

# Unidad 1



Centro Don Bosco  
Villamuriel de Cerrato

# Instalación y Configuración de Bases de Datos

usando Oracle 11g

Apuntes realizados para la asignatura de FP Grado Superior:  
**Administración de Bases de Datos**  
del ciclo Administración de Sistemas Informáticos en Red

**Autor: Jorge Sánchez Asenjo** ([www.jorgesanchez.net](http://www.jorgesanchez.net))  
Versión del documento: 2.15, Año 2012



Esta obra está bajo una licencia de Reconocimiento-NoComercial-CompartirIgual de Creative Commons.  
Para ver una copia de esta licencia, visite: <http://creativecommons.org/licenses/by-nc-sa/3.0/deed.es>





## Atribución-NoComercial-CompartirIgual 3.0 Unported (CC BY-NC-SA 3.0)

Esto es un resumen fácilmente legible del [Texto Legal \(la licencia completa\)](http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode).

<http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode>

### Usted es libre de:

Compartir - copiar, distribuir, ejecutar y comunicar públicamente la obra  
hacer obras derivadas

### Bajo las condiciones siguientes:



**Atribución** — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciante (pero no de una manera que sugiera que tiene su apoyo o que apoyan el uso que hace de su obra).



**No Comercial** — No puede utilizar esta obra para fines comerciales.



**Compartir bajo la Misma Licencia** — Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

### Entendiendo que:

**Renuncia** — alguna de estas condiciones puede **no aplicarse** si se obtiene el permiso del titular de los derechos de autor

**Dominio Público** — Cuando la obra o alguno de sus elementos se halle en el **dominio público** según la ley vigente aplicable, esta situación no quedará afectada por la licencia.

**Otros derechos** — Los derechos siguientes no quedan afectados por la licencia de ninguna manera:

- Los derechos derivados de **usos legítimos** u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.
- Los derechos **morales** del autor;
- Derechos que pueden ostentar otras personas sobre la propia obra o su uso, como por ejemplo **derechos de imagen** o de privacidad.





# índice

<b>(1.1) estructura de un SGBD</b>	<b>7</b>
(1.1.1) Sistemas Gestores de Bases de Datos	7
(1.1.2) repaso a los niveles conceptuales. modelo ANSI/X3/SPARC	7
(1.1.3) funciones del SGBD	9
(1.1.4) funciones avanzadas de un SGBD	9
(1.1.5) tareas del DBA	11
<b>(1.2) opciones de funcionamiento de un SGBD</b>	<b>12</b>
(1.2.1) SGBD monocapa	12
(1.2.2) SGBD de dos capas	12
(1.2.3) SGBD de tres o más capas	12
<b>(1.3) Sistemas Gestores de Bases de Datos Comerciales</b>	<b>12</b>
(1.3.1) licencias de software	12
(1.3.2) SGBD relacionales de código cerrado	13
(1.3.3) SGBD relacionales de código abierto	14
(1.3.4) bases de datos NoSQL	15
<b>(1.4) arquitectura de un SGBD</b>	<b>18</b>
(1.4.1) ¿qué es la arquitectura?	18
(1.4.2) estructuras lógicas de la base de datos	18
(1.4.3) estructuras físicas e internas de la base de datos	18
(1.4.4) instancias de bases de datos	18
<b>(1.5) arquitectura de Oracle</b>	<b>19</b>
(1.5.1) ¿qué es un servidor Oracle?	19
(1.5.2) arquitectura en memoria del servidor Oracle	20
(1.5.3) arquitectura en disco de Oracle	22
<b>(1.6) instalación de SGBD</b>	<b>25</b>
(1.6.1) paso 1: selección por requisitos	25
(1.6.2) paso 2: comprobar requerimientos	26
<b>(1.7) instalación de Oracle</b>	<b>26</b>
(1.7.1) documentación	26
(1.7.2) directorios de Oracle. comprender la estructura OFA	26
(1.7.3) instalación en Windows	28
(1.7.4) crear la base de datos	31
(1.7.5) conectar con la base de datos	35
(1.7.6) control de la instancia de Oracle	36
(1.7.7) control del listener	36
(1.7.8) conexión en modo administrador	37
(1.7.9) manejo de la instancia de la base de datos	37
(1.7.10) borrado de la base de datos	38
(1.7.11) Enterprise Manager	38

<b>(1.8)</b> diccionario de datos en Oracle	39
<b>(1.9)</b> configuración de Oracle	40
(1.9.2) ubicación del archivo de parámetros	40
(1.9.3) algunos parámetros	40
(1.9.4) gestión de los archivos de parámetros	41
(1.9.5) modificación de parámetros en caliente	42
<b>(1.10)</b> ficheros LOG en Oracle	42
<b>(1.11)</b> APÉNDICE: instalación de MySQL	43
(1.11.1) pasos previos	43
(1.11.2) documentación	43
(1.11.3) instalación en Windows	43
(1.11.4) instalación en Linux/Unix	48
(1.11.5) asegurando la instalación	53
(1.11.6) diccionario de datos de MySQL	54
(1.11.7) configuración de MySQL. Uso de opciones	55
(1.11.8) ficheros LOG en MySQL	57

# (1) instalación y configuración de un Sistema Gestor de Bases de Datos

## (1.1) estructura de un SGBD

### (1.1.1) Sistemas Gestores de Bases de Datos

Un Sistema Gestor de Base de Datos es el software que permite gestionar bases de datos, ocultando la física de la misma y permitiendo manejarla desde un nivel más conceptual. Dicho software permite separar las aplicaciones (los programas) de los datos; de modo que los programas negocian con el SGBD el acceso a los datos.

En definitiva se trata de un software complejo, pero de gran importancia por lo delicado de la rama de la información a la que se dedica. Los SGBD han crecido de manera exponencial estos últimos años por el éxito de Internet, que ha provocado el acceso a miles y miles de bases de datos por parte de millones de usuarios cada día.

### (1.1.2) repaso a los niveles conceptuales. modelo ANSI/X3/SPARC

El grupo de trabajo SPARC de la sección X3 de ANSI, diseñó un modelo en el que indicaba cómo debía funcionar un SGBD para asegurar la separación entre datos y aplicaciones y así especificó tres niveles:

- **Nivel externo.** Define el nivel en el que los usuarios y usuarias utilizan la base de datos. La forma de ver la misma que no tiene nada que ver con la estructura real de la base de datos. Los usuarios manejan los esquemas externos de este nivel, pero dichos esquemas los realizan los desarrolladores/as de aplicaciones.

- **Nivel físico.** Se refiere a la forma en la que realmente se almacena la información de la base de datos. Los administradores (DBA) de la base de datos son los encargados de manejar este nivel
- **Nivel conceptual.** Define la base de datos haciendo referencia a la forma en la que se relaciona la información. Es un nivel más humano que el físico, pero no tanto como el externo:

Hoy en día se definen más niveles en realidad de modo que se habla de cinco niveles en realidad. Empezando desde el más cercano al usuario:

- **Nivel externo.** Tal cual se explicó antes. En realidad los esquemas de este nivel son los últimos que se crean y lo hacen programadoras/es y analistas bajo la dirección de las/os analistas.
- **Nivel conceptual.** Con la misma idea indicada anteriormente. En realidad el esquema (los planos) conceptual de la base de datos es lo primero que se diseña por los o las analistas o diseñadores de la misma utilizando un modelo para realizar los esquemas. El modelo **Entidad/Relación** sigue siendo el modelo más popular.

- **Nivel lógico.** Acerca más el esquema anterior a la física de la base de datos. En este nivel se hace referencia a estructuras lógicas del tipo de SGBD a manejar (tablas, filas, columnas por ejemplo en el modelo relacional de bases de datos, el modelo lógico más utilizado). Este nivel sigue siendo manejado por los analistas. En muchos casos (aunque ciertamente es peligroso) los diseñadores/as de la base de datos empiezan por este nivel saltándose el anterior.

En la actualidad el modelo relacional sigue siendo el modelo más habitual para crear esquemas a nivel lógico.

- **Nivel interno.** Es el primero en el proceso de modelado de la base de datos que se realiza sobre el software gestor de la base de datos (teniendo en cuenta que lo externo, las aplicaciones, se crean más tarde). Usa el lenguaje de la base de datos para crear las estructuras de datos definidas en el nivel lógico. Este nivel lo maneja el administrador de la base de datos (o DBA).
- **Nivel físico.** Se refiere a como se organizarán los datos en el disco, en que ordenadores se crea la base de datos, si es distribuida o no, sistema operativo necesario, estructura de directorios y archivos, configuración de servidores y sistema operativo, política de copia de seguridad,...

La persona encargada de definir todo esto es la administradora de la base de datos o, mejor dicho, la administradora del sistema. En realidad hay acciones referidas a este nivel que se hacen antes que las del nivel anterior (las re; otras se irán haciendo después o a la vez.



### (1.1.3) funciones del SGBD

El manejo de la SGBD implica que nos permita realizar estas funciones:

■ **Función de descripción o definición.** Permite al diseñador de la base de datos crear las estructuras apropiadas para integrar adecuadamente los datos. Esta función se realiza mediante el **lenguaje de descripción de datos** o **DDL**. Mediante ese lenguaje:

- Se definen las estructuras de datos (los metadatos)
- Se definen las relaciones entre ellas
- Se definen las reglas que han de cumplir

■ **Función de manipulación.** Permite modificar y utilizar los datos de la base de datos. Se realiza mediante el **lenguaje de modificación de datos** o **DML**. Mediante ese lenguaje se puede:

- Añadir datos
- Eliminar datos
- Modificar datos
- Buscar datos

Actualmente se suele distinguir aparte la función de buscar datos en la base de datos (**función de consulta**). Para lo cual se proporciona un **lenguaje de consulta de datos** o **DQL**. También las transacciones (que usan lenguaje **DTL**) se consideran parte de esta función.

■ **Función de control.** Mediante esta función los administradores poseen mecanismos para proteger las visiones de los datos permitidas a cada usuario, además de proporcionar elementos de creación y modificación de esos usuarios. El lenguaje que implementa esta función es el **lenguaje de control de datos** o **DCL**.

### (1.1.4) funciones avanzadas de un SGBD

Para poder garantizar que una SGBD cumple las funciones anteriores de la mejor forma posible y además seguir garantizando la independencia entre los tres esquemas fundamentales (externo, conceptual y físico) y además facilitar su manejo, los SGBD tienen que cumplir de forma estricta una serie de reglas.

**Edgar F. Codd** (que teorizó el modelo relacional de la base de datos) escribió 12 reglas<sup>1</sup> que habían de cumplir todos los SGBD. Hoy en día casi todas las grandes bases de datos son relacionales y los SGBD se afanan en cumplir esas reglas, aunque con matices puesto que la realidad actual está haciendo quedar a este modelo un tanto obsoleto en algunos aspectos.

Hoy en día las funciones que se esperan de un buen SGBD son:

---

<sup>1</sup> Se pueden consultar en <http://www.jorgesanchez.net/bd> en el "Manual de Gestión de Bases de Datos usando SQL y Oracle" capítulo 2.4: "Las 12 reglas de Codd"

- Lenguaje que permita crear todos los elementos de la base de datos y gestionar el diccionario de datos. Normalmente este lenguaje será SQL (aunque cada SGBD impone variantes al SQL estándar)
- Herramientas gráficas que faciliten muchas tareas habituales tanto de gestión como de administración del sistema.
- Posibilidad de **establecer reglas de integridad avanzadas** e incluirlas como parte de la base de datos. Especialmente complicado suele ser poder establecer las reglas de integridad referencial (**foreign key**), por lo que algunos SGBD más simples no lo incluyen. Dentro de estas reglas están las restricciones estándar como: UNIQUE (unicidad, prohibir repetición de valores), CHECK (cumplimiento de condiciones simples), **NOT NULL** (obligatoriedad), **PRIMARY KEY** (establecimiento de las claves de las tablas) o la propia **FOREIGN KEY** (clave foránea); pero también restricciones más complejas como las que establecen los **Triggers** de los lenguajes procedimentales (como **PL/SQL**) presentes en la mayoría de sistemas.
- Gestión de **copias de seguridad**. Una de las funciones críticas de la base de datos ya que permite la recuperación de información en caso de problemas.
- Aplicaciones de **exportación e importación de datos**. Para poder utilizar datos de otros SGBD u otro software.
- Posibilidad de **recuperación en caso de desastre**. Para evitar perder información en caso de problemas serios con el software (errores de hardware, apagones prolongados,...)
- **Archivos LOG**. Desde los que podemos examinar las incidencias y monitorizar el funcionamiento de la base de datos.
- **Herramientas para programar aplicaciones**. Que permitan crear las aplicaciones (o facilidades) de usuario.
- **Gestión de la comunicación** con los clientes de la base de datos. Permiten establecer conexión con la base de datos desde máquinas remotas.
- **Optimización de consultas**. Busca el mínimo tiempo de respuesta para las operaciones sobre los datos.
- **Herramientas para automatizar tareas**. Permiten realizar programaciones sobre operaciones habituales sobre la base de datos.
- Posibilidad de **distribuir la base de datos** entre diferentes máquinas, y así mejorar su alta disponibilidad.
- **Gestión de transacciones, ACID**. ACID significa **Atomicity, Consistency, Isolation and Durability**: **Atomicidad, Consistencia, Aislamiento y Durabilidad** en español y es una norma obligatoria que deben de cumplir las bases de datos para que una transacción se pueda considerar como tal.

