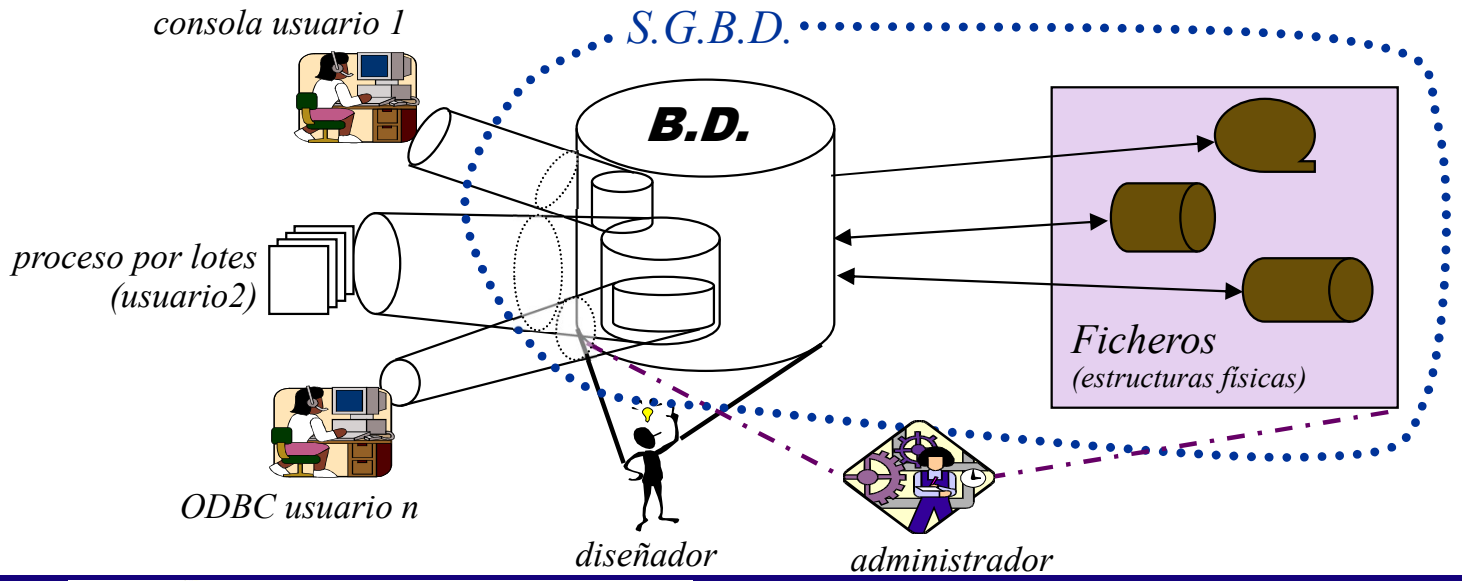
Tema 8: Sistemas Gestores de BBDD

- **Introducción.**
- **Arquitectura y Esquema Interno de un SGBDR: ORACLE®**
- **Introducción al Afinamiento de BBDD ORACLE®**
  - Conceptos Básicos
  - Estructuras: índices, clusters, parámetros
  - Consultas
  - Procesos: planes de ejecución y hints

Tema 8: Introducción a SGBD (DBMS)

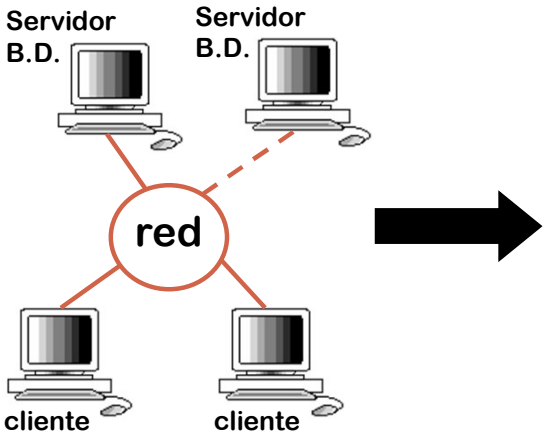
*Conjunto coordinado de **herramientas** que proporciona los medios necesarios para **interaccionar** con la base **a todos los niveles***

- herramientas: programas, procedimientos, lenguajes, ...
- interaccionar con la base: describir y manipular datos almacenados en la base, preservando su integridad, confidencialidad, y seguridad.
- a todos los niveles: usuario, programador, analista, ...

