para realizar ordenaciones y que resultan mucho más rápidas si se realizan en la memoria. Por supuesto, no todas las ordenaciones caben en memoria y en esos casos debe utilizar también el disco y, si hemos configurado bien el sistema, se realizarán el los tablespaces que hemos definido como **temporales**.

Elementos de Diseño Físico en ORACLE

Tablespace:
Como ya se ha comentado la BD se divide en varios tablespaces, y para cada uno de ellos se asocian uno o varios ficheros. La capacidad total del tablespace coincidirá con la suma de los tamaños de los ficheros.
Es pues una **decisión de diseño** especificar cuantos tablespace son necesarios y cual debe de ser el tamaño de los ficheros asociados.
Ejemplo:

```
CREATE TABLESPACE TS_DATOS
DATAFILE '/disco1/fichero1' SIZE 100M,
DATAFILE '/disco2/fichero2' SIZE 250M;
```

En este caso se crea un tablespace TS_DATOS asociado a dos ficheros con una capacidad total de 350M.

Tablas y extensiones de una tabla:
En el momento en el que se creen las tablas y en la propia definición en SQL se tendrá que indicar en qué tablespace se guardarán los datos. Las tablas no se pueden asignar a los ficheros.

Ejemplo: La tabla alumno se asigna a TS_DATOS pero puede que los datos se encuentren repartidos entre los dos ficheros.
CREATE TABLE Alumnos (...) **TABLESPACE TS_DATO**S;

Una parte importante del diseño será decidir el tamaño de las extensiones de cada una de las tablas que vamos a crear. Evidentemente lo mas interesante será tener las **tuplas de una relación almacenadas en espacios consecutivos en el disco**, para aprovechar las ventajas de la **recuperación de bloques consecutivos**. De esta manera será interesante tener pocas extensiones pues como se ha dicho una extensión es un conjunto de bloques consecutivos.

En la sentencia SQL de definición de las tablas se han de incluir los siguientes parámetros de almacenamiento (cláusula **STORAGE**):

- INITIAL: denota el tamaño de la extensión inicial.
- NEXT: indica el tamaño de las extensiones siguientes.
- MINEXTENTS: indica el numero mínimo de extensiones.
- MAXEXTENTS: indica el numero máximo de extensiones.
- PCTINCREASE: es un factor de crecimiento de una extensión a la siguiente.

Ejemplo:
CREATE TABLE Alumnos (...) TABLESPACE TS_DATOS
STORAGE (INITIAL 20K NEXT 30K MINEXTENTS 1 MAXEXTENTS 10
PCTINCREASE 0);

En este ejemplo la tabla alumno se almacenara en una extensión inicial de 20K, y las extensiones sucesivas serán de 30K, hasta un máximo de 10 extensiones.

Encadenamiento de tuplas:
Las tuplas se almacenan en Oracle como **registros de tamaño variable**, es decir, no se reserva espacio fijo para cada tupla. De esta manera y sobre todo cuando se producen **modificaciones sobre atributos que amplían su tamaño**, puede ocurrir que una tupla no quepa en un bloque, y haya que repartirla entre dos bloques (row chaining, filas encadenadas). En este caso el recuperar esta fila supondrá realizar muchos más accesos al disco.
Para reducir la posibilidad de encadenamientos por modificaciones de filas en el bloque es interesante dejar un porcentaje libre en cada bloque. El parámetro PCTFREE en la sentencia SQL de la definición de la tabla indica este porcentaje.

De la misma manera y para que varias inserciones se puedan hacer sobre un mismo bloque y así evitar muchos accesos al disco, se puede establecer un espacio mínimo para poder insertar filas en ese bloque, este porcentaje se establece mediante el parámetro PCTUSED.