### (1.1.5) tareas del DBA

Un administrador de bases de datos (DBA), tiene una serie de tareas asignadas. Todas ellas son de vital importancia para la base de datos; algunas son especialmente críticas. Las tareas más comúnmente aceptadas como parte de la profesión del DBA son:

- (1) **Configurar e instalar el hardware necesario.** Para el correcto funcionamiento del SGBD. Eso incluye ampliar memoria, discos duros,... además de configurarles para su óptimo rendimiento. También la gestión mínima del sistema operativo para que la base de datos funcione correcta y rápidamente.
- (2) **Instalación y mantenimiento del SGBD.** Seleccionando la más adecuada forma de instalación y configurando lo necesario para su óptimo rendimiento acorde con las necesidades, así como realizar las actualizaciones del sistema que sean necesarias.
- (3) **Crear las estructuras de almacenamiento de la base de datos.** Es quizá la tarea que de forma más habitual se relaciona con los DBA. Consiste en crear y configurar las estructuras físicas y los elementos lógicos que permitan un rendimiento optimizado de la base de datos.
- (4) **Crear y configurar la base de datos.** Creación de la estructura interna de la base de datos (tablas, usuarios, permisos, vistas,...). Es otra de las tareas más habitualmente relacionadas con el DBA y la primera fase (y la más crítica) en la administración de una base de datos.
- (5) **Control de los usuarios y los permisos.** En definitiva establecer las políticas de seguridad tan imprescindibles en toda base de datos.
- (6) **Monitorizar y optimizar el rendimiento de la base de datos.** Un DBA debe detectar los *cuellos de botella* del sistema y actuar en consecuencia. Esto incluye optimizar las instrucciones SQL por lo que implica asistir a los desarrolladores para que utilicen las instrucciones más eficientes sobre las bases de datos.
- (7) **Realizar tareas de copia de seguridad y recuperación.** Quizá la tarea más crítica. Consiste en realizar acciones para en caso de catástrofe poder recuperar todos los datos

## **(1.2) opciones de funcionamiento de un SGBD**

### **(1.2.1) SGBD monocapa**

Se trata de Sistemas Gestores instalados en una máquina desde la que se conectan los propios usuarios y administradores.

Es un modelo que solo se utiliza con bases de datos pequeñas y poca cantidad de conexiones.

### **(1.2.2) SGBD de dos capas**

Es el modelo tipo **cliente/servidor**. La base de datos y el sistema gestor se alojan en un servidor al cual se conectan los usuarios desde máquinas clientes. Un software de comunicaciones se encarga de permitir el acceso a través de la red. Los clientes deben instalar el software cliente de acceso según las instrucciones de configuración del administrador.

Hay dos posibilidades:

- Arquitectura **cliente/servidor único**. Un solo servidor gestiona la base de datos, todos los clientes se conectan a él para realizar las peticiones a la base de datos.
- Arquitectura **cliente/multiservidor**. La base de datos se distribuye entre varios servidores. El cliente no sabe realmente a que servidor se conecta, el software de control de comunicaciones se encargará de dirigirle al servidor adecuado. De forma lógica, es como si se tratara de un solo servidor aunque físicamente sean muchos.

### **(1.2.3) SGBD de tres o más capas**

En este caso entre el cliente y el servidor hay al menos una capa intermedia (puede haber varias). Esa capa (o capas) se encargan de recoger las peticiones de los clientes y luego de comunicarse con el servidor (o servidores) para recibir la respuesta y enviarla al cliente.

El caso típico es que la capa intermedia sea un servidor web, que recibe las peticiones a través de páginas web; de este modo para conectarse a la base de datos, el usuario solo requiere un navegador web, que es un software muy habitual en cualquier máquina.

Este modelo es el que más se está potenciando en la actualidad por motivos de seguridad y portabilidad de la base de datos.

## **(1.3) Sistemas Gestores de Bases de Datos Comerciales**

### **(1.3.1) licencias de software**

El gurú del software libre, **Richard Stallman** considera al **software propietario** (software cuyo uso y explotación se rige por un contrato propio de la empresa), software privativo, puesto que dicho software no permite examinar el código fuente con

el que se creo y, por lo tanto, impide modificar el mismo y adaptarlo a nuevas funcionalidades.

Por otro lado, él mismo define al software que sí permite este proceso, **software libre**. En cualquier caso ambos tipos de software no tienen por qué ser gratuitos, es decir la diferencia no es la gratuidad (aunque sí ocurre a menudo que el software de código abierto además suele ser gratuito) sino la libertad de utilizar el código fuente del software.

Una definición quizá menos tendenciosa es la que diferencia al software en: **software de código abierto** y **software de código cerrado u oculto**. Esta diferencia de software se debe a dos formas diferentes de hacer negocio con él; los defensores del código cerrado argumentan que es lógico protegerle para evitar copiar su tecnología por parte de la competencia e incluso por razones de seguridad del mismo al no poder asegurar su correcto funcionamiento ante modificaciones de terceros.

Los defensores del segundo modelo están a favor de la versatilidad del código abierto que permite poder modificar el código por parte de miles de programadores en todo el mundo que pueden compartir dichas mejoras y así mejorar enormemente y de manera dinámica el producto.

### (1.3.2) SGBD relacionales de código cerrado

Normalmente las licencias de uso de Sistemas Gestores de Bases de Datos con código cerrado usan licencias tipo **CLUF** o **EULA** (en inglés), acrónimo **contrato de licencia de usuario final**.

En estas licencias, el usuario firma unas condiciones de uso por el software, entre las que siempre figuran el hecho de no poder distribuir libremente el mismo y que está restringido a unas condiciones de trabajo concretas (por ejemplo el hecho de que normalmente sólo se pueda utilizar cada licencia en una sola máquina o por parte de un solo usuario). Ejemplos de SGBD de este tipo son:

- **Oracle**. Propiedad de Oracle Corporation. Es el SGBD más veterano y más influyente ya que la mayoría de mejoras al SQL original se desarrollaron para este SGBD. Sigue siendo uno de los SGBD comerciales más utilizados y además posee una gran relación con el lenguaje Java, acrecentada por la compra de la empresa creadora del mismo, **Sun Microsystems**.  
Presume de su gran estabilidad y escalabilidad, un control avanzado de transacciones y de sus lenguajes internos de manejo, especialmente famoso es su lenguaje procedimental PL/SQL. Es un SGBD multiplataforma, que tiene certificación para instalarse en Linux (aunque sólo con los compatibles con **Red Hat** y con condiciones muy concretas de instalación).
- **DB2**. Propiedad de IBM, es una de las bases de datos comerciales más populares. Desarrollada para Windows, UNIX y Linux. Implementa XML de manera nativa y dispone de amplias facilidades de migración de datos (especialmente desde Oracle) así como uso de transacciones avanzadas.
- **SQL Server (de Microsoft)**. Originalmente basado en el código del SGBD **SyBase** que Microsoft compró al propietario de **SyBase**, ahora es un SGBD distinto distribuido sólo para Windows y que compite con los dos anteriores. Dispone de una gran escalabilidad, estabilidad, uso de transacciones, entorno gráfico avanzado y de éxito entre los programadores de la plataforma .NET (también de Microsoft por su compatibilidad con esta).



Las tres son de las bases de datos más utilizadas en la actualidad por su contrastada potencia. Ninguna de las tres cumple completamente los estándares y aportan sus propios lenguajes y forma de trabajo. Además las tres disponen de versiones gratuitas para uso personal con base de datos más pequeñas.

### (1.3.3) SGBD relacionales de código abierto

Algunas utilizan la licencia **GNU GPL** (*GNU General Public License*), que utilizan por ejemplo los sistemas Linux y que permite modificar el código, redistribuirlo; pero manteniendo la licencia.

Otra licencia muy utilizada de código abierto es la que utiliza el sistema operativo **BSD**, que incluso permite redistribuir el software cerrando el código.

Los SGBD más conocidos de código abierto son:

- **MySQL**. Inicialmente creada por la empresa MySQL AB, posteriormente comprada por Sun Microsystems que, a su vez, fue comprada por Oracle. Ha sido considerada como la principal SGBD de la comunidad de programadores de código abierto y de hecho en Internet sigue siendo la principal base de datos asociada a una aplicación web. Mantiene su licencia de tipo GPL, pero posee una segunda licencia cerrada para opciones de trabajo más avanzadas.

Es muy popular por su histórica asociación con PHP, por su buena estabilidad, gran escalabilidad, e incluso uso de transacciones y lenguaje procedimental; además de ser un producto con infinitud de plataformas posibles para su instalación.

- **PostgreSQL**. Versión de código abierto basada en el producto **Ingres** de la **Universidad de Berkeley**. Usa licencia de tipo **MIT** (del **Instituto Tecnológico de Massachussets**) que es una de las más libres, permite su modificación, redistribución incluso hacia otro tipo de licencias del tipo que sean, sin usar en ningún momento copyright.

Está considerado como el SGBD de código abierto más potente y, sobre todo, más fidedigno con los estándares. Posee uso de transacciones avanzadas, lenguaje procedimental, gran estabilidad y escalabilidad.

Hoy en día está considerada como la más potente de las bases de datos de código abierto y a partir de su núcleo se han creado otros productos libres de software base de datos.

- **Firebird**. Se trata de un SGBD liberado del producto comercial **Interbase** que era propiedad de **Borland** y que fue liberado con una variante de la licencia que usa Mozilla (**MPL**) que, a su vez, se basa en la licencia BSD. Tiene soporte transaccional avanzado, buena estabilidad pero no es muy escalable.
- **Apache Derby**. Anteriormente, IBM **Cloudscape**, usa licencia Apache que es poco restrictiva con las redistribuciones. Está programada en Java y pensada para ser utilizada en ese mismo lenguaje.

### (1.3.4) bases de datos NoSQL

#### introducción

Las bases de datos relacionales han sido el modelo más popular desde finales de los años 70 por su solidez y gran facilidad para diseñar sistemas complejos. Sin embargo en estos últimos años empiezan a estar desbordadas ante el uso de bases de datos que tienen que dar servicio veloz y concurrente a miles de usuarios que son capaces degenerar ingentes cantidades de información en poco tiempo.

Esta información en una base de datos habría que validarla con las reglas e integridad que se imponen en esas bases de datos, indexarla y asegurar su uso en transacciones... y todo eso significa que un sistema con miles de entradas por minuto (como ocurre con las redes sociales) se bloquearía. Por ello se han diseñado bases de datos que se saltan el modelo relacional y en especial el lenguaje SQL y de ahí el nombre de sistemas **NoSQL**.

Aunque se utiliza para designar a las bases de datos documentales, gráficas y otros esquemas de bases de datos; actualmente se utiliza especialmente para designar a las bases de datos que requieren tantas transacciones por segundo, que el esquema relacional tradicional no daría abasto para ello.

La base teórica de este modelo se basa en el **teorema de CAP**, que indica que en un sistema distribuido no se pueden asegurar simultáneamente estas tres reglas:

- **Consistencia (C)**. Que hace que la información sea la misma en todos los nodos que almacenan los datos.
- **Disponibilidad (A de Availability)** que hace que cada petición sobre los datos reciba la confirmación de si ha sido satisfactoria o no.
- **Tolerancia a fallos (P de Partition tolerance)**, que permite que el sistema siga funcionando aun cuando ocurran errores o caídas en algunos nodos.

Un sistema relacional de base de datos podría asegurar la C y la P, pero no la disponibilidad en caso de una gran demanda de peticiones. Las bases de datos NoSQL varían las dos reglas a cumplir para dar más importancia a la disponibilidad (en general eliminan la primera la regla).