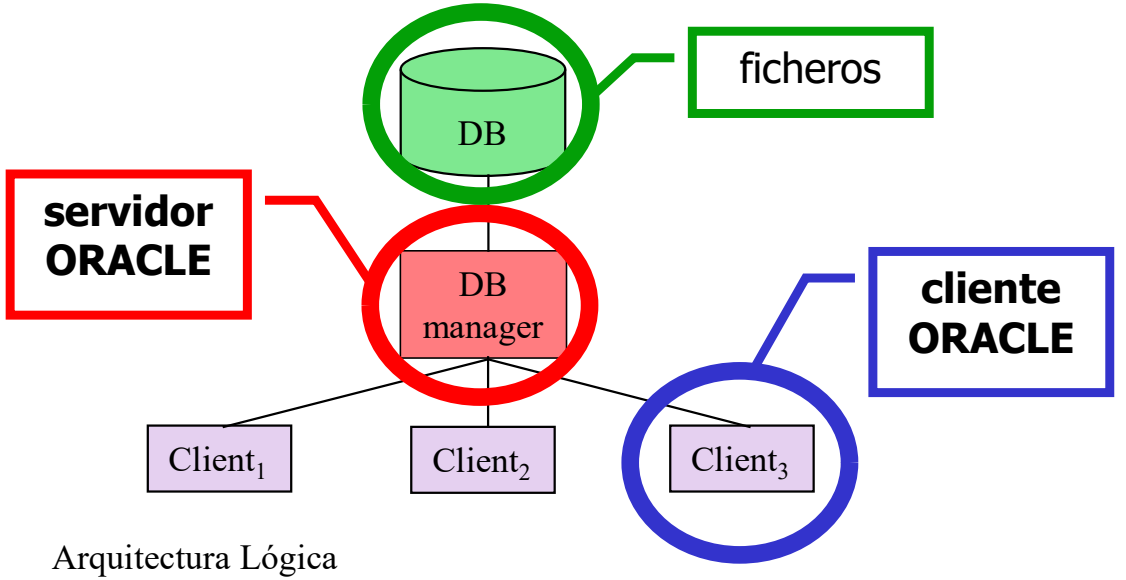


Tema 8.1: Ejemplo de SGBD El SGBDR ORACLE®

- **ORACLE:** Sistema Gestor de Base de Datos Relacional;
  Versátil + probada Eficiencia y Escalabilidad + amplia Difusión
- Basado en el lenguaje de datos PL/SQL (extensión de SQL)
- Entorno multiusuario (Cliente/Servidor).
- El servidor alberga: repositorio Sw, ficheros, y servicios (instancias)



Arquitectura Comunicaciones



Arquitectura Lógica

Tema 8.1.1: Instancias ORACLE®

El **servidor Oracle** puede albergar varias instancias independientes (al menos una) que comparten repositorio Sw y se reparten los recursos)

Instancia:

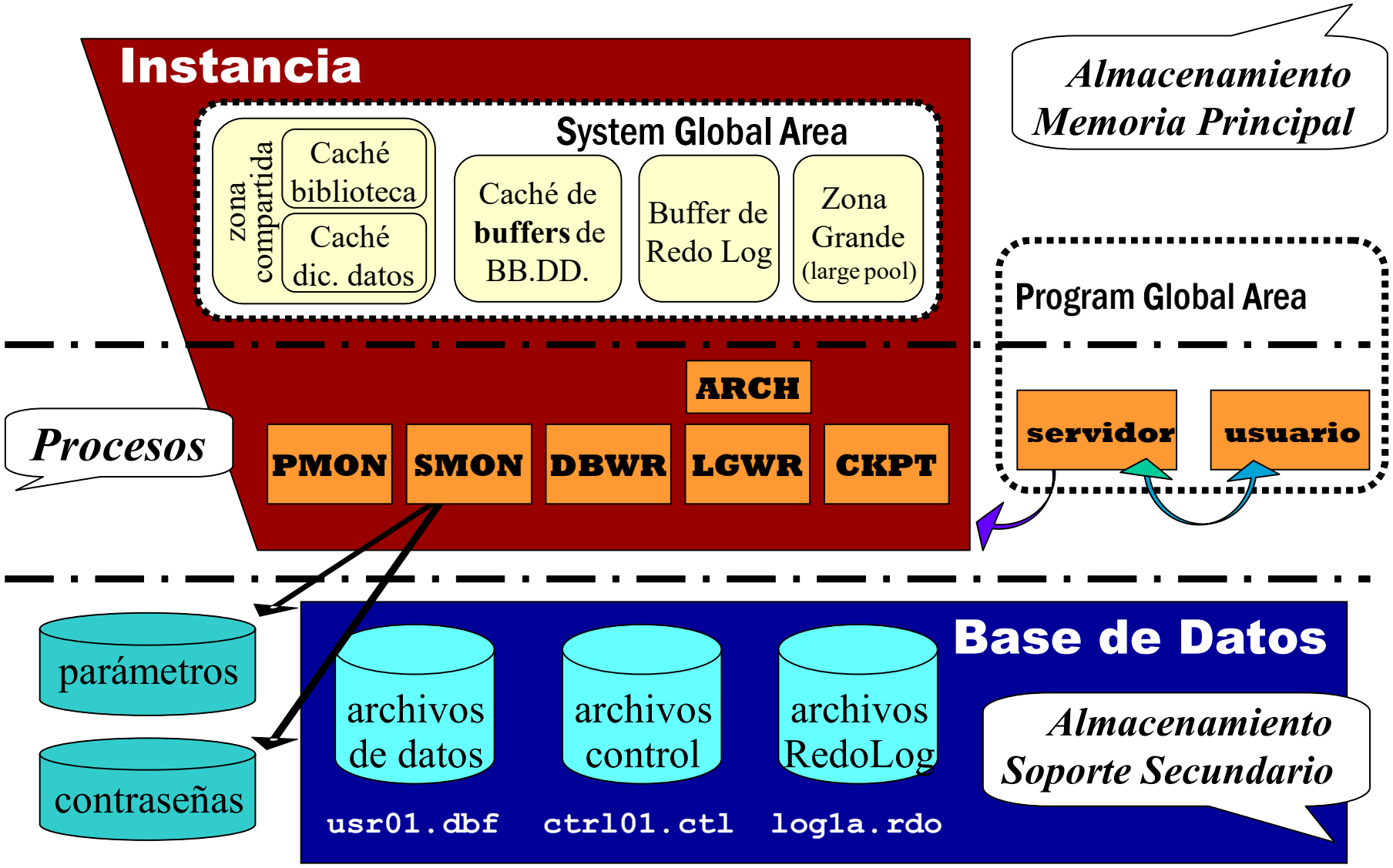
- Es un **conjunto completo de servicios** de BBDD
  (si existen o no varias instancias en el mismo servidor, es transparente)
- Servicios de acceso, control y uso de las BD.
- Se compone de **procesos** y **estructuras de datos** (físicas y en memoria)
- Sus recursos son compartidos por todos los usuarios (de la instancia).
- Las estructuras en memoria se organizan en dos áreas: **SGA** y **PGA**
- Las estructuras físicas se apoyan en el concepto de ***tablespace***.
  Cada tablespace podrá almacenarse en uno o más ficheros de datos.

Tema 8.1.1: Instancias ORACLE®

Instancias y Bases de Datos:

- Una **BD** es un conjunto de datos **almacenado** y **accesible** según una **estructura lógica** (*esq. relacional*, ~ cjto. de tablas interrelacionadas)
- Sus elementos pueden pertenecer a uno o más usuarios, almacenarse en uno o más *tablespaces*, pero siempre en **una sola instancia**.
- Dentro de una instancia, los objetos de una BD se referencian mediante la notación de punto (por ejemplo: *dueño.tabla.atributo*), si bien el usuario *dueño* puede omitir la primera sección.
- Un usuario (o cjto. de usuarios) aislado con sus objetos puede ser considerado como BD, aunque frecuentemente se asocia al concepto de BD todo lo contenido en la instancia.
- BBDD de distintas instancias pueden *federarse*, posibilitando su interrelación (si se requiere mayor integración → misma instancia).

