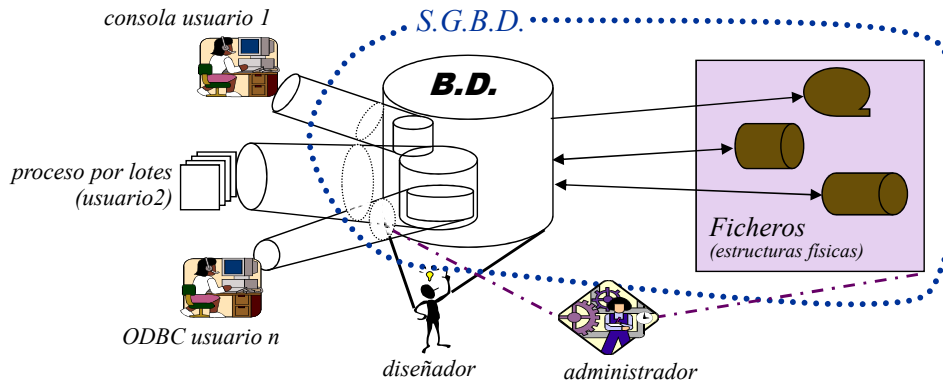


- **Introducción**
- **Ejemplo de SGBDR: ORACLE®**
 - Arquitectura del SGBDR Oracle
 - Esquema Interno del SGBDR Oracle
- **Administración, control y afinamiento de la BD**
 - Introducción
 - Configuración local y de instancias
 - Monitorización y estadísticas
 - Afinamiento: índices, clusters, parámetros y *hints*
- **Conectividad de un SGBD (JDBC)**
- **Concurrencia en un SGBD (Oracle®)**

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: Introducción - Historia



Gestores Navegacionales

- apuntamientos físicos o relativos
- Modelos de Datos: Jerárquico y en Red
- desarrollados a finales de 60', y explotados en los 70'
- tecnología **eficiente** (OLTP, BD convencionales, ...)

Gestores Relacionales

- apuntamientos lógicos
- Modelo Relacional (y otros conceptuales...)
- gestados en los 70', y explotados desde los 80'
- tecnología **eficaz** y accesible

Desktop Databases: adaptación de la tecnología a pequeña escala

Object Oriented databases: adaptación a un paradigma de análisis

- productos comerciales a partir de los 90'

NoSQL – NoREL – no structure, no model, no limit... BIG DATA

- otros enfoques (key-value, column, document,...)
- ... orientados a necesidades específicas (analíticas)

Next?

NewSQL, CloudDB ...

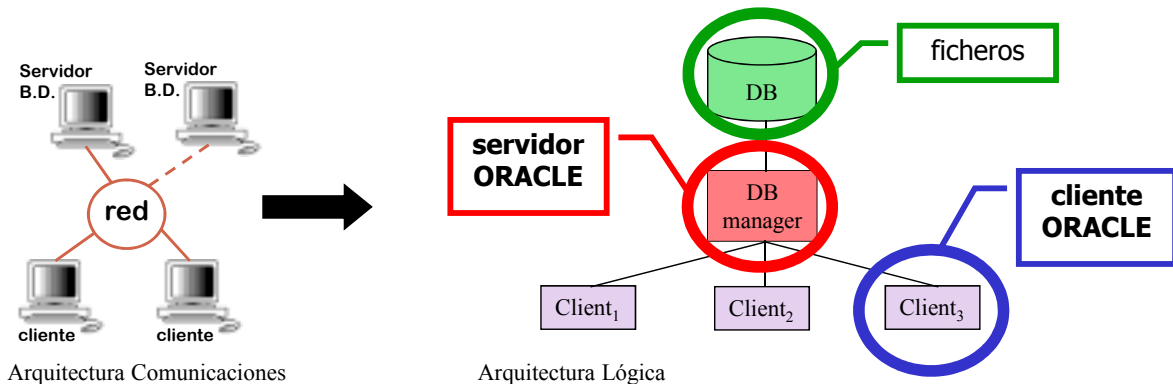
Tema 8: Introducción - Eventos

1959	Nace el consorcio CODASYL
1964	Primer gestor: Integrated Data Storage (IDS; pre-network)
1966-68	IBM desarrolla el <i>Information Management System</i> (M. Jerárquico)
1968-69	Especificaciones y Lenguajes de Datos para los Gestores en Red Cincom lanza el SGBD TOTAL (shallow-network)
1970	Lanzamiento de ADABAS (online trans. proc. DB; inverted lists)
1971	...y otros similares como Datacom/DB (<i>key-driven</i> , for huge volumes)
1973...	Otros productos más cercanos al M. en red: IDMS
...	
1974	IBM desarrolla System R (gestor para el M. Relacional de EF Codd)
1978-79	Lanzamiento de Oracle V1-V2 (primer gestor relacional comercial)
1979-80	Lanzamiento de dBase (gestor para ordenadores domésticos)
1981	Le siguen SQL/DS (IBM) , Informix, Sybase, y otros gestores relacionales
1983	IBM lanza DB/2
...	
1990	Comercialización de OO-DBMS: Gemstone, Objectivity/DB, ...
1992	SAP R3 (suite basada en la arq. <i>three-tier</i>)
...	
2004-06	BigTable... big boost for Big Data
2008	BigData spring...