#### diferencias con bases de datos SQL

Por esto utilizan un modelo diferente en el que los datos se almacenan de forma menos estricta, en especial no siguen estas reglas:

- **TransaccionesACID**, como sí hacen los SGBD potentes relacionales (como **Oracle, DB2, SQLServer** o **PostgreSQL**). ACID hace referencia a las propiedades atomicidad, consistencia, aislamiento y durabilidad de las bases de datos. Y permite que las operaciones de manipulación sobre la base de datos sean revocables y así poder arreglar problemas de consistencia. Cumplir esas propiedades de forma estricta no permite que sitios como web puedan atender con garantías a todas sus peticiones por el tiempo necesario para actualizar los índices, por lo que necesitan otro esquema más rápido (aunque sea menos consistente).
- **No usar operaciones de tipo JOIN**. Los datos se almacenan sin validar su relación con otras tablas para hacerlo de forma más rápida.

- **No usan SQL como lenguaje de consulta.** En su lugar utilizan lenguajes de programación como **Java** o **C++** para acceder a los datos y otros como **XML** o **JSON** para definir los metadatos.
- **Datos no relacionales.** No se da importancia a las relaciones avanzadas con los datos. Por lo que no sirven para almacenar bases de datos complejas, sino más bien simples, con escasa relación y reglas. Por ejemplo los tweets de twitter, estadísticas a tiempo real, streaming de vídeo, conexiones a servidores remotos,... Es decir datos simples (de los que apenas se usan muy pocos valores clave) pero que llegan a gran velocidad.

La clave de este tipo de bases de datos es su capacidad para ser sistemas muy distribuidos y ultrarrápidos al almacenar datos: asegurando en todo momento su altísima disponibilidad y tolerancia a fallos.

Su desventaja es la ya comentada: al no poder establecer relaciones y reglas complejas sobre los datos, no sirven para almacenar bases de datos complejas o donde se desee realizar tareas como ordenar los datos en base a distintas claves o realizar búsquedas avanzadas en base a múltiples criterios. Triunfan donde ese tipo de tareas de gestión no son críticos (como las redes sociales, donde las cosas sólo se suelen ordenar en base a la fecha y hora).

### tipos de bases de datos NoSQL

Se consideran dentro de esta clasificación a estos tipos de bases de datos:

- **Clave/valor.** La forma de relacionar los datos se basa en estructuras indexadas en base a una clave a la que se asocia un valor (que puede estar compuesto de varios valores simples) usan estructuras de los lenguajes de programación conocidas como **tablas de símbolos** o **mapas**. Normalmente son el tipo de base de datos asociadas a las NoSQL. Las más conocidas son:
  - **Amazon Dynamo DB.** Desarrollada por la empresa Amazon, es un proyecto propiedad de esta empresa implementada para gestionar la enorme cantidad de transacciones de esta empresa y como una de las bases de su negocio en la nube. Almacena los datos en estructuras clave, valor: los valores son elementos binarios y, por lo tanto, opacos salvo desde la propia base de datos.
  - **Google BigTable.** Base de datos propietaria utilizada para muchos de los servicios de almacenamiento de Google. Los datos se almacenan en una estructura multidimensional de tres claves (fila, columna y fecha) y permiten ser particionados. El modelo físico se basa en el **Google File System**, modelo de archivos propietario de Google.
  - **Apache Cassandra.** Con licencia Apache, es la más popular y de contrastada potencia ya que es de código abierto. Se desarrolló inicialmente por **facebook**. Se dice de ella que intenta unir lo mejor de las dos anteriores, pero los datos se almacenan en tuplas sin relaciones de integridad (una variante de datos clave-valor, donde hay varios valores por cada clave). Presume, además, de un crecimiento exponencial en estos últimos tiempos y de ser el motor de base de datos de servicios como **twitter** o **netflix**.
  - **Azure Tables.** Uno de los sistemas de almacenamiento de la plataforma **Windows Azure** de **Microsoft**, creada para implementar su computación en la nube.

- **Berkeley DB.** Utilizadísima para manipular este tipo de bases de datos debido a su facilidad para comunicarse con todo tipo de lenguajes. Ahora pertenece a una empresa que respeta su licencia de código abierto, pero permite cerrarle en caso comercial.

■ **Tabulares.** Usan tuplas para almacenar la información rápidamente. No se agrupan en tablas, pero sí se asocian con una o varias claves. Es difícil establecer de forma clara una diferencia con las anteriores. Las más conocidas:

- **BigTable.** Comentada en el punto anterior se la considera de los dos tipos, pero es el SGBD que ha inspirado a todo este grupo de bases de datos.
- **HBase/Hadoop.** Creada por Apache para competir con la anterior.

■ **Almacenes documentales.** En las que la información importante utiliza un formato documental; es decir, un formato que encapsule la información y su formato. Por ejemplo **XML**, **JSON** o incluso formatos binarios como **PDF** o los formatos de **Microsoft Office**.

Los documentos se asocian a un valor clave (**key**) que permite indexarlos. Algunas bases de datos comerciales son:

- **MongoDB.** La más popular quizá, con licencia **GNU** y formato de datos parecido a JSON. Periódicos como **New York Times** o **The Guardian**; e incluso servicios como **foursquare** y otros, la utilizan. Siendo el software de este tipo quizá más popular.
- **Apache CouchDB.** Dentro de la familia Apache (y usando licencia de código abierto Apache) es la base de datos NoSQL orientada a documentos. Usa formato JSON para los datos y las consultas se realizan mediante **JavaScript**. Es utilizado por **Ubuntu**, la **BBC**, **Meebo**,...
- **Amazon SimpleDB.** Utilizada para el almacenamiento extenso de datos en la nube proporcionada de forma comercial por Amazon (es parte de **AWS**, **Amazon Web Services**).

■ **Bases de datos nativas XML.** El formato documental XML se utiliza en casi cualquier base de datos comercial actual. Se habla de **nativas** cuando el sistema utiliza en todo momento XML para almacenar y gestionar los datos. EN ellas la información se almacena en forma documental utilizando XML y se maneja con los lenguajes relacionados con XML como consultas **XQuery**, **XPath**, formato con **XSL**,... La gestión avanzada se realiza con conectores que permiten manejar el XML desde lenguajes como Java. Ejemplos:

- **Mark Logic Server.** Con licencia comercial.
- **BaseX.** Utilizada por la comunidad **GitHub** (que es un servicio en la nube utilizado por programadores), de libre uso.
- **eXist.** Con licencia GNU y con todas las capacidades XML.

■ **Basados en grafos.** Utilizan una estructura de los lenguajes de programación conocida como grafo, que permite relacionar los datos a través de enlaces que facilitan el recorrido por los mismos:

- **Neo4j.**
- **Infinite Graph**
- **AllegroGraph.**

■ **Orientadas a objetos.** Basadas en los principios de la **programación orientada a objetos**, en la que junto a los datos se indican las operaciones que se pueden

realizar entre ellos. Aunque se las suele agrupar en este tipo de bases de datos, lo cierto es que no persiguen los mismos objetivos que las anteriores.

## (1.4) arquitectura de un SGBD

### (1.4.1) ¿qué es la arquitectura?

La arquitectura de un SGBD hace referencia al modelo interno de funcionamiento del sistema. Es decir a las estructuras internas/físicas que proporciona para el almacenamiento y su relación con las estructuras lógicas/conceptuales. Como es lógico, cada SGBD propone diferentes arquitecturas.

### (1.4.2) estructuras lógicas de la base de datos

En todas las bases de datos relacionales disponemos de estas estructuras lógicas para organizar la información:

- **Tablas.** Compuestas de filas y columnas en las que se almacenan los datos relevantes de cada base de datos. La mayoría de SGBD usan distintos tipos de tablas, pero en general cuando se habla de tablas se habla del elemento lógico encargado de almacenar los datos.
- **Restricciones.** Se definen al crear las tablas, pero se almacenan aparte. Están disponibles en el diccionario de datos y marcan las reglas que han de cumplir los datos para que se consideren válidos.
- **Índices.** Se trata de una lista ordenada de claves que permite acceder a los valores de una o más columnas de una tabla de forma veloz.
- **Vistas.** Son consultas almacenadas que nos permiten mostrar de forma personalizada los datos de una o varias tablas.
- **Procedimientos y funciones.** Código del lenguaje procedimental de la base de datos utilizado para ejecutar acciones sobre las tablas (incluidos los triggers).

### (1.4.3) estructuras físicas e internas de la base de datos

Al final todos los elementos lógicos se deben almacenar en archivos cuyo tamaño, dirección,... etc. debe de ser controlado por el DBA. En los distintos tipos de SGBD hay variaciones sobre las estructuras lógicas, en el caso de las físicas su diferencia puede ser total, lo que obliga a conocer muy bien la parte interna del sistema concreto que estemos utilizando.

Las estructuras internas permiten analizar un nivel intermedio entre estructuras lógicas (como las tablas) y las físicas (como los archivos). Por ejemplo, Oracle proporciona espacios de tabla o tablespaces para aglutinar distintos elementos lógicos con distintos elementos físicos a fin de optimizar el rendimiento de la base de datos.

### (1.4.4) instancias de bases de datos

Los usuarios que deseen conectarse a una base de datos, se conectan a lo que se conoce como la instancia de la base de datos (del inglés **instance**).

En el modo más sencillo de trabajo, el usuario dispone de un software en su máquina local, por lo que se encuentra en el lado del cliente, capaz de conectar con el SGBD. En ese momento se lanza un **proceso de usuario**. Ese proceso deberá comunicarse (a



través de las redes apropiadas) con el **proceso de servidor**, un programa lanzado en el lado del servidor que está permanentemente en ejecución.

El proceso de servidor comunica a su vez con la **instancia de la base de datos**, otro proceso en ejecución a través del cual se accede a la base de datos.



Ilustración 1, Proceso de trabajo con la instancia de una base de datos

En el caso de bases de datos distribuidas, habrá varias instancias de base de datos con capacidad de atender concurrentemente más usuarios.

## (1.5) arquitectura de Oracle

### (1.5.1) ¿qué es un servidor Oracle?

Un servidor Oracle se considera al conjunto formado por la base de datos y la instancia de la misma. Entender la comunicación de estos dos elementos, es conocer el funcionamiento del servidor.

## (1.5.2) arquitectura en memoria del servidor Oracle

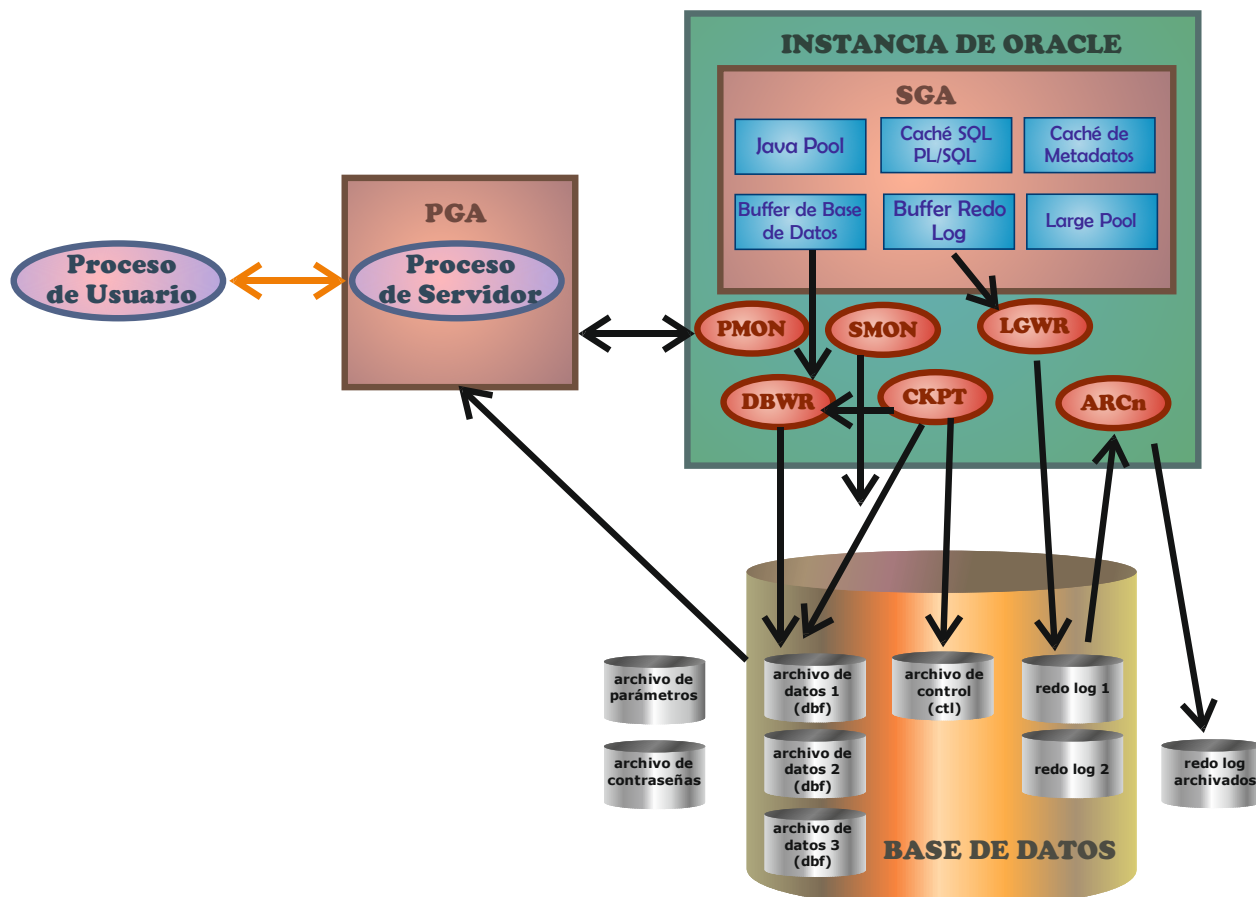


Ilustración 2, Arquitectura interna de Oracle

### instancia de Oracle

Es el conjunto de procesos del servidor que permiten el acceso a la base de datos. Es un conjunto de estructuras de datos y procesos en memoria. Está formado por:

- SGA. Area global de sistema.** Se trata de la zona de memoria común para todos los procesos de servidor, contienen las siguientes estructuras de datos fundamentales:
  - Buffer de caché de base de datos.** Almacena bloques de datos leídos de la base de datos a fin de que las próximas consultas no necesiten acudir a disco y se las pueda servir de estos datos en la caché.
  - Buffer redo log.** Estructura que almacena los datos anteriores y posteriores a cada instrucción y así facilitar tanto su anulación, como su realización en caso de problemas.
  - Large pool.** Área de la memoria que proporciona espacio para los datos necesarios para realizar operaciones de backup y restauración, así como los datos de sesión y otros que permitan aliviar el trabajo de la instancia.
  - Shared pool.** Consta de la caché del diccionario de datos y de la caché de instrucciones **SQL, PL/SQL**. De esa forma se acelera la ejecución de consultas e instrucciones que utilicen los mismos metadatos o bien que se traten de instrucciones parecidas a otras anteriormente ejecutadas.

- *Cache Library.*
- *Data Dictionary Cache.*
- *Java Pool.* Sólo se usa si hemos instalado Java para agilizar el proceso de las instrucciones en ese lenguaje.
- **Procesos en segundo plano.** Programas en ejecución que realizan las tareas fundamentales sobre la base de datos, entre ellos:
  - **DBWR.** Escribe los datos del buffer de cache de la base de datos de la SGA a la base de datos en disco (a los archivos de datos). Eso no ocurre en todo momento, sino cuando se produce un evento de tipo **checkpoint**.  
Un checkpoint ocurre cuando se ha consumido un tiempo determinado por el DBA, que se establece para que cada cierto tiempo los datos pasen a grabarse en ficheros de datos y así asegurarles en caso de problemas. El hecho de que esto se haga solo cada cierto tiempo (el tiempo establecido para el checkpoint) se debe a que, de otro modo, el funcionamiento sería muy lento si se accediera más a menudo al disco.
  - **LGWR.** Es el proceso que genera escrituras secuenciales en los **redo logs** (archivos *log de rehacer*) que son los archivos que guardan la información necesaria para poder recuperar un estado anterior en los datos.  
Las instrucciones DML están limitadas por la velocidad de este proceso al guardar los datos. LGWR escribe desde el buffer del caché redo en el SGA hacia los archivos redo en disco.
  - **CKPT.** Proceso encargado de comunicar la llegada de un *checkpoint*, punto de control que ocurre cíclicamente (y que se puede modificar por el DBA) tras el cual se deben de escribir los datos de memoria a los archivos de datos.
  - **SMON.** *System Monitor.* Proceso encargado de monitorizar el sistema para que funcione correctamente tras un error grave. Además se encarga de la optimización del sistema mejorando el espacio en disco y eliminando definitivamente (mediante *rollbacks*) datos irrecuperables.
  - **PMON.** *Process Monitor.* Se encarga de la comunicación con la PGA y especialmente con el proceso servidor para manejar la conexión con el cliente, eliminando transacciones de usuarios erróneas (por desconexión por ejemplo) y liberando la memoria que se reservó para los usuarios.
  - **ARCn.** Proceso de archivado de los archivos Redo. Sirve para que esos datos siempre estén disponibles. Sólo funciona en modo **ARCHIVELOG** de la base de datos, se explica más adelante.

## PGA

La *Program Global Area* o área global de programa, es la memoria que se reserva por cada usuario para almacenar los datos necesarios para la conexión de un usuario con la base de datos.

Cada conexión tiene su propia PGA con los datos a los que accede el **proceso servidor**. Entre los datos que almacena están:

- La información sobre la sesión con el cliente
- El estado de procesamiento de la instrucción SQL actual
- Datos de caché para acelerar algunas instrucciones SQL (como por ejemplo índices temporales)

### proceso servidor y proceso cliente

El proceso cliente es el programa en la memoria de la máquina en la que el usuario ha conectado con Oracle. Este proceso se comunica con un proceso servidor que es lanzado cuando el cliente establece conexión con Oracle.

Puede haber un mismo proceso servidor para más de un cliente en caso de una configuración compartida de proceso servidor. Cuando el proceso cliente y el servidor establecen conexión, se crea la sesión de usuario, que es manejada por el proceso servidor. El proceso de usuario no puede acceder directamente a la base de datos.

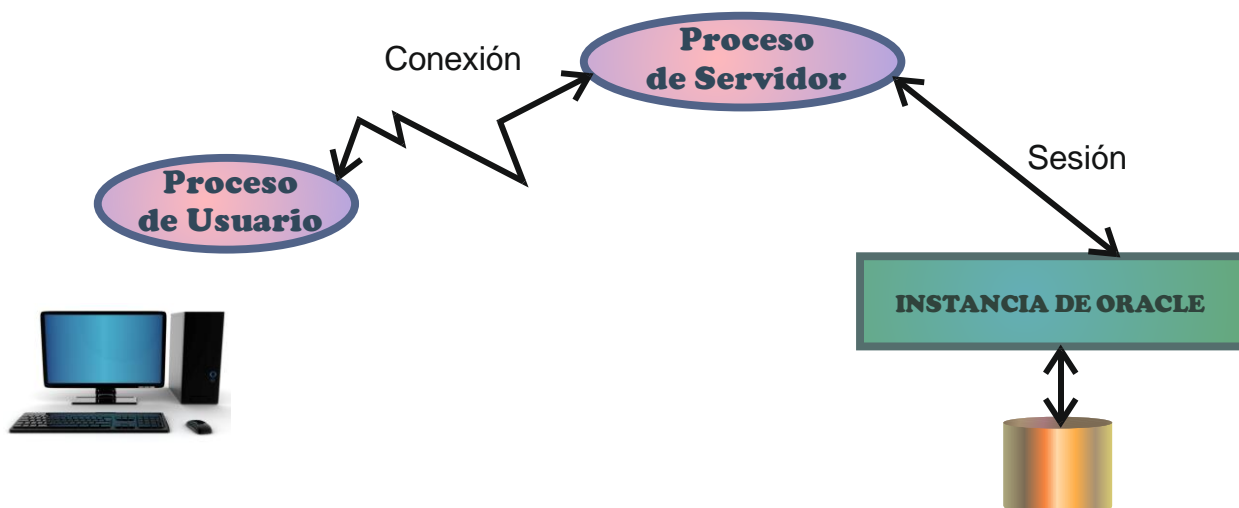


Ilustración 3, Funcionamiento del proceso servidor y de usuario

## (1.5.3) arquitectura en disco de Oracle

### estructuras lógicas de almacenamiento de Oracle

#### tablespaces

Los **espacios de tabla** o tablespaces, son una organización lógica interna de datos de Oracle que puede aglutinar varios archivos de datos.

Los datos que un usuario de la base de datos ve en tablas, al final proceden de algún archivo de datos en disco. Entre una óptica (la lógica de las tablas) y otra (la física de los archivos de datos), está el elemento que Oracle denomina tablespace (espacio para tablas).

#### tablespace 1



#### tablespace 2



Ilustración 4, Esquema básico de funcionamiento de los tablespaces de Oracle

Por defecto Oracle proporciona los espacios de tabla **USERS** (para los usuarios) **SYSTEM** (para los objetos del sistema) y **SYSAUX** (apoyo a SYSTEM); pero, lógicamente, podemos crear otros y asignarles los archivos de datos que deseemos.

### segmentos

En cada archivo de datos existen segmentos, que están relacionados directamente con un objeto de la base de datos y que sería reconocible por un usuario de la base de datos. Hay cuatro tipos de segmentos: de datos, de índice, temporales y de anulación (éste último sólo disponible en el espacio de tabla **SYSTEM**).

El mismo segmento puede estar presente en más de un archivo de datos (como en la Ilustración 4 ocurre con el segmento que almacena la [tabla 3](#))

### extensiones

Los segmentos se dividen en extensiones, que es el espacio que crece de golpe en un archivo cuando el objeto que se almacena en el segmento. Una extensión no puede ocupar más de un archivo.

### bloques

Es el elemento de datos más pequeño distinguible por Oracle. Cada extensión se compone de una serie de bloques. EL tamaño de bloque se puede configurar por parte del DBA para optimizar el rendimiento. Cada bloque de datos de Oracle equivale a uno o más bloques de datos del Sistema Operativo. Es decir si en un disco concreto, el Sistema Operativo tiene un tamaño de bloque de 16KB, sólo podremos asignar en Oracle tamaños de bloque de 16, 32, 48, 64 etc. Kilobytes.

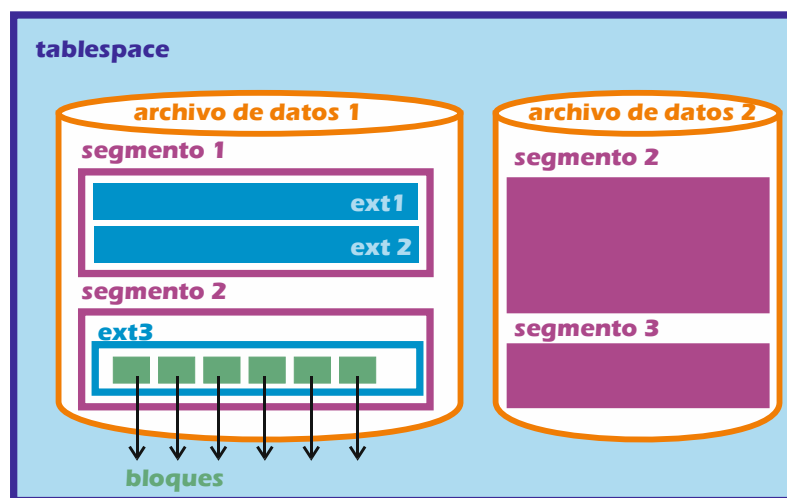


Ilustración 5, Arquitectura de los tablespaces



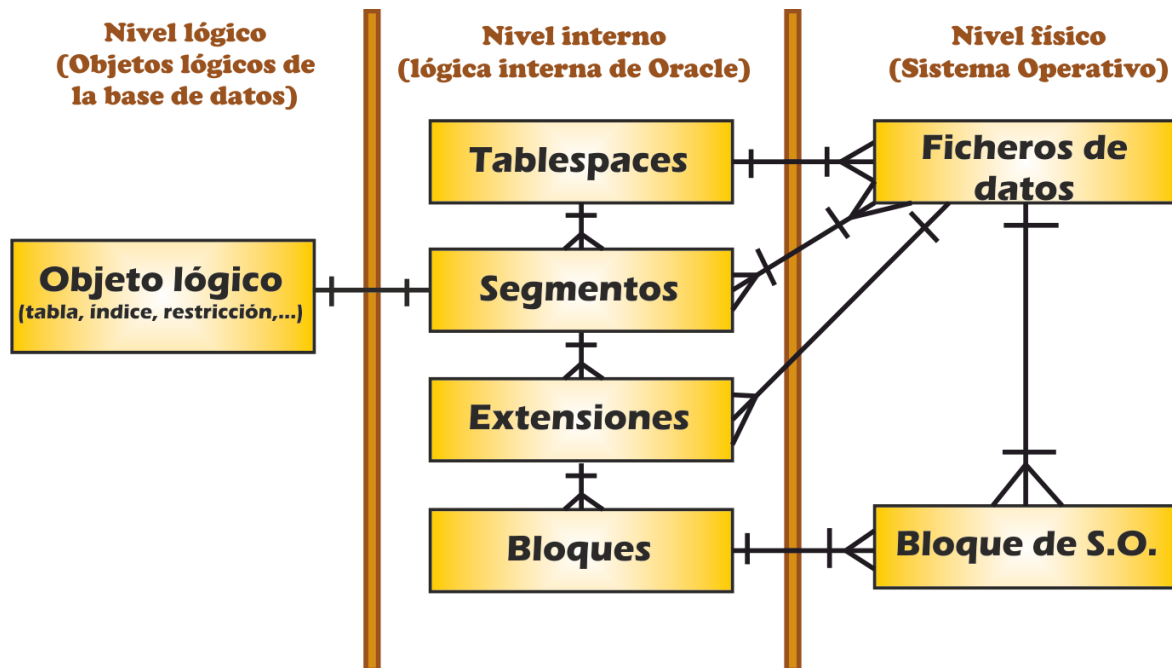


Ilustración 6, Correspondencia entre los distintos elementos físicos y lógicos de Oracle

## archivos de Oracle

### archivos de datos

Son los archivos que almacenan los datos en sí de la base de datos. Su estructura está comentada en la ilustración anterior.