Tema 8.1.1: Arquitectura ORACLE®



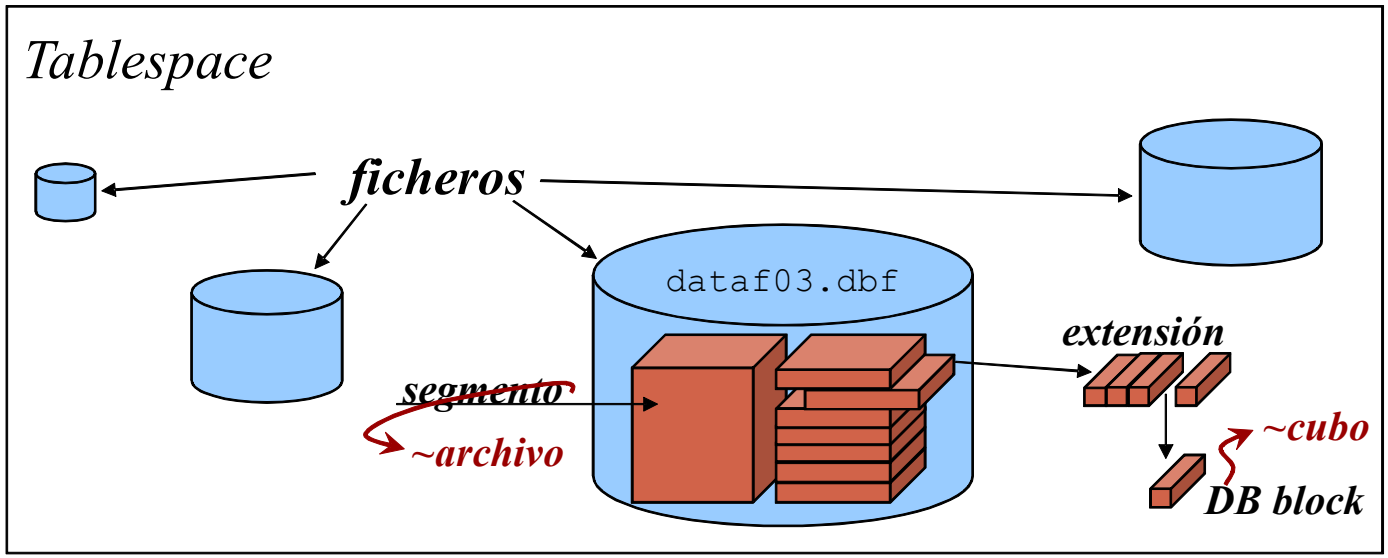
Tema 8.1.1: Ficheros de ORACLE®

- ***Parámetros*** (pfile / spfile): información para la inicializar la instancia.
- ***Contraseñas***: información de acceso a la instancia.
- ***Control***: contienen la información necesaria para la utilización de la instancia (nombre BD, nombre y ubicación de ficheros, back-up, etc.)
- ***Redo log***: protege la BD con *journaling* (registro de cambios que sufre la base, anotando cada operación antes y después de realizarse).
- **Datos** (*datafiles*): almacenan los segmentos de la BD

Tema 8.1.2: Esquema Interno ORACLE®

Tablespaces y Datafiles

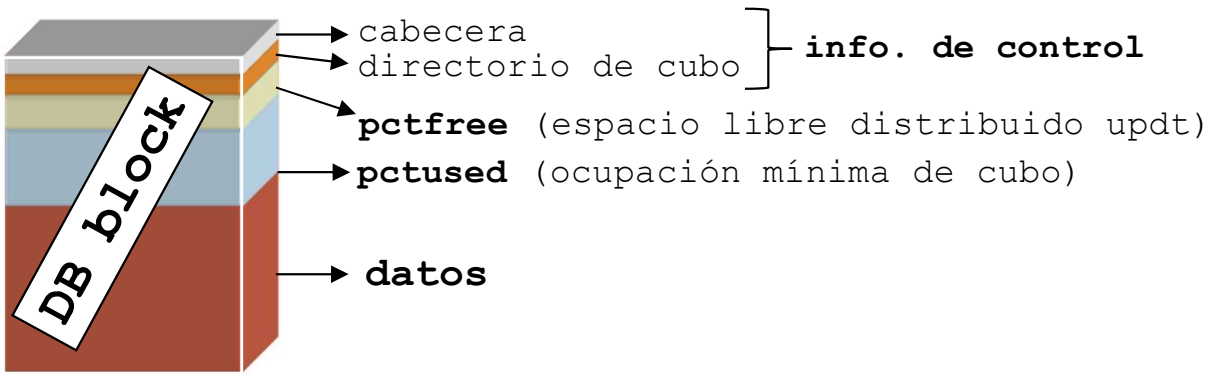
- El ***tablespace*** es un almacén de datos.
- Puede tener asociados varios **ficheros** de datos (*datafiles*), y estos se asignan a un solo tablespace. El tamaño máximo de datafile es 32 GB.
- El tablespace se organiza en **segmentos**, uno para cada elemento (tabla, índice, ...) que contiene. Cada segmento se compone de **extensiones.**



Tema 8.1.2: Esquema Interno ORACLE®

Cubos y Extensiones

- La **extensión** es un conjunto de *DB-blocks* contiguos asignados a un elemento
  - Cuando un elemento se crea, a su segmento se le asigna una extensión inicial
  - Cuando a un segmento se le acaba el espacio asignado, crece en una extensión
- El ***DB-block*** responde al concepto de **cubo** (1 'DB-block' ≡ 1 ó más bq físicos). Sus características y espacio (*blocksize*) son únicas para todo el tablespace, si bien pueden redefinirse para algunos objetos.