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).



<div>uc3m</div> <div>Tema 8.1: Evolución Oracle (resumen)</div>		
V. 1	1978	No llega a ser lanzada comercialmente
V. 2	1979	basic SQL operations (desarrollada en ensamblador)
V. 3	1983	desarrollada en C; g. concurrencia (transaccional) y distribución
V. 4	1984	incorpora consistencia (en lecturas) + ed. doméstica (para PC)
V. 5	1985	arquitectura cliente-servidor y BBDD distribuidas
V. 6	1988	PL/SQL + row-level locks + hot backup
V. 7	1992	int. referencial + procedures + triggers + motor ConText + optimizador
V. 8	1997	ORDBMS (7.3) or. objetos + multimedia storage + Oracle Spatial
V. 8i	1999	interoperabilidad con internet + Aurora (máquina virtual Java)
V. 9i	2001	RAC (Real Application Cluster) + XML support
V. 10g	2003	adaptada a <i>grid computing</i>
V. 11g	2007	mejorado en casi todo (eficiencia, seguridad, comodidad,...) + Exadata
V. 12c	2013	adaptada a <i>cloud computing</i> + in-memory engine + json + multitenant
V. 18c	2018	<i>polymorphic table functions</i> (PTF) + más estándar
<div>© 2020 JCallé</div> <div>uc3m Universidad Carlos III de Madrid</div> <div>FFBDD – Tema 8: Sistema Gestor de BBDD</div> <div>8M - 6</div>		

SystemR- -SQL/DS-

DB2

-Informix-

-Sybase-

-Informix-

(DB2)

-SQL Server-

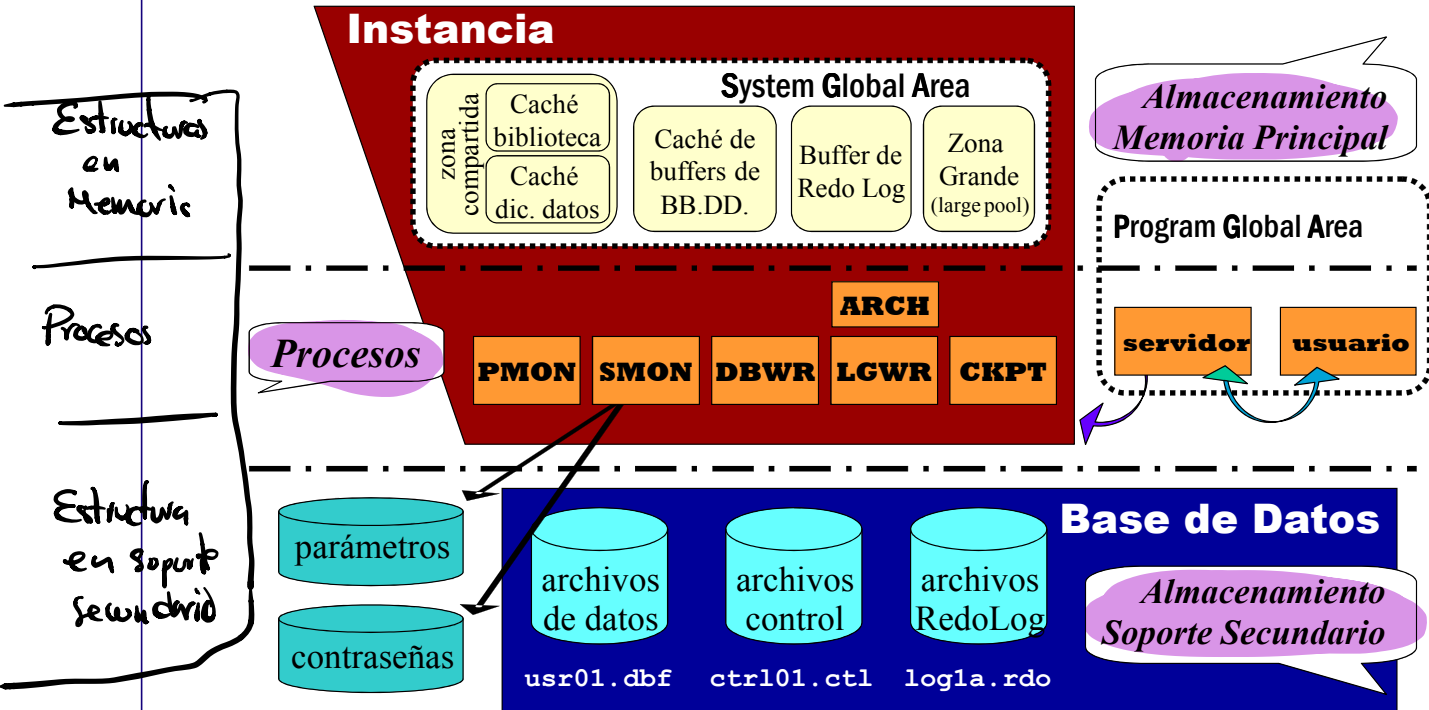
Instancia: (de B.D.) *Conjunto operativo*

- **Conjunto de procesos y estructuras** (físicas y en memoria).
- Proporciona **mecanismos de acceso y control de las BD.**
- Sus procesos son compartidos por todos los usuarios (de esa instancia).
- Las estructuras físicas se apoyan en el concepto de **tablespace**.
Cada tablespace podrá almacenarse en uno o más ficheros de datos.
- Las estructuras en memoria se organizan en dos áreas: **SGA y PGA**
- En un servidor de BB.DD. pueden existir varias instancias.