### archivos del registro rehacer (redo log)

Los redo log son dos o más archivos que almacenan los cambios que se van registrando en la base de datos. Graban las instrucciones ejecutadas y los datos necesarios para volver a realizarlas. Son al menos dos, para asegurar que uno siempre recoge los cambios de la base de datos mientras el otro se almacena en disco.

Su funcionamiento es circular. Es decir se graba la información en uno y cuando se llena se pasa al siguiente. Tras llenar el último, comenzaremos de nuevo por el primero. Su número y funcionamiento se puede configurar

### archivos del registro rehacer archivados (archive redo log)

La base de datos puede funcionar en modo **ARCHIVELOG** o **NOARCHIVELOG**. La razón es que al serlos archivos redo log una estructura circular, podemos perder datos por ello en modo ARCHIVELOG la base de datos va copiando los archivo redo a medida que se llenan y así aseguramos que no perdemos datos. Estos archivos con copia de los redo son los archivos redo archivados o rehacer archivados.

### archivos de control

Se trata de archivos binarios y de tamaño pequeño que contienen la estructura de la base de datos, es decir los metadatos. Este archivo también se puede multiplexar, con lo que puede haber varios.

- Nombre de la base de datos
- Fecha y hora de la creación de la base de datos
- Información sobre **checkpoints** y **redo logs**.
- Modo de archivado de la base de datos.
- Número de secuencia del redo log actual.
- Metadatos para backup y recuperación de datos.

### archivos de parámetros

Comentados en el tema uno. Contienen los parámetros generales de configuración de la base de datos

### archivos de contraseñas

Contienen la lista de contraseñas cifradas, en caso de que ésta sea la forma de autentificar usuarios (según lo comentado en el tema uno).

### archivos de traza

Permiten establecer el seguimiento de los errores de procesos como PMON o SMON o para conexiones concretas.

### archivos de copia de seguridad

Imprescindibles para la recuperación de la base de datos en caso de desastre.

## (1.6) instalación de SGBD

A la hora de instalar el Sistema Gestor de la Base de Datos necesitamos primero hacer un planteamiento de necesidades a fin de encontrar el más apropiado para resolver el problema.

### (1.6.1) paso 1: selección por requisitos

Se trata de analizar qué necesitamos del SGBD. En este sentido algunas cosas a tener en cuenta pueden ser:

- **Tamaño de la base de datos.** Un gran tamaño de base de datos requiere se software muy potente para la gestión de la misma, además podría plantear el hecho de separar los datos en distintas unidades de disco o incluso máquinas lo que podría requerir clústeres o sistemas distribuidos.
- **Conectividad.** Si necesitamos que la base de datos sea accesible desde Internet, una Intranet o incluso si bastaría con un solo equipo de acceso.
- **Número de usuarios.** Un número grande de usuarios requiere controles avanzados de seguridad.
- **Número de conexiones simultáneas.** Suele ser el punto álgido de requisitos, ya que un gran número de conexiones simultáneas implica SGBD con grandes

capacidades de trabajo concurrente y pocos Sistemas serían capaces de aceptarlo.

- **Aprovechamiento de hardware.** Puede ser que sea el propio hardware de la empresa el que predetermine la selección al estar limitados por el mismo.
- **Política de empresa.** Por ejemplo si la empresa tiene una política de impulso de software libre o acuerdos con empresas concretas de software.

### (1.6.2) paso 2: comprobar requerimientos

Una vez seleccionado el SGBD ahora tenemos que asegurarnos de cumplir nosotros los requisitos que exige. Todos los sistemas indican qué requisitos necesitan en cuanto a:

- **Sistemas operativos.** No todos los SGBD son multiplataforma, lo normal es que sean compatibles con unas cuantas plataformas: Windows, Linux, Unix,...
- **Paquetes o aplicaciones preinstaladas.** A veces se requiere que el sistema posea algún software previo a la instalación del SGBD. En el mundo Linux se suele requerir de paquetes (como por ejemplo el compilador de C, o librerías especiales de entrada salida,...); en Windows es alguna actualización (como sus clásicos **Service Pack**) o software de terceros que se requiere (como la máquina **Java**, el **Framework .Net** o un servidor web concreto).
- **Memoria RAM.** Es el requisito que más importa: más RAM, más ligero funciona el sistema. Oracle en su versión 11g aconseja al menos 1 GB de RAM
- **Procesador.** Se suele exigir un modelo y una velocidad mínima en el mismo.
- **Disco duro.** Se exige un espacio mínimo de disco.
- **Requisitos de red.** Se puede exigir que el equipo tenga una función concreta como que sea un servidor de dominio, o que tenga una conectividad particular (como una dirección IP fija).
- **Incompatibilidades.** A veces se indican productos con los que existen problemas de compatibilidad.

## (1.7) instalación de Oracle

### (1.7.1) documentación

Oracle también dispone de manuales de libre descarga en la dirección: <http://download.oracle.com/docs/> Desgraciadamente toda la documentación sólo se encuentra en inglés.

### (1.7.2) directorios de Oracle. comprender la estructura OFA

El SGBD comercial Oracle sólo se puede instalar en sistemas Windows, Linux y Solaris. Normalmente se instala a través del software conocido como **Oracle Universal Installer (OUI)**, que al ser un programa Java, es el mismo en todas las plataformas.

Por otro lado Oracle ha recomendado un estándar en las instalaciones que conviene seguir y conocer tanto si instalamos nosotros Oracle como si son otros los instaladores, ya que si son instaladores profesionales seguirán esta estructura. Se trata del **Oracle's Optimal Flexible Architecture (OFA)**.

OFA impone la estructura de directorios y archivos en el sistema que Oracle debe utilizar. La razón es facilitar las tareas de administración, especialmente las de añadir bases de datos, instalar software y administrar usuarios.

Sus elementos fundamentales son las variables de sistema ORACLE\_BASE y ORACLE\_HOME. Cada versión de cada producto debería tener una versión diferente de estas variables que representan la ruta global a un directorio.

## Oracle Base

El directorio **Oracle Base**, es la raíz de las instalaciones de Oracle (de todos los productos, de todas las versiones). En el modelo OFA en Linux deben cumplir la forma:

```
/pm/h/u
```

Donde:

- **p**. Es el nombre de un texto estándar que suele ser corto y conciso. EL más usado es la letra **u**. Muchos instaladores usan la palabra **ora**.
- **m**. Es una expresión numérica que va de **01** a **09**.
- **h**. Es el nombre de una carpeta estándar. Se suele usar el nombre **app**.
- **u**. El nombre del usuario DBA de la instalación (por ejemplo **oracle**)

La ruta más habitual es:

```
/u01/app/oracle
```

En el caso de Windows la ruta sigue la expresión:

```
\oracle\app
```

y esta ruta debe estar en la raíz de cualquier unidad de disco. Por ejemplo **C:\oracle\app**

## Oracle Home

Se trata del directorio raíz de una instalación concreta de Oracle (Oracle 11g R2, Oracle 10g, etc.). Su ruta OFA es:

```
ORACLE_BASE/product/versión/nombre_instalación
```

La versión es la versión del producto que se instala y el nombre de la instalación es una cadena que ponen los instaladores de Oracle. Para los servidores Oracle se usa db seguida de un número, por ejemplo **1**. Ejemplo de ruta completa Oracle Base Linux podría ser:

```
/u01/app/oracle/11.2.1/db_1
```

En Windows:

```
c:\oracle\app\product\11.2.1\db_1
```

## ruta a los archivos de datos

La base de datos en el modelo OFA se almacena en un directorio llamado **oradata** que se debería encontrar dentro de la ruta a **Oracle Base**. Dentro de **oradata** se utiliza el nombre de instancia dada en la instalación para almacenar la base de datos. Por ejemplo (Linux):

```
/u01/app/oracle/oradata/orcl
```

## Oracle Inventory Directory

Este directorio se comparte con todas las instalaciones de Oracle. Sirve para indicar qué productos hay instalados. En cuanto se instala el primero producto Oracle, comprueba a ver si existe directorio de inventario; de no ser así, se crea. Si puede respetar la estructura recomendada OFA, en Linux se instalará en la raíz (por ejemplo `/u01/app/oralInventory`), de no ser así se instalará en el directorio del usuario que instala Oracle (`$HOME/oracle/oralInventory`).

En Windows se coloca siempre en la carpeta de Oracle dentro de la carpeta de aplicaciones de Windows (por ejemplo en `C:\Program files\Oracle\Inventory`).

Por otro lado Oracle dispone de un archivo que permite encontrar el inventario, que en Linux suele ser `/etc/oralnst.loc` y en Windows es la clave de registro: `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\inst_loc`

## Oracle Network Files Directory

Se trata del directorio que contiene los archivos de configuración de la red, en especial `tnsnames.ora` y `listener.ora` que contienen la configuración fundamental. Se suele encontrar en la ruta: `ORACLE_HOME/network/admin`

### (1.7.3) instalación en Windows

#### prerrequisitos

Oracle en Windows requiere un PC con al menos 1 GB de RAM y el doble en virtual. Disco duro con al menos 5,35 GB y tarjeta capaz de mostrar 1024 X 768 píxeles como mínimo.

#### instalación

Es aconsejable crear un usuario relacionado con Oracle con permisos administrativos y con él instalar el software. La razón: ser coherente con las rutas OFA comentadas anteriormente, que de otra forma harán referencia al usuario con el que instalemos Oracle sea o no relacionado con él.

En Windows, las variables de entorno y casi cada detalle se hace de manera automática sin intervenir previamente (a diferencia de la instalación Linux) se ocupa Oracle.

- (1) Descargar los archivos ZIP correspondientes a nuestra instalación y descomprimirlos **en la misma carpeta**. Desde esa carpeta (suele llamarse **database**) lanzar el instalador (archivo **setup.exe**).
- (2) Indicar correo electrónico al que Oracle enviará información sobre problemas críticos
- (3) Indicar si deseamos con el instalador instalar ya la instancia de base de datos. Normalmente habría que elegir instalar sólo la instancia (creando la base de datos después disponemos de más posibilidades).



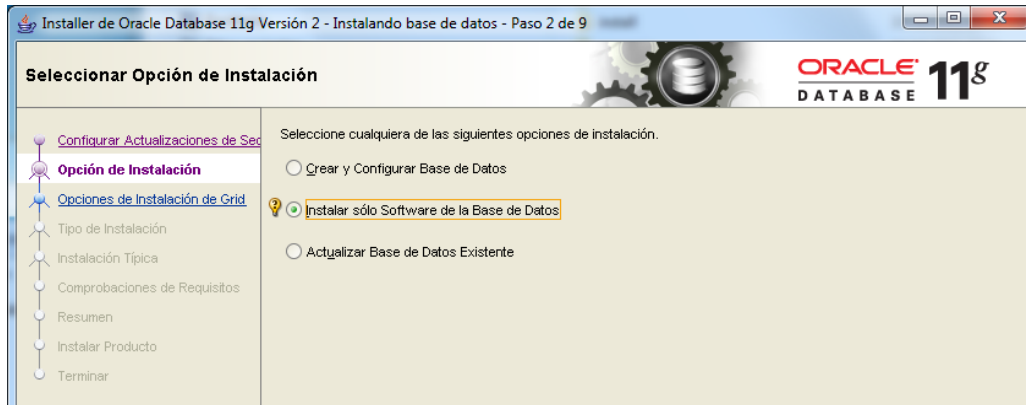


Ilustración 7, Selección de la opción de instalación de base de datos deseada

- (4) Indicar si deseamos instalar una instancia única de Oracle o bien instalar instancias múltiples (base de datos distribuida). Lo lógico, por ahora, es instalar una instancia única

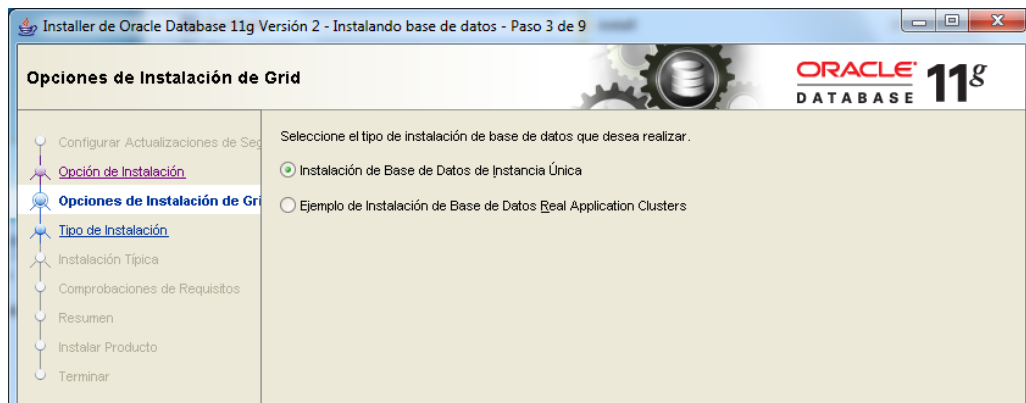


Ilustración 8, Selección del tipo de distribución de base de datos a instalar

- (5) Elegir los idiomas
- (6) Elegir el tipo de instalación. **Enterprise** es la opción si deseamos instalar todas las posibilidades de Oracle.