  - PCTFREE: porcentaje reservado para modificaciones (por defecto: 10%)
  - PCTUSED: porcentaje mínimo ocupado (por defecto: 60%); si una actualización deja al cubo demasiado vacío, éste será candidato para inserciones.

Tema 8.1.2: Esquema Interno ORACLE®

Cubos y Extensiones

- El *DB-block* admite cinco conf. de espacio: 2 KB, 4 KB, **8 KB**, 16 KB, y 32 KB.
- Un espacio de cubo grande
  - aprovecha la secuencialidad del dispositivo (disco),
  - aumenta la densidad (menos información de control y *gaps* más pequeños)
  - y reduce el tiempo de acceso a la totalidad (full-scan).
- Pero puede implicar
  - desperdicio de espacio (especialmente en clusters),
  - aumento de accesos a disco en procesos indexados,
  - menor eficiencia del *buffering*.
- Las extensiones grandes también aprovechan la secuencialidad de disco.
- El *DB-block* se corresponde con una página en el **buffer** (Mem $_{\text{Intermedia}}$).
  → debe tenerse un buffer distinto adaptado a cada espacio de cubo en uso.
- Determinados elementos (*cluster*) permiten el almacenamiento en **celdas** (*subconjunto del cubo*; su tamaño es divisor entero del cubo donde se aloja).
  Utilizar celdas puede mejorar la densidad, aprovechamiento de espacio, y coste de *fullscan*

Tema 8.1: Qué es la administración de BBDD

- La gestión de datos es el corazón de (casi toda) maquinaria empresarial. Garantizar su funcionamiento, seguridad y eficiencia es crucial.
- El administrador (DBA) es un profesional clave, con este perfil:
  - Conoce las tecnologías que soportan la BD (Hw, Sw, comm.)
  - Domina las tecnologías de BD (y los SGBD)
  - Conoce la estructura de sus BD (idealmente, involucrado en el desarrollo)
  - Otras competencias: sociales, organizativas, gestión personal, …
- Las funciones del administrador (DBA) incluyen:
  - Gestión: posibilitar el uso de la BD y administrar sus recursos
  - Protección: asegurar la persistencia y confidencialidad de la información
  - **Afinamiento: maximizar la eficiencia de la BD**
- Otros perfiles: científico de datos, ingeniero sistemas información, …

Tema 8.2: Conceptos básicos: espacio

- **tablespace**: espacio en la base de datos para almacenar objetos. Tipos:
  - permanente: almacena los objetos persistentes
  - temporal: almacena objetos cuyo alcance no supera a la sesión
  - *undo*: almacena datos de recuperación (alternativa a segmentos de *rollback*)

```
CREATE [bigfile|smallfile] [TEMPORARY|UNDO] TABLESPACE <name>
       [{DATAFILE|TEMPFILE} <file_spec> [, <file_spec>…] ]
       [BLOCKSIZE <int> [k] ]
       [MINIMUM EXTENT <size> ] … ;
```

http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_7003.htm

- **datafile**: fichero de datos, asignado a un tablespace

```
ALTER  TABLESPACE <name>
       ADD DATAFILE <fichero> [SIZE <int> {k|m|g}];
```

http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_7003.htm

Tema 8.2: Conceptos básicos: el *catálogo*

- el **catálogo relacional** en Oracle se denomina diccionario de datos:

```
SQL> SELECT * FROM DICTIONARY;

SQL> SELECT * FROM DICT_COLUMN; /* conviene seleccionar table_name… */
```

- Para simplificar el acceso, existen numerosas vistas (user/all/dba) que muestran objetos propiedad del usuario, accesibles por él, y de toda la BD
- Algunas de las vistas más utilizadas (no existen todas las combinaciones):

```
*_tables, *_tab_columns, *_views, *_constraints, *_source,
*_indexes, *_ind_columns, *_objects, *_catalog, *_synonyms,
*_tablespaces, *_users, *_role_privs, *_free_space, …
```