Base de Datos:

- **Conjunto de datos almacenado y accesible según una estructura lógica** (esquema relacional, en este caso, representado mediante tablas).
- Los elementos de que consta pueden pertenecer a uno o más usuarios, almacenarse en uno o más tablespaces, pero siempre en la misma instancia.
- Las BD de distintas instancias pueden federarse, para posibilitar su interrelación.

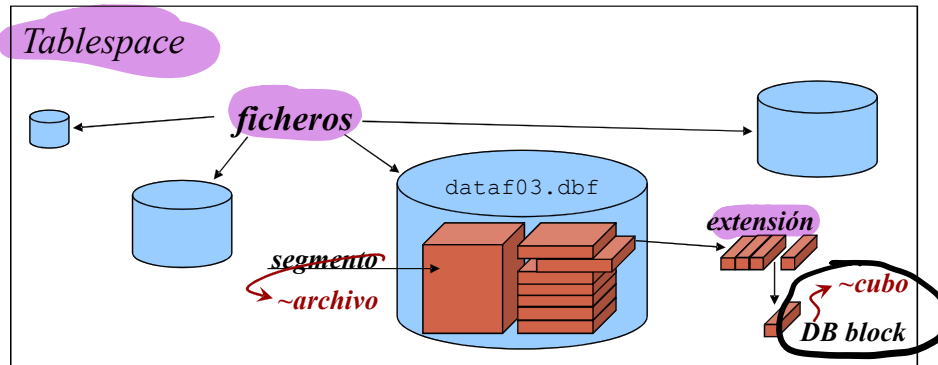
Tema 8.1.1: Arquitectura ORACLE®



Tablespaces y Datafiles

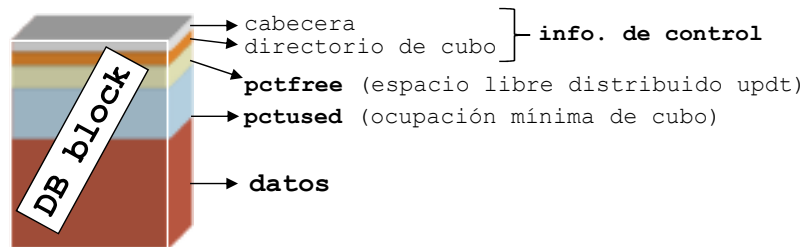
→ Unidad virtual de espacio. Es un gran cubo, con todas sus partes.

- El **tablespace** es un almacén de datos.
- Puede tener 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**.



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' \equiv 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.

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), pero no en SSD's
 - 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_{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*

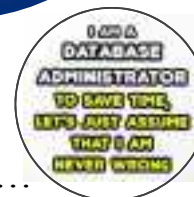
Tema 8.1.1: Ficheros de ORACLE®

- **Datos** (datafiles): almacenan los segmentos de la BD
- **Parámetros** (pfile / spfile): información para la inicializar la instancia.
- **Contraseñas**: información de acceso
- **Control**: contienen la información necesaria para la utilización de la BD (nombre BD, nombre y ubicación de ficheros, back-up, etc.)
- **Redo log**: protegen la BD, almacenando los cambios que debería haber experimentado la base (y aún no han sido completamente efectuados).

→ *journaling*: Almacena un registro de las operaciones que se hacen por si ocurre algún problema saber cual fue o restaurar posibles problemas

Tema 8.2: Administración de BB.DD.

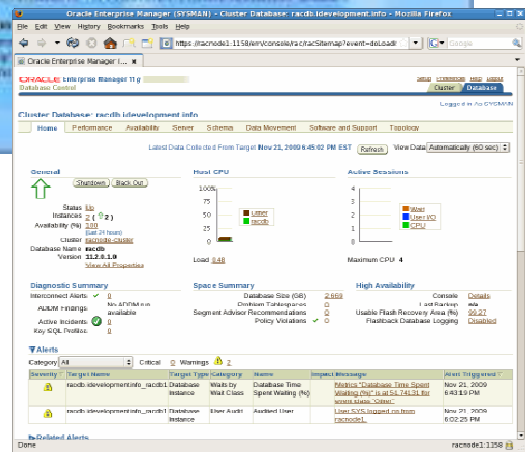
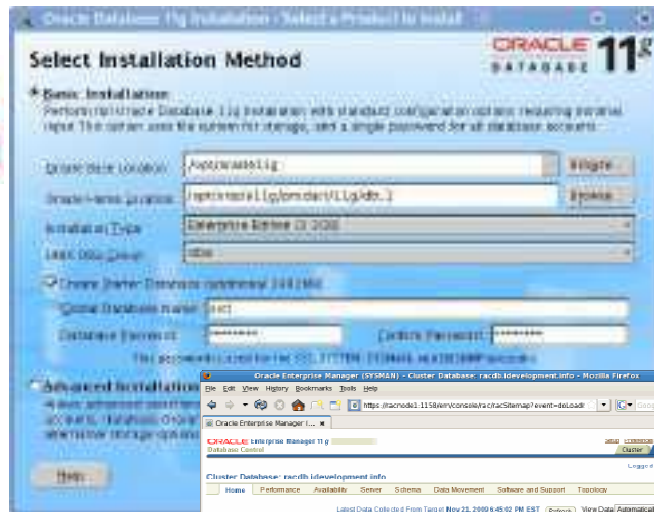
- La gestión de datos es el corazón de (casi toda) maquinaria empresarial. Garantizar su funcionamiento, seguridad y eficiencia es crucial.
- El administrador (DBA) es un profesional clave, con este perfil:
 - Conoce las tecnologías que soportan la BD (Hw, Sw, comm.)
 - Domina las tecnologías de BD (y los SGBD)
 - Idealmente, involucrado en el desarrollo
 - Conocedor de la estructura de la BD
 - Gestiona y posibilita el uso de la BD
 - Garantiza la integridad y seguridad de los datos
 - Garantiza la confidencialidad de la información
 - **Maximiza la eficiencia de la BD**
 - Otras competencias: sociales, organizativas, gestión personal, ...
- Otros perfiles: científico de datos, ingeniero sistemas información, ...