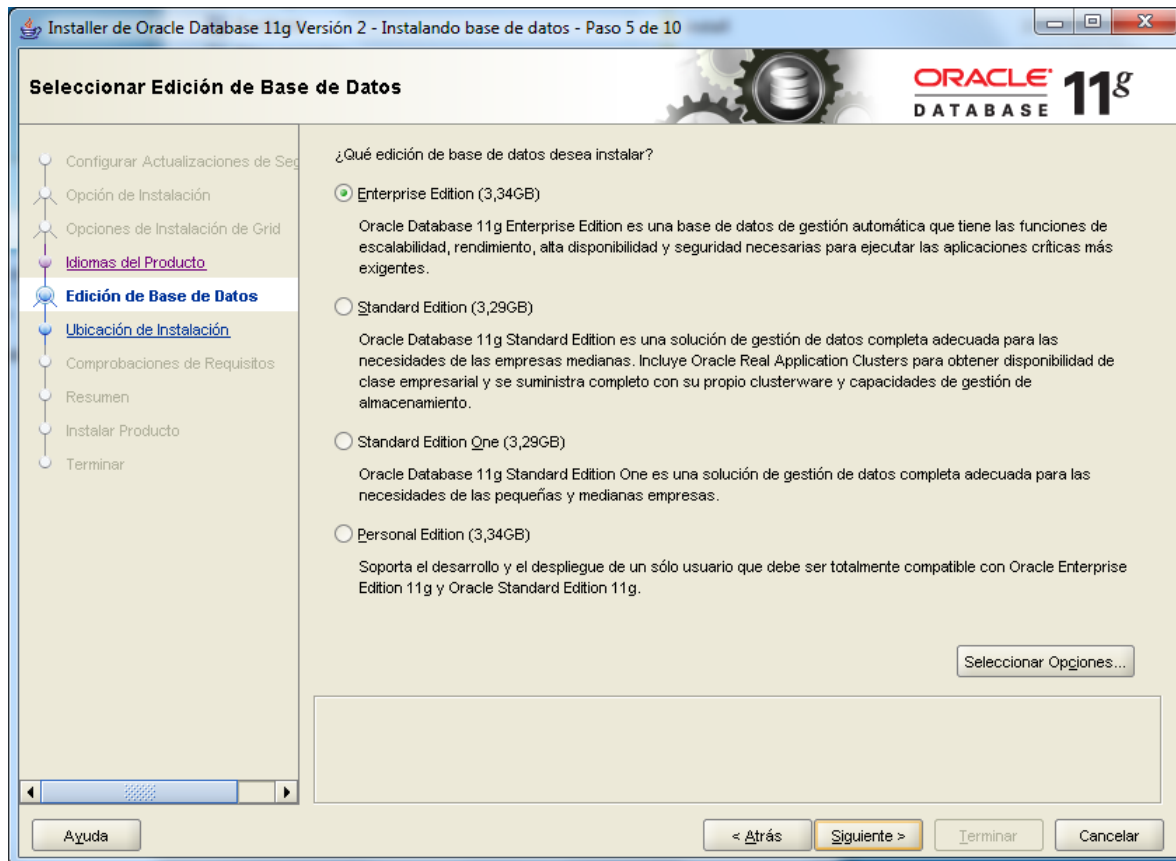


Ilustración 9, Edición de base de datos deseada

- (7) Elegir la ubicación de instalación. Normalmente el instalador la creará de forma coherente; se indicarán la base general de Oracle y el Oracle Home del producto que se está instalando en base al esquema OFA. En cualquier caso se puede cambiar la disposición según comprobemos lo que más nos interesa.

Si el usuario de instalación se llama **Oracle**, la instalación de Oracle Base (si es que no hay) será **C:\app\oracle**, el Oracle Home será algo del tipo: **C:\app\oracle\product\11.2.0\dbhome\_1**

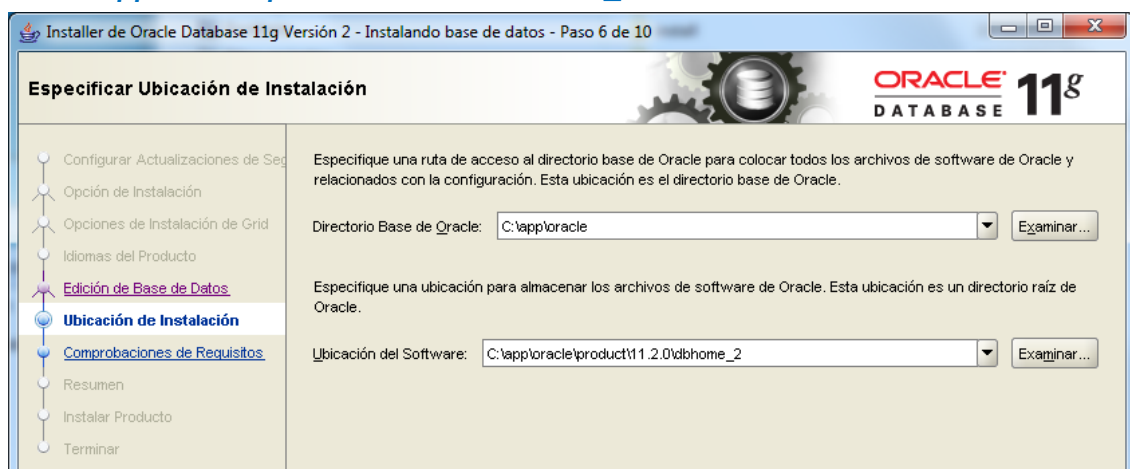


Ilustración 10, Ubicaciones del Oracle Base y el Oracle Home

- (8) Comprobación de requisitos. Se comprueba si el hardware y el software cumplen el mínimo de Oracle. De ser así se comienza la copia de archivos.

## variables de sistema

Conviene editar una serie de variables de sistema y configurarlas en Windows para su funcionamiento (por ejemplo desde Administrar Sistema de Windows) son:

- **ORACLE\_BASE**. Por ejemplo al valor **C:\app\oracle**
- **ORACLE\_HOME**. Por ejemplo a **%ORACLE\_BASE%\product\11.2.0\dbhome\_1**
- **PATH**. Añadir la ruta **%ORACLE\_HOME%\bin**

### (1.7.4) crear la base de datos

La base de datos no se tiene por qué crear durante la instalación. Es posible hacerlo después, tanto en Linux como en Windows.

Normalmente se utiliza el asistente **DBCA** (*Data Base Creator Assistant*) que en Windows se encuentra en **Inicio-Programas** en el grupo de Oracle en el apartado de **Herramientas de Configuración** bajo el nombre **Asistente de Configuración de Bases de Datos**. En Linux es un archivo en la carpeta **bin** de la raíz de Oracle (el **ORACLE\_HOME**) llamado **dbca**; para funcionar correctamente se deben configurar las variables de entorno, ejemplo (Linux):

```
export ORACLE_BASE=/u01/app/db11g
export ORACLE_HOME=$ORACLE_BASE/product/11.2.0/dbhome_1
export PATH=$ORACLE_HOME/bin:$PATH
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
```

En Windows habría que comprobar las variables de sistema en Propiedades del Equipo, en el apartado de opciones avanzadas. Es importante situar la carpeta **bin** de Oracle en el **Path** del sistema también en Windows).

En ambos casos hay que configurar la conexión de red a Oracle. Lo que se conoce como el **Listener**. Un listener es un programa residente en memoria (un proceso) que sirve para escuchar lo que llega por un puerto de red y enviarlo a su destino correspondiente. Sin él, por lo tanto, Oracle no obtendría las peticiones por red.

Los pasos completos para crear una base de datos son:

- (1) Lanzar el **Asistente de Configuración de Red** (comando **netca** desde la línea de comandos).
- (2) Elegir **Agregar un Listener**.
- (3) Elegir un nombre para él (lo habitual es **LISTENER**)
- (4) Especificar el protocolo por el que se comunica (**TCP** normalmente)
- (5) Elegir el número de puerto que utilizará (normalmente el **1521**)
- (6) Finalmente, salir del asistente (tras decir que no configuramos más listeners)

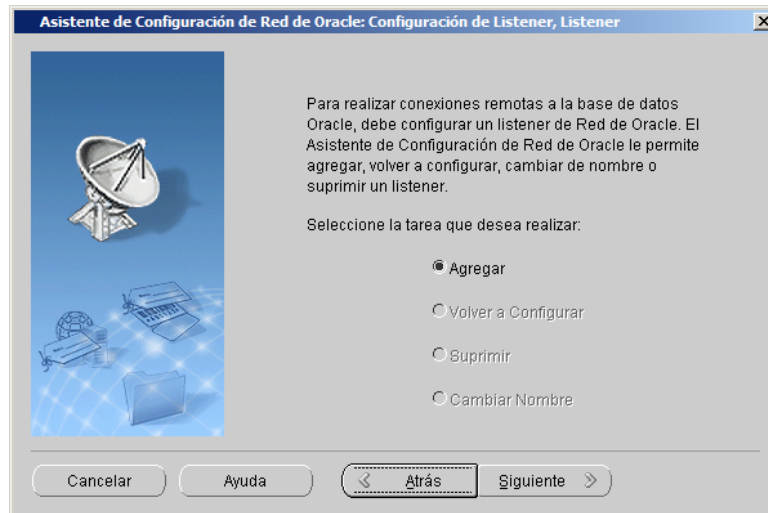


Ilustración 11, Asistente de conexión de red, a punto de crear el Listener

- (9) Lanzar el **Asistente de Configuración de Bases de Datos** (comando dbca)
- (10) **Paso 1.** Avanzar y elegir **Crear una base de datos**
- (11) **Paso 2.** Avanzar y elegir **Personalizar la base de datos**, de esa forma no se crearán datos en la misma y podremos configurar detalladamente las opciones de instalación.

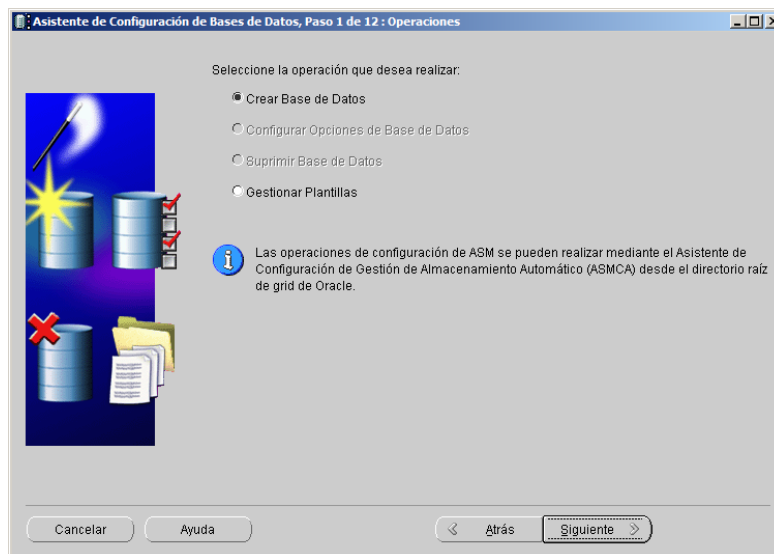


Ilustración 12, , EL asistente de configuración de bases de datos de Oracle

- (12) **Paso 3.** Indicar el nombre de la base de datos (toda nuestra red usará ese nombre) y un identificador de la base de datos (sid). Normalmente se pone el mismo nombre en ambos apartados. Este nombre será por el que nos referiremos en todo momento a la instancia de base de datos, es uno de los pasos fundamentales por lo tanto.
- (13) **Paso 4.** Asegurar que la casilla que permite configurar el servidor Oracle con **Enterprise Manager** está activada. El Enterprise Manager es un administrador visual de las bases de datos Oracle. Lo demás se suele dejar con sus valores por defecto.