- Vistas de uso de espacio: `sm$ts_free, sm$ts_used, sm$ts_avail`
- Otras vistas interesantes (V$*):

```
v$session, v$process, v$rollstat, v$db_object_cache,
v$datafile, v$tablespace, v$database,…
```

Tema 8.2: Conceptos básicos: Monitorización

- Existen también diversas vistas que proporcionan **estadísticas de uso**. Algunas de las más utilizadas son las siguientes:
  - `v$sesstat`: estadísticas de las sesiones activas
  - `v$statname`: nombre de las estadísticas de la vista anterior
  - `v$sess_id`: operaciones i/o lógicas y físicas por cada sesión
  - `v$filestat`: lecturas/escrituras en cada datafile
  - `v$librarycache`: rendimiento de la caché de la sga
  - `v$sgastat`: estadísticas de sga global
  - `v$sqlarea`: estadísticas de la caché de cursor (workspaces)
- ***Autotrace*** proporciona el plan y algunas estadísticas: `set autotrace on`
- Por otro lado, existen estadísticas del optimizador (para aplicar opt. por coste), configurables con el paquete *dbms_stat*, y activadas (enable/disable) con el paq. *dbms_auto_task_admin* (parám. client_name ='auto optimizer stats collection')

Tema 8.2.1: Afinamiento (tuning)

Estructuras: Indexación

- La selección de índices (**ISP**) forma parte del *diseño físico*.
- Consiste en decidir las **estructuras auxiliares** para optimizar el rendimiento de la BD de acuerdo a los procesos que la actualizan o consultan
- La sintaxis básica de creación de indices en ORACLE® es:

```
CREATE [ind_type] INDEX ind_name
       ON table_name(ind_key) [FROM … ];
```

donde


```
ind_type := UNIQUE | BITMAP |    (default)
```

`ind_key :=` columna(s) de *table_name*, separadas por comas
(o una función sobre esas columnas...)

Tema 8.2.2: Afinamiento - Estructuras

Clusterización en Oracle®

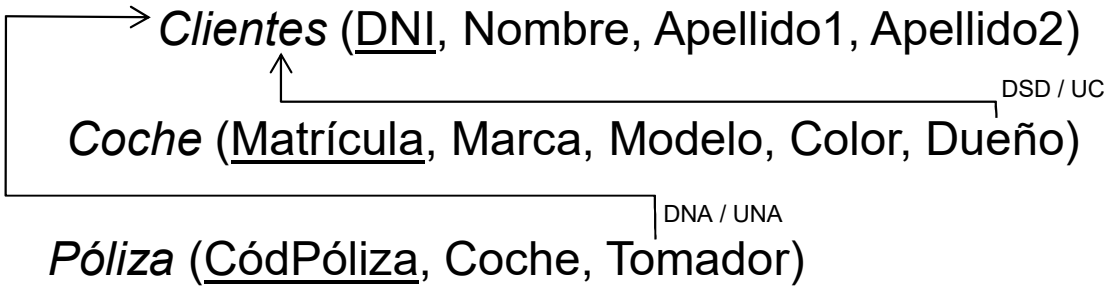
- Para ORACLE, un *cluster* es la definición de clave privilegiada.
- A través del cluster, varias tablas pueden almacenar físicamente los datos combinados mediante esa clave (eficiente para JOIN y accesos por la clave privilegiada, ineficiente para todo lo demás).
- El cluster debe **crearse antes de crear la tabla**
- El cluster garantiza que **toda la fila** combinada (el resultado del join de todas las tablas implicadas para un valor del cluster) se almacena físicamente **en el mismo cubo**

- **Ventaja:** el acceso a elementos combinados es más eficiente
- **Inconveniente:** el acceso individual puede ser muy ineficiente

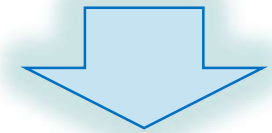
Tema 8.2.2: Afinamiento - Estructuras

Clusterización en Oracle®

Ejemplo:




```
CREATE  CLUSTER identidad (DNI VARCHAR2(9));
CREATE  TABLE cliente(…) CLUSTER identidad (DNI);
CREATE  TABLE coche(…) CLUSTER identidad (dueño);
CREATE  TABLE poliza(…) CLUSTER identidad (tomador);
CREATE INDEX ind_dni ON CLUSTER identidad;
```



```
identidad
( DNI C(9),
  cliente (nombre C(25), apellido1 C(15), apellido2 C(15)),
  coche (matrícula C(7), marca C(20), modelo C(20),color C(10) )*,
  poliza (cod C(30), coche C(7) )*
);
```

Tema 8.2.2: Afinamiento - Estructuras

Clusterización en Oracle®

- La **elección de la clave** es **crítica**: puede bajar la densidad.
- Como otros objetos, permite definir características físicas
- El *cluster* puede ser **indizado** o **disperso** (con **orden** opcional).
- Un *cluster* ***mono-tabla***: permite cambiar la organización base
  - **Ventaja**: eficiencia en acceso por clave clusterización
  - **Inconveniente**: menor densidad, peor eficiencia en accesos por claves de selección no privilegiadas.