Tema 8.2: Administración en ORACLE®



fuentes: <http://docs.oracle.com>



Tema 8.2: Configuración local (conn)

- **Instancia:** conjunto de procesos y estructuras que albergan una BD
- **Conexión:** sesión establecida con un cliente para operar la BD
- **Definición SID:** se almacena en el cliente en el fichero tnsnames.ora

```
SID_local =  
(DESCRIPTION = (ADDRESS_LIST = (ADDRESS =  
                                (PROTOCOL=TCP) (HOST=localhost) (PORT=1521)  
                                )  
                (CONNECT_DATA = (SERVICE_NAME=SID_host))  
                )
```

- **Consola sql+:** aplicación que permite a un cliente conectarse con la BD

```
> sqlplus [username/password[@SID]] [AS role] [@script.ora] ...  
SQL> disconn[ect] /* sqlplus: commit & log out, without exiting */  
SQL> conn[ect] [username[/password][@SID] [AS role] ]  
SQL> exit          /* or quit; sqlplus: disconn and quit */
```

- **Usuario:** par name/passwd que da acceso a la BD según unos privilegios (ver diap. 4M9)
- **Privilegios:** acceso a operación de objetos de la BD, que pueden ser concedidos y revocados (ver diap. 4M.10)
- **Rol:** conjunto de privilegios; un usuario puede tener 0, 1 ó más

Roles predefinidos: sysoper; sysdba (este puede operar la instancia):

```
SQL> conn sys/admin@ORCL AS sysdba
SQL> shutdown [{ABORT|IMMEDIATE|TRANSACTIONAL|NORMAL}]
SQL> startup [pfile=rutalocal] [force] [nomount] [quiet] ...
SQL> disconn
SQL> conn / as sysdba
SQL> startup mount
```


Tema 8.2: Configuración de instancia

- La configuración se almacena en ficheros de parámetros (pfile/spfile)
 - pfile es textual (estático), y se edita con un editor de texto;
 - spfile es del servidor, y se actualiza con *alter system set variable=valor;* (scope={spfile|memory|both} indica si cambiar el fichero, el valor efectivo o ambos)
- La vista SYS.v\$parameter contiene los parámetros efectivos (la fila name='spfile' indica si se está aplicando un pfile (null) o un spfile). También pueden visualizarse con la instrucción *show parameters* (sql*plus)
- Algunos par. se pueden cambiar en *caliente*, y otros req. reinicializar
- Algunos ejemplos de parámetros relevantes:
 - open_cursors = int (0..65535) default 50
 - spfile = ruta
 - db_block_size = int (2048..32768); default 8192
 - db_cache_size = int {k|m|g}
 - db_{2k|4k|8k|16k|32k}_cache_size = int {k|m|g}
 - job_queue_processes = int (0..1000)
 - compatible, log_archive_dest, cluster_database, sessions, ...