- (14) **Paso 5.** Establecer las contraseñas para los usuarios administrativos (en especial para **SYS** y **SYSTEM**). Se puede elegir la misma para todos (válido solo para sistemas en prueba, por seguridad conviene crear contraseñas distintas). De esos usuarios hablaremos en el siguiente tema, pero SYS será el usuario con el que realizaremos las tareas administrativas fundamentales.
- (15) **Paso 6.** Paso en el que se elige la forma de almacenar los archivos. Se suele elegir un sistema de archivos normal como sistema de almacenamiento, usar las ubicaciones de la plantilla y se puede pulsar el botón **Variables de ubicación de archivos** para analizar los directorios creados y las variables del sistema. Las otras son opciones avanzadas de gestión de los archivos, que nos permite automatizar de otra forma dicha gestión.
- (16) **Paso 7.** La siguiente permite configurar la recuperación de **Flash**. Este modo de recuperación nos permite volver hacia atrás en los datos en caso de desastre. Esta utilidad consume mucho disco (Oracle recomienda al menos el doble del tamaño original de la base de datos), pero es enormemente interesante.
- (17) **Paso 8.** En la pantalla siguiente se suelen desactivar todas las casillas, excepto: **Enterprise Manager Repository**. El resto le añade funciones que no suelen ser necesarias en una base de datos estándar. Lo mismo ocurre con los componentes que aparecen al hacer clic en **Componentes de datos estándar**, conviene desactivar todos, a no ser que deseemos utilizar alguna de esas funciones, por otro lado muy interesantes aunque consumen recursos. Todo se podría instalar después en caso de necesidad.

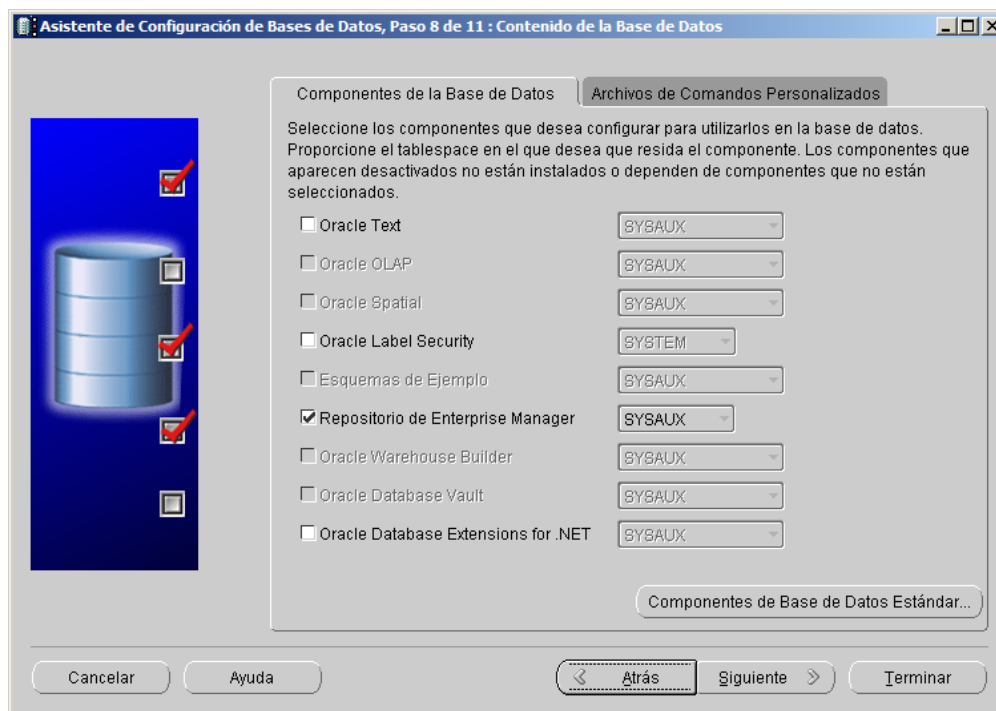


Ilustración 13, Conviene instalar sólo lo necesario

**(18) Paso 9.** La pantalla siguiente (cuatro pestañas) permiten modificar los parámetros de inicio de la base de datos. La mayoría son opciones avanzadas, hay que manipularlas sabiendo muy bien lo que se hace. Las pestañas son:

- **Memoria RAM.** Permite indicar la memoria que dejará nuestro servidor para almacenar los datos dinámicos configurando lo que se permite que ocupe tanto los datos globales (**SGA**) como los propios de cada proceso (**PGA**). Se puede configurar al detalle el gasto en memoria, incluso especificar la memoria que destinamos a cada espacio de almacenamiento (el Buffer de datos, el caché redo,...).
- **Bloque y conexiones simultáneas.** El tamaño del bloque no se puede modificar después. La razón para un tamaño u otro es la velocidad de la base de datos y el consumo que se hace en disco. Como se comentó anteriormente el tamaño del bloque debe de ser un múltiplo del tamaño de bloque mínimo del disco según el Sistema Operativo. El otro valor (normalmente 150), permite limitar o ampliar el número de usuarios simultáneos que pueden acceder a la base de datos, subir ese número, aumentaría los recursos gastados por Oracle. En el caso de una base de datos de prueba se puede bajar perfectamente ese valor.
- **Juegos de caracteres.** Permite elegir la forma de codificar el texto. En cualquier software actual se recomienda el uso de Unicode (urf-8); pero Oracle suele tomar por defecto la codificación del Sistema Operativo.
- **Modo servidor.** Se trata de la forma en la que trabajará el proceso servidor. Lo normal es elegir **dedicado** y así cada proceso cliente es atendido por un proceso servidor. Cuando tenemos enormes cantidades de conexiones clientes, en este modo consumiríamos todos los recursos y por eso se configura de modo **compartido**, en el que un mismo proceso servidor atendería a varios procesos clientes. Se configurar al detalle el modo compartido, aunque lo usual es usar un modo dedicado.

El resto de parámetros del SGBD están accesibles a través del botón **Todos los parámetros de inicialización** e incluso se pueden ver los más avanzados con un botón que lo indica.

**(19) Paso 10.** Ahora se puede gestionar la configuración de los archivos (de control, redo, archivados y de datos). Incluida la gestión de tablespaces, tamaños máximos de archivos, auto ampliación del tamaño,... Es muy interesante ver todas las posibilidades de esta pantalla (especialmente permite ver la ubicación de cada archivo).

**(20) Paso 11 y último del dbca.** Elegir **Crear la base de datos**, En todo caso deberíamos elegir crear script de creación de base de datos y así examinar cómo serían los comandos manuales para crear la base de datos (muy recomendable hacerlo)

**(21)** Finalmente se nos informa de la instalación. Es importante apuntar la página del Enterprise Manager para acceder a ella cuando lo necesitemos. Es más, lo lógico es guardar la configuración que se nos ofrece mediante el botón **Guardar HTML**.

**(22)** Una vez iniciada la generación de la base de datos, merece la pena estar atentos a los mensajes de Oracle a fin de entender el proceso y dónde se almacena cada elemento.



- (23) En la última pantalla (si todo ha ido bien), se presentan los datos de la instalación. Esos datos son fundamentales, entre ellos está la dirección del archivo de configuración de Oracle **ora.ini** y la URL para entrar en el servidor de aplicaciones que gestiona la base de datos.

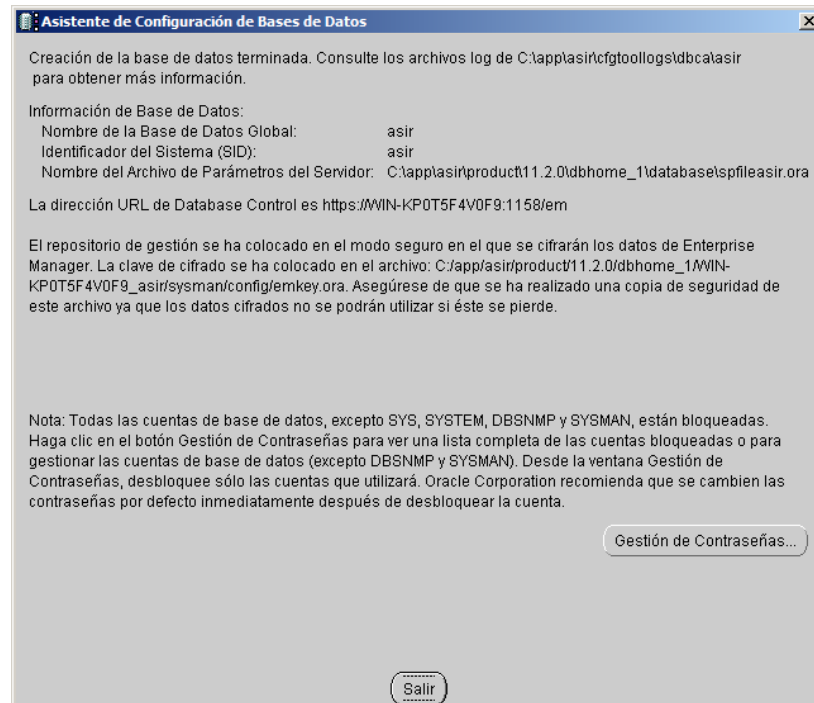
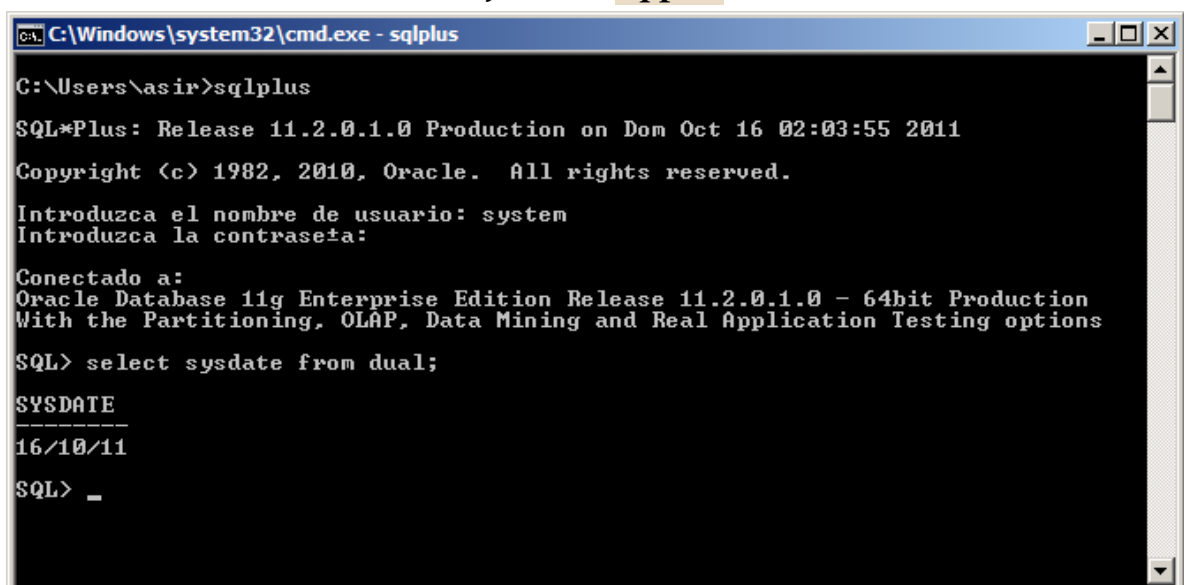


Ilustración 14, Pantalla final del asistente de configuración de bases de datos de Oracle

### (1.7.5) conectar con la base de datos

La forma clásica de conectar con Oracle, es utilizar la utilidad SQL\*Plus, que es un cliente básico que permite enviar comandos a Oracle. Para utilizarlo, bastaría con ir a la consola de comandos del sistema y escribir **sqlplus**:



```
C:\Windows\system32\cmd.exe - sqlplus

C:\Users\asir>sqlplus

SQL*Plus: Release 11.2.0.1.0 Production on Dom Oct 16 02:03:55 2011

Copyright (c) 1982, 2010, Oracle. All rights reserved.

Introduzca el nombre de usuario: system
Introduzca la contraseña:

Conectado a:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> select sysdate from dual;

SYSDATE
-----
16/10/11

SQL> _
```

### (1.7.6) control de la instancia de Oracle

En cuanto Oracle finaliza la instalación, coloca como servicios que arrancan con el sistema, los necesarios para el funcionamiento de la instancia de la base de datos. La forma habitual de gestionar el funcionamiento de cada instancia de Oracle es mediante el programa **Database Control**.