Tema 8.2.3: Afinamiento (tuning)
Parámetros Físicos

- Oracle permite definir parámetros físicos en la creación de objetos (tablas, clusters, índices, vistas materializadas).
- **Tablespace**: al definir este parámetro, se pueden elegir otros como el *blocksize* (espacio de cubo) que puede ser distinto en cada tablespace (cuidado con definir cubos no adecuados). En la instancia, se pueden definir cachés de cada formato.
- En los *clusters*, SIZE permite definir *celdas* ('cubos' más pequeños que el cubo)
- Espacio libre distribuido: `PCTFREE` y `PCTUSED` a la medida del objeto
- **STORAGE**: parámetros de almacenamiento
  - Tamaño de las extensiones: `initial, next, pctincrease, maxsize, maxextents, minextents`
  - Memoria intermedia: cuál de los *pools* será utilizado para ese objeto: `buffer_pool {keep|recycle|default}`
  - … y algunos más (como las listas de puntos de inserción, o `freelists`)

Tema 8.2.4: Afinamiento (tuning)

Optimización de Consultas en Oracle®

- La mayoría de instrucciones de manipulación de datos pueden resolverse, en general, siguiendo distintos caminos físicos (distintos algoritmos).
- Es importante describir bien las consultas (algebraicamente) para conseguir ejecuciones más directas y eficientes.
- Si bien es cierto que muchos SGBD tienen un Optimizador capaz de *reinterpretar expresiones* (para operar $\sigma_{=} \times$ como si fuera $*$, por ejemplo) es conveniente describirlas bien para no depender de ese componente.
- Además, es necesario conocer los detalles de aplicación de cada operador en el SGBD, para controlar sus mecanismos. Por ejemplo, en Oracle:
  - el operador LIKE fuerza un recorrido serial: `... where name LIKE 'John';`
  - una función/operación evita el uso de índices: `... where enddate+0 <...`
  - a menos que el índice se base en la función: `...where UPPER(name)='...'`

```
CREATE INDEX my_index ON Clients(UPPER(name));
```

Tema 8.2.5: Afinamiento (tuning)

Plan de Ejecución

- El camino físico para resolver una instrucción de manipulación de datos, también denominado *plan de ejecución*, es el resultado de una o varias secuencias de decisiones, que se pueden representar de forma arbórea (árbol de decisión).
- El SGBD suele contemplar instrucciones (en el LCD) que permiten al usuario conocer el plan de ejecución.
- Para elegir el camino físico que resolverá la selección descrita, se pueden seguir distintas estrategias. El SGBD contará con una o más de estas estrategias (y mecanismos para determinar cuál seguir, si son varias).
- Hay que establecer un objetivo: priorizar reducir el coste de los primeros resultados o el coste de la operación completa. Depende si el consumidor aprovecha los resultados según se obtienen o necesita la completitud.

Tema 8.2.5: Afinamiento (tuning)

Obteniendo el *Plan de Ejecución*

- La instrucción SQL para obtener un plan de ejecución es **EXPLAIN**

```
EXPLAIN PLAN [SET statement_id = '...'] FOR <dml_sentence>;
```

  - EXPLAIN requiere privilegios de consulta en … **V_$SESSION, V_$SQL, V_$SQL_PLAN, V_$SQL_PLAN_STATISTICS_ALL**

- En Oracle, el plan se almacena en una tabla temporal global (de sesión) denominada **SYS.PLAN_TABLE$** (con sinónimo **PLAN_TABLE**)
  - La tabla de planes se crea automáticamente (versión 10). En versiones anteriores (o para crear una tabla local permanente) utiliza utlxplan.sql

- EXPLAIN proporciona una previsión de plan 'a priori' (sin ejecutarlo)
- En Oracle, el modo autotrace (**set autotrace on**) proporciona una descripción básica del plan efectuado, y un resumen de estadísticas.