http://docs.oracle.com/cd/B19306_01/server.102/b14237/initparams002.htm#CJAJHDED

- **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

- 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,...
```


- 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'`)

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 índices en ORACLE® es:

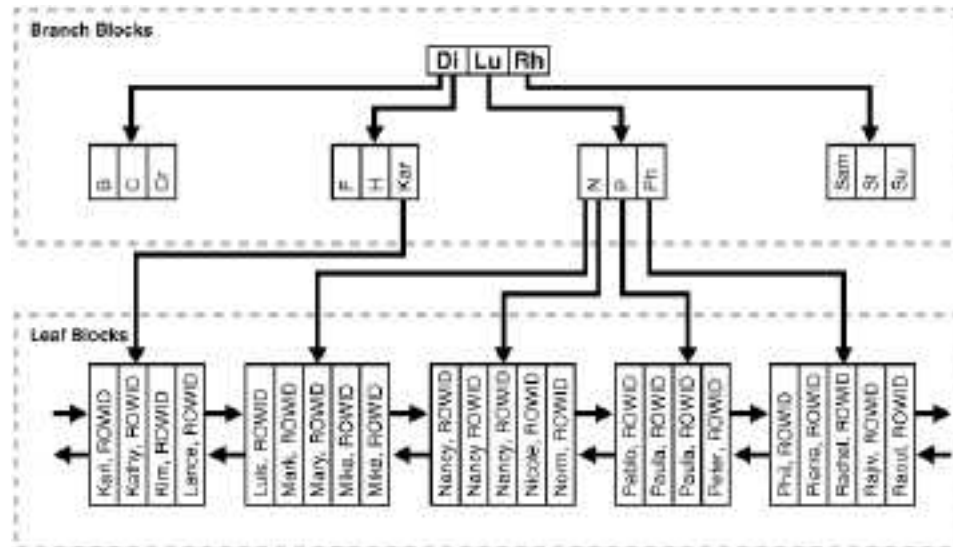
```
CREATE [ind_type] INDEX ind_name  
ON table_name(ind_key) [FROM ... ];
```

donde


`ind_type` := UNIQUE | BITMAP | (default)
`ind_key` := columna(s) de `table_name`, separadas por comas
(o una función sobre esas columnas...)

Indexación en Oracle®

- **Primario en B tree, secundario en B⁺ tree:**



Indexación en Oracle®

- Bitmap *clusterizado* Oracle®:
 - Debe aplicarse sobre objetos poco volátiles (tablas constantes o vistas materializadas reconstruidas periódicamente)
 - Comprime valores repetidos adyacentes (puede ser eficiente con $\#valores > 1\%$ si la clusterización es elevada)
 - Requiere un bit más por esquema (semántica: ‘new bm’/‘same as previous’)
 - Al tener tamaño reducido, mezclar índices es eficiente

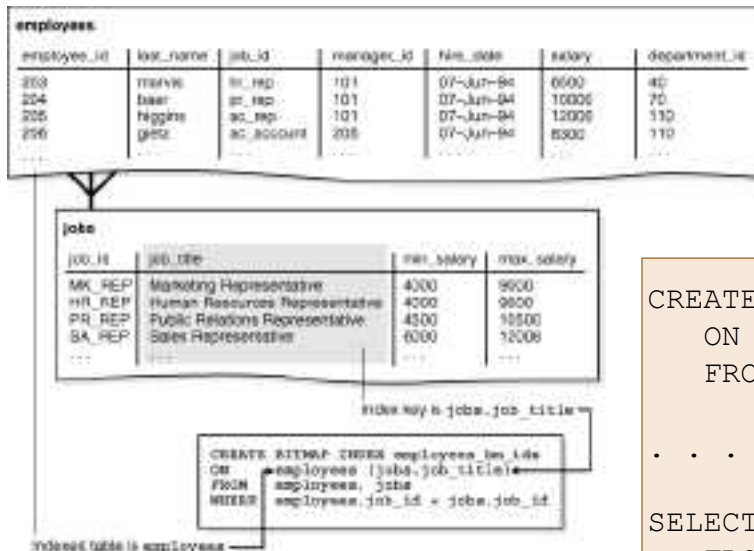
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
N	●	●	●							●	●	●	●			
S								●	●							
E				●	●	●	●									
W														●	●	●



	1	4	8	10	14
N	●			●	
S			●		
E		●			
W					●

Indexación en Oracle®

- Ejemplo de índice en Oracle®: índice sobre columna externa (*bitmap join index*)



```
CREATE BITMAP INDEX emp_jobs_idx
ON employees (jobs.job_title)
FROM employees JOIN jobs
USING (job_id);
```

...

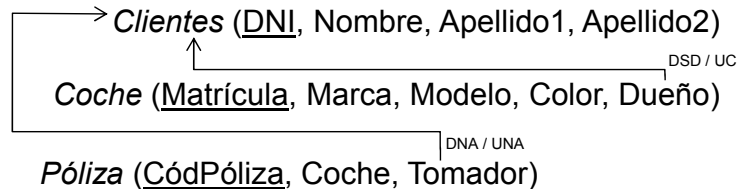
```
SELECT last_name, salary
FROM employees NATURAL JOIN jobs
WHERE jobs.job_title = 'Currito';
```


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

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) )*
);
  
```

Clusterización en Oracle®

Ejemplo 2:

```
CREATE CLUSTER emp_dept (deptno NUMBER(3))
SIZE 600
TABLESPACE users
STORAGE (INITIAL 200K NEXT 300K
MINEXTENTS 2 PCTINCREASE 33);
```

```
...
CREATE TABLE dept
(deptno NUMBER(3) PRIMARY KEY,
... )
CLUSTER emp_dept (deptno);
```

```
CREATE TABLE emp (
empno NUMBER(5) PRIMARY KEY,
ename VARCHAR2(15) NOT NULL,
... ,
deptno NUMBER(3) REFERENCES dept)
CLUSTER emp_dept (deptno);
...
```

Clustered Key (DEPTO)

10	ENAME	LOC
SALES	BOSTON	

EMPNO	ENAME	...
1000	SMITH	...
1021	JONES	...
1041	WARD	...

20	ENAME	LOC
ADMIN	NEW YORK	

EMPNO	ENAME	...
1001	KEHR	...
1139	WILSON	...
1277	NORMAN	...