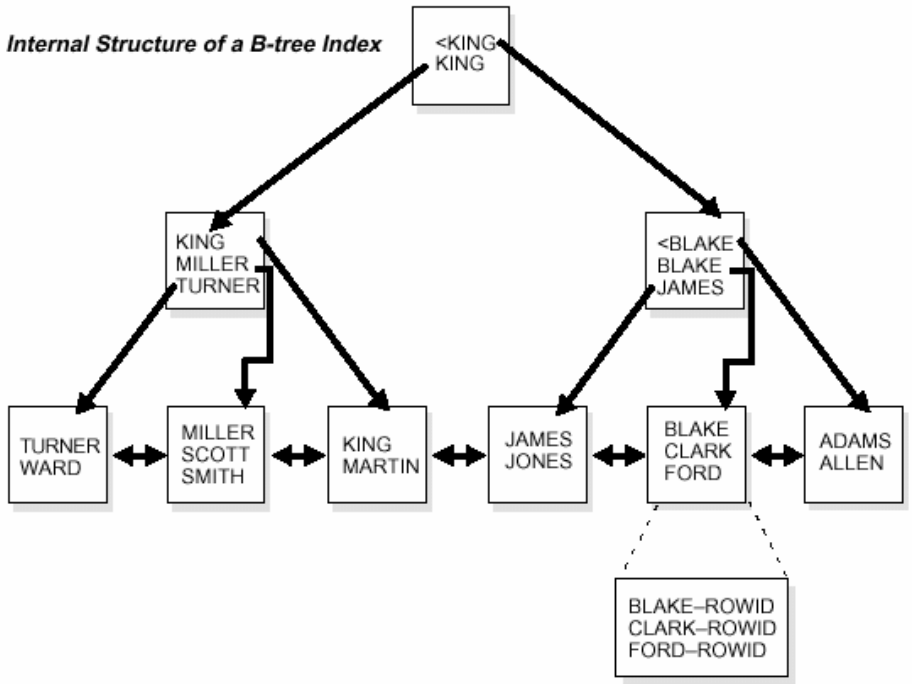
Ejemplo:
CREATE TABLE Alumnos (...) TABLESPACE TS_DATOS
PCTFREE 20 PCTUSED 40
STORAGE (INITIAL 20K NEXT 30K MINEXTENTS 1 MAXEXTENTS 10 PCTINCREASE 0);

En este ejemplo:
- Se reservará el 20% de cada bloque para evitar encadenamientos.
- Un bloque que se ha llenado previamente, no se volverá a utilizar para inserciones hasta que ese porcentaje de utilización sea inferior al 40%.

Recomendaciones:
PCTFREE: Si la fila no va a sufrir modificaciones → valor entre 5 y 10; pero si las filas pueden sufrir muchas variaciones en cuanto a tamaño → 20-40.
PCTUSED: Un valor entre 80 y 40; si no se van a producir muchas modificaciones 80 y en caso contrario 40.
La suma entre este valor y el anterior no debe de superar a 100.

Creación de índices:

Los índices son estructuras sobre un conjunto de atributos de una tabla y que permiten establecer una **ordenación** lógica de las tuplas de una relación por el valor de los atributos que forman la clave del índice.



Oracle permite la creación de índices B* (similares a los B+). Como los índices necesitan de espacio en el disco para almacenar sus valores, en la definición de éstos se utilizarán los parámetros que acabamos de indicar en la creación de tablas.

Ventajas de la estructura de B-tree

- Todos los nodos hoja están a la misma profundidad. La recuperación de cualquier registro lleva aproximadamente el mismo tiempo.
- Permanecen automáticamente balanceados.
- Todos los bloques del árbol están llenos de media en $\frac{3}{4}$.
- Excelente desempeño para una amplia variedad de consultas, desde emparejamiento exacto (exact match) a búsquedas por rango de valores.
- Inserts, updates, y deletes son eficientes, manteniéndose el órden de clave.

Ejemplo:

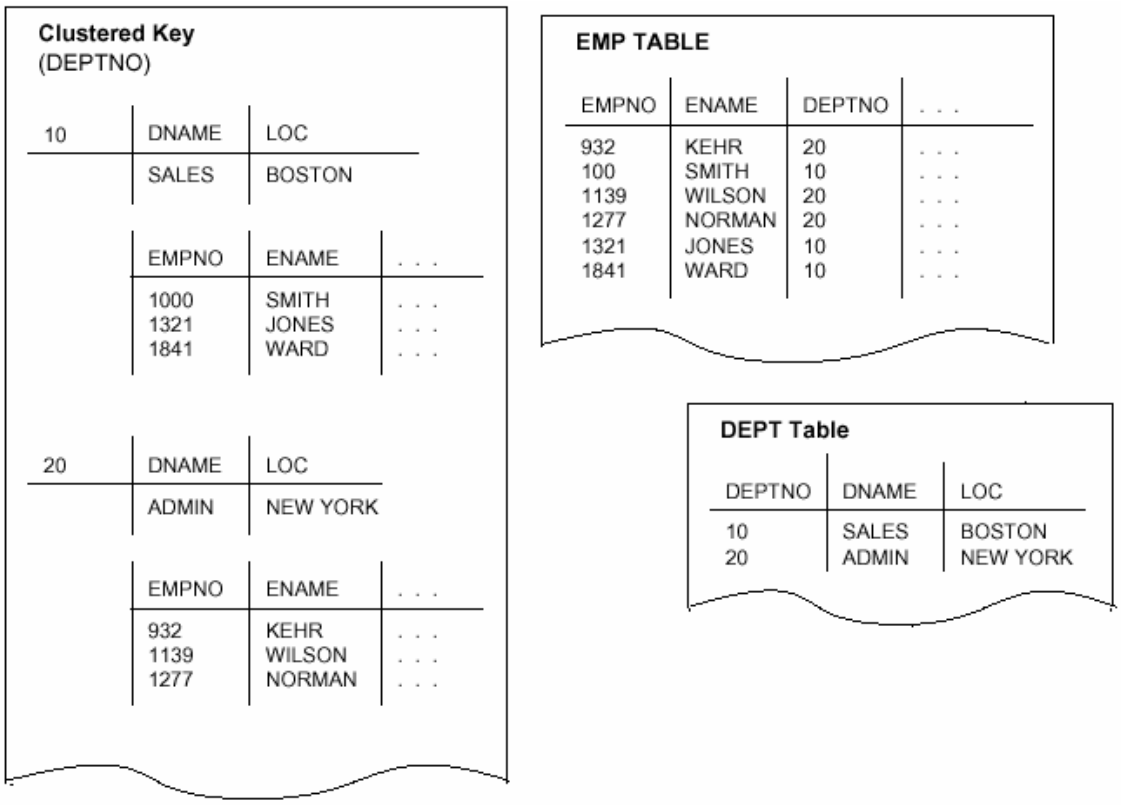
CREATE TABLESPACE TS_INDICES DATAFILE '/disco1/indices' SIZE 100M;

CREATE INDEX nom_ind ON Alumnos (...atributos....)
TABLESPACE TS_INDICES PCTFREE 10
STORAGE (INITIAL 10K NEXT 10K MINEXTENTS 1 MAXEXTENTS 10 PCTINCREASE 0);

Conviene por razones de eficiencia separar las tablas de los índices en distintos tablespace que estén ubicados en discos diferentes para poder realizar en paralelo peticiones de E/S sobre los discos.

Clusters:

Cuando **dos tablas** tienen atributos mediante los cuales es usual realizar concatenaciones, puede resultar más eficiente guardar las tuplas que se concatenan de ambas relaciones en el mismo bloque, el cluster es un método que nos permite esto.



Clustered Key
(DEPTNO)

10	DNAME	LOC
	SALES	BOSTON

EMPNO	ENAME	. . .
1000	SMITH	. . .
1321	JONES	. . .
1841	WARD	. . .

20	DNAME	LOC
	ADMIN	NEW YORK

EMPNO	ENAME	. . .
932	KEHR	. . .
1139	WILSON	. . .
1277	NORMAN	. . .

EMP TABLE

EMPNO	ENAME	DEPTNO	. . .
932	KEHR	20	. . .
100	SMITH	10	. . .
1139	WILSON	20	. . .
1277	NORMAN	20	. . .
1321	JONES	10	. . .
1841	WARD	10	. . .

DEPT Table

DEPTNO	DNAME	LOC
10	SALES	BOSTON
20	ADMIN	NEW YORK

El utilizar clusters solamente afecta a la organización física de los datos y no a la estructura lógica. Y en este caso la ventaja es doble pues se aceleran las concatenaciones (joins) y se disminuye espacio de utilización en el disco pues los atributos que se concatenan solamente se guardan una vez.