Tema 8.2.5: Afinamiento (tuning)

Consultando el *Plan de Ejecución*

- Se puede consultar un plan directamente de **PLAN_TABLE** …

```
SELECT * FROM PLAN_TABLE WHERE statement_id='…';
```

- …pero es una tortura interpretar sus 36 ilegibles columnas, especialmente porque cada plan consta de varias filas.
- Para simplificarlo, Oracle facilita el paq. **DBMS_XPLAN** con la función **DISPLAY(plan_table, statement_id, format, filter)**

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

SELECT PLAN_TABLE_OUTPUT FROM TABLE
   (DBMS_XPLAN.DISPLAY(NULL,'statement_id','BASIC'));

SELECT * FROM TABLE (DBMS_XPLAN.DISPLAY(FORMAT=>'+ALLSTATS'));
```

- Formatos: **BASIC | TYPICAL | ALL** (se pueden añadir más opciones)
  https://docs.oracle.com/database/121/ARPLS/d_xplan.htm#ARPLS74741

Tema 8.2.6: Afinamiento (tuning)

Optimización basada en... *Reglas* vs *Costes*

- El plan de ejecución se puede alcanzar por distintos métodos, cuya disponibilidad dependerá del SGBD. Los más habituales son :
  - REGLAS: cada decisión se toma en base a criterios apriorísticos, que presentan buen comportamiento (en general).
  - **COSTES**: las decisiones se toman calculando la mejor opción; el camino será óptimo si los datos (*estadísticas de estado de la BD*) son fiables. Esa fiabilidad dependerá de la significatividad del muestreo elegido, de la frecuencia de actualización, etc.
  - ESTADÍSTICAS DE USO: algunos SGBD almacenan estadísticas de uso (aplicación de planes pasados), permitiendo adoptar aquellas decisiones que anteriormente fueron exitosas.
  - DIRIGIDOS por el usuario: el usuario interviene en el proceso de decisión. Para esta vía, Oracle cuenta con el mecanismo de *Hints*.

Tema 8.2.6: Afinamiento (tuning) HINTS en Oracle®

- Ningún optimizador es infalible: las reglas son generales; los costes dependen de la actualización de parámetros; las estadísticas son rígidas; y todos esos métodos ignoran la semántica de los datos.
- **HINTS**: podas del árbol de decisión, marcados por el usuario.
- No estándar, en PL/SQL se especifican como comentarios:

```
SELECT /*+ HINT */ attributes FROM tablename … ;

SELECT --+ HINT
   attributes FROM tablename … ;
```

- Se pueden especificar varios HINTS para la misma instrucción (separados por espacios).
- Algunos HINTS tienen versión opuesta (ejemplo: INDEX→ NOINDEX)

Tema 8.2.6: HINTs Oracle® más usuales

Sintaxis del HINT			Descripción
`/*+RULE */`	`/*+ALL_ROWS*/`	`/*+FIRST_ROWS(n)*/`	elige tipo de optimizador (reglas/costes)
`/*+ FULL(tablename) */`			recorrido a totalidad (*full scan*) de tabla
`/*+ ROWID(tablename) */`			rowid scan (cubos individuales)
`/*+ CLUSTER(tablename) */`			tabla en cluster accedida por el mismo
`/*+ HASH(tablename) */`			usa la dispersión de una tabla en *cluster*
`/*+ ORDERED */`	`/*+ LEADING(tab1 tab2)*/`		join tables in Q order, or specify an order
`/*+USE_NL(t)*/`	`/*+USE_MERGE(t)*/`	`/*+USE_HASH(t)*/`	join tables with nested loops/merge/hash (*inner*)
`/*+ INDEX(tablename) */`	`/*+NO_INDEX(table)*/`		use/forbids any index on the given table
`/*+ INDEX(tablename index1 index2 ... ) */`			use/forbids specific index/es (one or +)
`/*+ AND_EQUAL(tablename index1 index2 […]) */`			use more than one index (up to 5)
`/*+ INDEX_ASC(…) */`	`/*+ INDEX_DESC(…) */`		index range scan in asc/desc order
`/*+ INDEX_FFS(…) */`	`/*+ INDEX_SS(…) */`		index fast full scan // index skip scan
`/*+ INDEX_JOIN(tablename [indexname […]]) */`			join indexes (sort of inverted access)
`/* CACHE */`	`/* NOCACHE */`		situar cubos al comienzo/final de la lista LRU
`insert /*+ APPEND */ …`	`insert /*+NOAPPEND*/`		escritura directa HWM (~~buffer, stack, RI, trigger~~)