EMP TABLE

EMPNO	ENAME	DEPTNO
1000	SMITH	10
1021	JONES	10
1041	WARD	10
1001	KEHR	20
1139	WILSON	20
1277	NORMAN	20
1321	JONES	10
1841	WARD	10

DEPT Table

DEPTNO	ENAME	LOC
10	SALES	BOSTON
20	ADMIN	NEW YORK



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.

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`)

HINTS en Oracle®

- Oracle puede que no elija un camino óptimo (no utiliza los índices creados o los utiliza no de manera eficiente)
- Los HINTS fuerzan el camino físico para resolver sentencias (select, insert, delete, update), son especificados como comentarios

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

- Se pueden especificar varios HINTS para la misma instrucción

- **Ejemplo:**

Si se especifica **index(clients)**, aplicará al menos un índice sobre la tabla clientes. Al añadir **index(clients ind1 ind2)** forzamos a que se aplique al menos uno de esos dos índices. En cambio, **and_equal(clients ind1 ind2)** fuerza a utilizar todos los índices especificados (en este caso, esos dos).

- Oracle posee alrededor de 75 hints documentados (iy otros 55 no documentados!)

HINTs Oracle® más usuales

Sintaxis del HINT	Descripción
<code>/*+ FULL(tablename) */</code>	full scan of table <i>tablename</i>
<code>/*+ ROWID(tablename) */</code>	rowid scan of table <i>tablename</i>
<code>/*+ INDEX(tablename [indexname [...]]) */</code>	use an index (or several of them)
<code>/*+ NO_INDEX(tablename) */</code>	forbids any index (on <i>tablename</i>)
<code>/*+ NO_INDEX(tablename [indexname [...]]) */</code>	forbids specific index/es
<code>/*+ INDEX_FFS(tablename [indexname [...]]) */</code>	full scan of the index
<code>/*+ AND_EQUAL(tablename [index1 index2 [...]]) */</code>	use more than one index (up to 5)
<code>/*+ INDEX_JOIN(tablename [indexname [...]]) */</code>	join indexes (sort of inverted access)
<code>/*+ CLUSTER(tablename) */</code>	use cluster for <i>tablename</i>
<code>/*+ HASH(tablename) */</code>	use a hash for clustered <i>tablename</i>
<code>insert /*+ append */ ... select...</code>	inserción directa (no buffer, no fl, no RI,...)

- Para el desarrollo de aplicaciones complejas, existen lenguajes de programación capaces de conectar y operar BD relacionales: Pro*C, Java...
- En Java se usa la API JDBC (Java DataBase Connectivity)
- El paquete `java.sql` proporciona las clases que lo implementan
<http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-10201-088211.html>

DriverManager, SQLException, Connection, Statement, ResultSet, ...

- Para usar BD Oracle, es necesario contar además con el manejador jdbc para Oracle (que debe registrarse) y las clases de Oracle SQL:

```
import java.sql.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
DriverManager.registerDriver(
    new oracle.jdbc.driver.OracleDriver());
```


Tema 8.3: Conexión por JDBC

- El objeto *conexión* establece un canal de comunicación para enviar instrucciones SQL y obtener el resultado

- Puede hacerse con el driver *oci* de Oracle, utilizando la definición local de instancias (*tnsnames*)... *Si hay una preconfiguración*

```
Connection conexion = DriverManager.getConnection(
    "jdbc:oracle:oci8:@sid", "username", "password");
```

Lo objeto de comunicación en Java. jdbc---

- ... o bien mediante el driver *thin* (definición explícita) *Si no hay preconfiguración.*

```
Connection conxn2= DriverManager.getConnection(
    "jdbc:oracle:thin:IP:puerto:srv", "username", "password");
```

Instanciación *servicio remoto* *fib...*

- Conviene imbuirlo en `try {...} catch {...}` para manejar excepciones (`SQLException`), por si el servidor rechaza la conexión, está caído, ...
- Al finalizar, debe cerrarse el objeto: `conexion.close();`

Tema 8.3: Instrucciones y Resultados

- A través de una conexión, se puede instanciar una instrucción...

```
Statement instruccion = conexion.createStatement();
```

crear inst.
Lo sobre la instanciación correcta.

- ...que podemos ejecutar y de la que podemos obtener su resultado:

Instanciamos un resultado

```
ResultSet resultado = instruccion.executeQuery(  
    "select * from dual");
```

ejecutor...

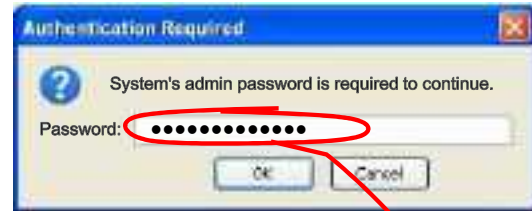
- Ese objeto *ResultSet* puede consultarse y actualizarse
<http://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html>
- *ResultSet* se recorre fila a fila (mantiene un puntero actualizable por *first*, *last*, *relative(int)*, *next*, *previous*), y de cada fila se puede obtener cualquier columna:

```
While resultado.next()  
    {System.out.print(resultado.getString(1));}
```

- Al finalizar, deben cerrarse los objetos:
- ```
resultado.close();
instruccion.close();
```

## Tema 8.3: Precauciones - Inyección

- Extraer texto libre de un formulario y concatenarlo a una instrucción es muy peligroso, porque el usuario puede escribir lo que quiera...



```
ResultSet resultado = instruccion.executeQuery(
 "SELECT sueldo FROM nominas
 WHERE EXISTS (SELECT 'x' FROM credentials
 WHERE usr='SYS' AND (passw='\" | (txt) | '\");"
```

- En este ejemplo, ¿Qué pasaría si el usuario escribe...?