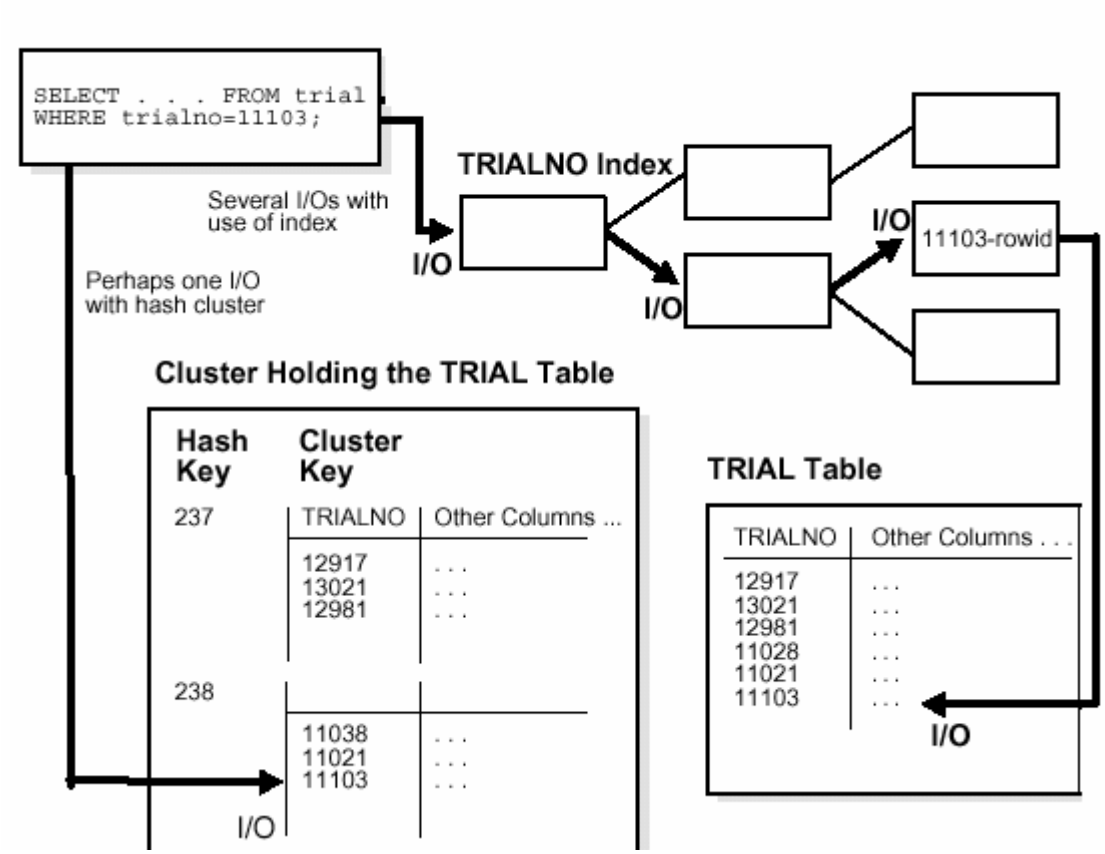
Ejemplo:

```
CREATE CLUSTER CL_1 ( clave number(4) ) SIZE 512
TABLESPACE TS_DATOS
PCTFREE 20
STORAGE ( INITIAL 20K NEXT 30K MINEXTENTS 1 MAXEXTENTS 10
PCTINCREASE 0);

CREATE TABLE Alumno( … ) CLUSTER CL_1( a1 );
```

Como vemos los parámetros de almacenamiento se asocian al cluster y no a las tablas. El parámetro **SIZE** nos indica el tamaño máximo que ocuparán las tuplas que se concatenan y este debe de ser una estimación.

Dispersión:
Existe una variación en el uso de los clusters y es el uso de dispersión. En este caso las filas no se almacenaran por el valor de la clave del cluster sino por el resultado de aplicar una función de dispersión a el valor de la clave del cluster. La dispersión que implementa Oracle es **estática** y es necesario conocer cuándo se crea el rango de valores de la función de dispersión, en este momento se reserva espacio para todos los bloques del cluster y los conflictos se resuelven mediante cubos de desborde.



Ejemplo:
CREATE CLUSTER CL_1 (clave number(4))
SIZE 512
HASH IS clave HASKEY 300
TABLESPACE TS_DATOS
PCTFREE 20
STORAGE (INITIAL 20K NEXT 30K MINEXTENTS 1 MAXEXTENTS 10
PCTINCREASE 0);
En este ejemplo la función de dispersión devuelve valores entre 0 y 299.