' or 'x'='x  
Escapa el control, trata de poder dejar que sea x

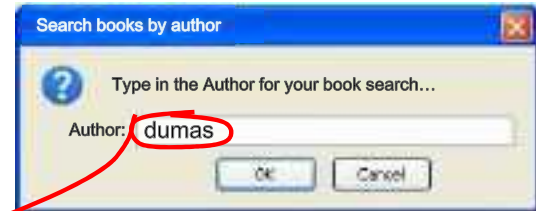
- **Precauciones:** procesa las cadenas de texto procedentes de formularios, asegurando la literalidad de caracteres de control (escape):

```
searchWord.replace("\\", "\\\\") .replace("'", "\\'")
```

## Tema 8.3: Precauciones - Inyección

- Alguien malintencionado podría alterar el resultado de una consulta.

```
..."SELECT title from books
 where author = ' "|txt|"'..."
```



- ¿Y si el "autor" que busco fuera...?

`' UNION SELECT table_name FROM user_tables; --` *conecta lo siguiente.*

- En ese primer paso veo que existe una tabla 'orders', y ahora "busco"...

```
' ; DROP TABLE orders; --
```

- **Precauciones:**
- protege tus **metadatos** y **estructuras** controlando los privilegios.
- Crea usuarios específicos para cada aplicación y otorga **privilegios** mínimos.
- Crea **vistas** para consultas, y tablas **to\_do** para escrituras (el usuario registra la operación a realizar, y otro proceso lee la tabla, analiza la acción y la hace si procede).

## Tema 8.3: Recuerda a *Bobby Tables*

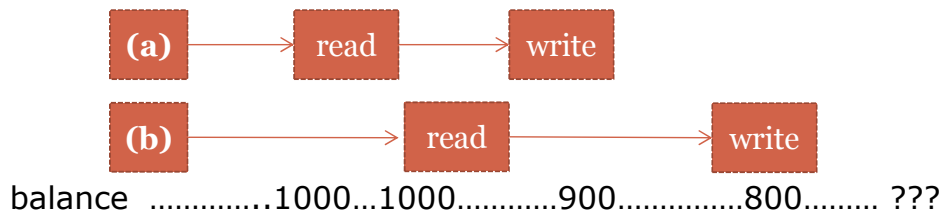


- **Concurrencia:** varios usuarios operan la BD simultáneamente.
- **Condición de Carrera:** si una operación requiere varios pasos afectando alguno a un recurso compartido (en este caso, el estado de la BD), y el resultado depende de la secuencia que está afectada por otro proceso que opera el mismo recurso.

Ejemplo: proc. a) `UPDATE account SET balance=balance-100;`

proc. b) `UPDATE account SET balance=balance-200;`

Ambos procesos requieren leer el estado (p.e., balance=1000) y después escribir el resultado; si ambos leen a la vez y luego escriben su resultado parcial, el resultado final podría ser erróneo (800 ó 900, en lugar de 700).



- En BD, las carreras pueden afectar incluso a la integridad de los datos.

Ejemplos:

*falta de entidad*

a) `DELETE client WHERE DNI=123;`

b) `INSERT INTO car (plate,owner) VALUES (0000AEI, 123);`



*falta de integridad*

a) `INSERT INTO client (DNI,name) VALUES (123, 'John');`

b) `INSERT INTO client (DNI,name) VALUES (123, 'Mary');`

*lectura sucia*

a) `SELECT saldo INTO var FROM accounts WHERE DNI=123;`

b) `UPDATE accounts SET saldo=saldo-100 WHERE DNI=123;`

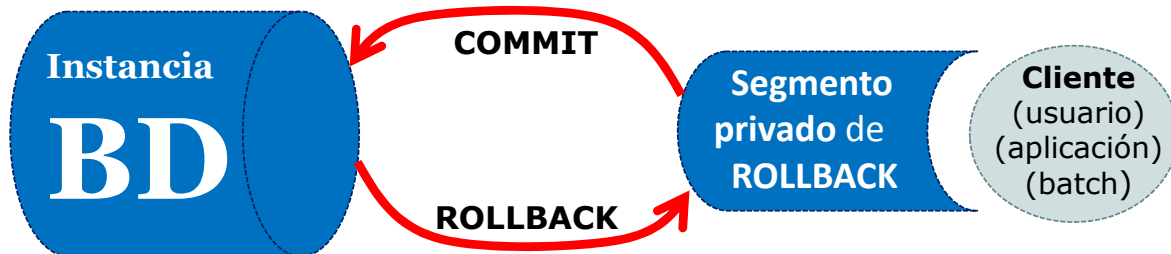
- Para resolverlo, se define la *granularidad* al nivel de *transacción*.
  - granularidad de ejecución: especificidad con la que se definen las *ejecuciones atómicas* (cuya secuencia no puede ser interrumpida).
  - transacción: secuencia de operaciones que se opera en conjunto (indivisible o atómica), pudiendo ser perpetrada o desestimada. *Comienza* con la *primera* instrucción DML (o con *set transaction*) y *finaliza* al perpetrarse o desestimarse (también con el *fin de sesión*).
  - granularidad de bloqueo: esquema, tabla, columna, fila, ...

## Tema 8.4: Concurrency - Multicopia

- **Multicopia:** para aliviar la gestión de la concurrencia en el servidor, se puede mantener una copia (virtual) de la BD para cada sesión abierta.
  - Las actualizaciones no perpetradas que se operen en una sesión es almacenado en un segmento de **rollback**.
  - El estado de la BD visible desde cierta sesión es el estado general más las operaciones en el segmento de rollback.
  - Desestimar una transacción equivale a vaciar ese segmento.
  - Perpetrar una transacción equivale a ejecutar de modo atómico ese segmento sobre la BD.
  - A medida que se definen las operaciones de la transacción, se bloquean los recursos afectados con los correspondientes **cerrojos**
  - Los cerrojos se eliminan al finalizar la transacción.



- **Transacción**: conjunto de instrucciones de **actualización** que deben ser llevadas a cabo de modo atómico (como conjunto, "o todo o nada")
- **Autocommit**: define que todas las transacciones son mono-instrucción.
- **Instrucciones**: COMMIT (realizar) y ROLLBACK (deshacer)
  - COMMIT [WORK]
  - ROLLBACK [WORK] [TO [SAVEPOINT] <savepoint>]
  - SAVEPOINT <savepoint>



## Tema 8.4: Cerrojos en Oracle

Normalmente  
Nowait, no espera.

- Los cerrojos permiten algunas operaciones y otras no.
  - Si una operación necesita acceder a un recurso bloqueado (para ese tipo de operación) deberá esperar a que se libere (si la operación lleva la opción **NOWAIT** devuelve inmediatamente el control con un error).
  - También puede esperar una cantidad de tiempo dada en segundos.
- **Cerrojo de datos** (DML Lock): bloquea datos (nivel de tabla o de fila). Pueden ser automáticos o creados por el usuario.
- **Cerrojo de Catálogo** (DDL Lock): bloquea la estructura (tablas o vistas) para evitar cambios DDL sobre una estructura que está siendo operada
  - **Cerrojo interno** (Latch): bloquea estructuras internas de la BD (blocks)
  - **Cerrojo distribuido**: asegura la consistencia entre varias instancias de un servidor distribuido. En particular, un PCM (parallel ce management) es un bloqueo distribuido sobre uno o más bloques de datos (de tablas o de índices) en la memoria intermedia (buffer).

## Tema 8.4: Cerrojos de Datos en Oracle

- Algunos cerrojos son de creación automática (RX), y todos ellos pueden crearse con la instrucción: `LOCK TABLE <tablename> IN <mode>`
- Todos los cerrojos permiten consultar (lectura consistente: en caso de que la tabla esté siendo modificada, se accede a la versión anterior).
- Contemplan dos niveles (tabla/fila) y dos modos (compartido/exclusivo)
  - fila

    - Row Share (**RS**): el menos restrictivo; impide que otra transacción obtenga un cerrojo exclusivo sobre determinadas filas. Se puede obtener con `SELECT... FOR UPDATE [OF column]` (cerrojo de cursor; con `OF column` afecta sólo a las tablas con atributos enlistados).
    - Row Exclusive (**RX**): como el (X) pero sólo sobre determinadas filas (Oracle aplica este por defecto durante INSERT/UPDATE/DELETE).
  - tabla

    - Share (**S**): bloquea una tabla, impidiendo que otras transacciones obtengan un cerrojo exclusivo (impide actualizaciones).
    - Share Row Exclusive (**SRX**): idem, pero además impide (S) a otros
    - Exclusive (**X**): es el más restrictivo; bloquea todo (menos la query).

# Tema 8.4: Compatibilidad de cerrojos

| other users<br>can...<br>a user has... | X | SRX | S | RX | RS |
|----------------------------------------|---|-----|---|----|----|
| RS                                     | × | ✓   | ✓ | ✓  | ✓  |
| RX                                     | × | ×   | × | ✓  | ✓  |
| S                                      | × | ×   | ✓ | ×  | ✓  |
| SRX                                    | × | ×   | × | ×  | ✓  |
| X                                      | × | ×   | × | ×  | ×  |

\* sobre filas diferentes

- Algunos DBMS escalan cerrojos (de fila a tabla), pero Oracle no.
- ¡Cuidado con los cerrojos restrictivos en transacciones largas!
- Interbloqueo: dos transacciones bloqueando sendos recursos y tratando de bloquear el otro recurso quedan interbloqueadas (deadlock). Para evitarlo, Oracle resuelve cancelar una de ellas (por marcas de tiempo).
- En transacciones distribuidas, Oracle puede confundir los interbloqueos con bloqueos largos (resuelve con un temporizador en trans. distribuidas).