Cada instancia posee su propio Database Control, que en realidad es un programa llamado **emctl** que se encuentra en el directorio **bin** de cada **Oracle Home**. Desde la línea de comandos, se permiten estos (suponiendo que el directorio **bin** del **Oracle Home** está accesible en el PATH del sistema):

```
emctl start dbconsole
emctl stop dbconsole
emctl status dbconsole
```

Los dos primeros detienen o lanzan la consola. El tercero muestra información sobre **Database Control** (cómo acceder, estado actual, etc)

El acceso al entorno de Database Control, llamado **Enterprise Manager** se hace desde

```
http://nombreDeServidor:puerto/em
```

Para saber exactamente la localización, basta con usar el comando **emctl status dbconsole**. Eso permite acceder al control de la base de datos mediante el navegador de Internet. En la dirección **ORACLE\_HOME/install/portlist.ini** se pueden consultar los puertos que usa Oracle para el **Enterprise Manager**.

Desde este entorno se pueden realizar todas las operaciones de administración con la base de datos. La forma de conexión es entrar con SYS (con opción SYSDBA) y su contraseña.

En el caso que falle la carga del Database Control hay que asegurar que disponemos de la variable de sistema **ORACLE\_UNQNAME** conteniendo el nombre de la instancia (sid). En caso de que no sea así hay que definirla en las variables del sistema, de manera rápida en Windows sería:

```
set ORACLE_UNQNAME=nombreBD
```

Donde **nombreBD** es el nombre identificativo de la base de datos.

### (1.7.7) control del listener

Como ya se ha comentado; el programa capaz de escuchar por el puerto asignado a la base de datos, las peticiones a la misma, se conoce como **Listener**. Para gestionarle se usa el programa **lsnrctl**, disponible también en el directorio **bin** del Oracle Home. Sus posibilidades para lanzar o quitar el **listener** son:

```
lsnrctl start nombreListener
lsnrctl stop nombreListener
lsnrctl status
```

También se puede lanzar desde el Enterprise Manager, y como servicio en el caso del entorno Windows.

### (1.7.8) conexión en modo administrador

Si usamos la sintaxis:

```
sqlplus /nolog
```

Entonces entramos en sql\*plus sin indicar usuario, de otra forma el programa nos pediría usuario y contraseña para conectar. La opción **/nolog** permite muchas formas de conectar. La forma más avanzada de conectar es mediante CONNECT:

```
CONNECT usuario/contraseña [@alias] [AS {SYSOPER | SYSDBA} ]
```

Los usuarios normales no tienen capacidad de manejar la instancia de la base de datos. Sólo los que tienen roles de DBA o privilegios para cortar e iniciar la instancia. A este respecto hay dos privilegios especiales que permiten a un usuario (que tenga la posibilidad de acceder en modo administrador) operaciones avanzadas en la base de datos, son:

- **SYSOPER**. Tiene capacidad de arrancar (**STARTUP**) detener una instancia (**SHUTDOWN**), además de hacer modificaciones a las bases de datos y los *tablespace*.
- **SYSDBA**. Además de lo anterior se le permite crear bases de datos, así como usuarios con privilegios SYSDBA y SYSOPER.

Ejemplos de conexión:

```
CONNECT adolfo/natura; --conecta como usuario normal
CONNECT adolfo/natura@asir; --conecta como usuario normal indicando
                             --a qué base de datos se conecta
CONNECT adolfo/natura@asir AS SYSOPER; --ídem pero indicando que
                             --se conecta con rol de SYSOPER
CONNECT / AS SYSDBA
```

En la última (sólo para Windows), no se indica usuario. Se recogerá el usuario del sistema de Windows que hará posible la conexión, siempre que ese usuario sea propietario del grupo Oracle (si es el administrador, se toma como usuario **SYS**).

El comando para desconectar es **DISCONNECT**.

### (1.7.9) manejo de la instancia de la base de datos

#### arranque

Una base de datos Oracle puede estar en uno de estos cuatro estados:

- **SHUTDOWN**. La base de datos está cerrada, todos los archivos de la misma no están en ejecución.
- **NOMOUNT**. La instancia de base de datos está latente en memoria, con los procesos comunes funcionando. No es posible conectar con la base de datos.
- **MOUNT**. Al estado anterior se añade la lectura de los archivos de control para optimizar el funcionamiento de la instancia. Los usuarios no pueden conectar.
- **OPEN**. La base de datos está completamente lanzada y disponible a los usuarios.

El comando que acepta estos estados es **STARTUP**. Pero **ALTER DATABASE** seguida de una de esas palabras permite modificar el estado actual.

### parada

Una instancia cuando es arrancada, hasta estar disponible atraviesa todos los estados anteriores.

El comando de apagado de la instancia es **SHUTDOWN**, su sintaxis:

```
SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT]
```

Las opciones son:

- **NORMAL**. No se admiten más conexiones a la base de datos, pero las actuales se mantienen. Cuando se cierre la última, la base de datos también se cerrará.
- **TRANSACTIONAL**. Igual que la anterior, pero ahora se cortan todas las conexiones que no hayan empezado una transacción.
- **IMMEDIATE**. No se aceptan nuevas conexiones y se cortan las actuales. Las transacciones se cortan mediante una instrucción ROLLBACK, cuando se cancele la última, se apaga la instancia de base de datos
- **ABORT**. Apagado brusco. No se graba nada en disco. Simula un apagón.

### (1.7.10) borrado de la base de datos

Requiere de estos comandos

- (1) Necesitamos cortar la instancia actual:  
**SHUTDOWN IMMEDIATE;**
- (2) Iniciar la base de datos en modo MOUNT  
**STARTUP MOUNT EXCLUSIVE RESTRICT;**
- (3) Borrar  
**DROP DATABASE;**

Borrar una base de datos requiere de muchísima prudencia, ya que no se puede deshacer y podemos hacer desaparecer cantidades enormes de datos importantes.

### (1.7.11) Enterprise Manager

Prácticamente toda la administración de la base de datos se puede gestionar desde la aplicación Enterprise Manager que es un sitio web que normalmente es accesible en la dirección <https://localhost:1158/em>, aunque durante la instalación se nos indicará la dirección concreta en nuestra máquina para este servicio. El usuario SYS entrando en modo SYSDBA es el apropiado para administrar desde esta plataforma.

Lo malo es que falla a menuda y los arreglos no son siempre fáciles, pero es una forma muy visual de gestionar Oracle.

## (1.8) diccionario de datos en Oracle

En el caso de Oracle, no utiliza el esquema estándar **INFORMATION\_SCHEMA** para consultar el diccionario de datos, en su lugar usa una serie de vistas que comienzan con estos términos:

- **USER\_** Las vistas que comienzan por esta palabra muestran objetos que pertenecen al usuario/a que hace la consulta
- **ALL\_** Muestra objetos a los que el usuario tiene acceso (sean o no de su propiedad).
- **DBA\_** Muestra todos los objetos de la base de datos

Así **USER\_TABLES** es la vista que muestra todas las tablas del usuario actual. Otras vistas son (disponibles con el prefijo **USER\_**, **DBA\_** o **ALL\_**):

Vistas del INFORMATION_SCHEMA	Uso
<b>TABLES</b>	Muestra todas las tablas accesibles desde nuestro usuario.
<b>COLUMNS</b>	Muestra las columnas de las tablas.
<b>SYS_PRIVS</b>	Lista de privilegios de usuario
<b>TAB_PRIVS</b>	Privilegios de los objetos
<b>VIEWS</b>	Vistas
<b>TRIGGERS</b>	Lanzadores de código

Otras vistas interesantes para los administradores son:

Vistas del INFORMATION_SCHEMA	Uso
<b>V\$INSTANCE</b>	Muestra el estado de la instancia de la base de datos
<b>V\$SYSTEM_PARAMETER</b>	Parámetros del sistema
<b>V\$SESSION</b>	Sesiones activas
<b>ROLES_SYS_PRIVS</b>	Lista de roles y privilegios
<b>DBA_ROLES</b>	Roles de privilegios del sistema
<b>DBA_FREE_SPACE</b>	Espacio libre en los <i>tablespace</i>
<b>DBA_TABLESPACES</b>	Lista de <i>Tablespaces</i>
<b>DBA_TS_QUOTAS</b>	Gasto de los usuarios en los <i>tablespaces</i>

## (1.9) configuración de Oracle

Al igual que MySQL, los comandos de Oracle tienen parámetros posibles para enviar. Normalmente los parámetros se configuran en archivos especiales que son leídos por la instancia de Oracle antes de iniciarse, para así hacerlo con la configuración que indica el archivo (o archivos) de parámetros.

El archivo de parámetros puede ser:

- (1) Un archivo de texto (**PFILE**, acrónimo de *Parameters File*)
- (2) Un archivo binario (**SPFILE**), acrónimo de *Server Parameter File*.

La recomendación actual es utilizar un archivo binario, de hecho en la versión 11g estamos obligados a que así sea. La razón es que permite su modificación mediante el comando SQL: **ALTER SYSTEM** y porque en binario la configuración queda más oculta.

### (1.9.2) ubicación del archivo de parámetros

En Oracle 11g el archivo de parámetros **SPFile** (que es el que se usa por defecto), está en:

- **Linux/Unix.** En **ORACLE\_HOME/dbs/spfileSID.ora**, donde el SID es el identificador de la base de datos.
- **Windows.** En **ORACLE\_HOME/database/spfileSID.ora**, donde el SID es el identificador de la base de datos

En el caso de no disponer de SPFile, Oracle puede utilizar un archivo de texto para almacenar parámetros. Su ubicación sería:

- **Linux/Unix.** Está en **ORACLE\_HOME/dbs/initSID.ora**. Por ejemplo:  
/u01/app/oracle/11.2.1/db\_1/dbs/initbddd.ora
- **Windows.** Está en **ORACLE\_HOME\database\initSID.ora**

### (1.9.3) algunos parámetros

En estos archivos los parámetros (en los binarios no se ve) se marcan así:

```
parámetro=valor
```

El valor puede ser una lista de valores separados por comas. Los comentarios se ponen con el símbolo **#**. Algunos parámetros muy utilizados son:

- **db\_name=nombre.** Identificador de la base de datos
- **db\_domain=dominio.** Dominio al que pertenece la base de datos
- **control\_files=ruta.** Ruta a los archivos de control que se indique. Si no se indicara alguno, se crea en el directorio del archivo de parámetros
- **processes=número.** Máximo número de procesos que podrá lanzar la base de datos.
- **sessions=número.** Máximo número de sesiones concurrentes que se permiten.
- **socket=ruta.** Fichero o nombre de socket a usar en las conexiones locales.



- **log\_archive\_dest\_n**. Permite indicar el destino del archivos LOG de tipo REDO. *n* puede ser un valor entre 1 y 31 (números de archivo) y después se puyeden indicar un rosario de opciones para controlar exactamente lo que se graba.
- **db\_recovery\_file\_dest=ruta**. Permite indicar un directorio para la recuperación de la base de datos.

### (1.9.4) gestión de los archivos de parámetros

Para gestionar archivos SPFILE usa esta instrucción:

```
CREATE SPFILE [= 'rutaAlArchivoSPFILE']  
FROM [PFILE= 'rutaAlArchivoDetTexto' | MEMORY]
```

Con esa instrucción se crea un archivo SPFILE usando los parámetros en uso actualmente (opción MEMORY) o a partir de los contenidos de un archivo de parámetros de texto (opción PFILE). El archivo SPFILE por defecto (si no se indica ruta) estará en el mismo directorio que el de texto, solo que ahora en lugar de llamarse por ejemplo *initdb1.ora* sería *spfiledb1.ora*. El archivo SPFile por defecto es el que usa Oracle al iniciar la base de datos. Ejemplos de uso:

```
CREATE SPFILE FROM MEMORY; 'crea un archivo de parámetros  
binario usando los parámetros actualmente en uso'  
CREATE SPFILE 'spfile2.ora' FROM MEMORY; 'como el anterior, pero  
ahora le da nombre al archivo'  
CREATE SPFILE 'spfile2.ora' FROM PFILE='initasir.ora';  
'crea el archivo SPFile usando los parámetros y valores  
almacenados en el archivo de texto indicado'
```

Si en la instrucción anterior se indica destino para el SPFILE, entonces se tomará como copia de seguridad, ya que el que actúa es el archivo SPFILE por defecto.

También se puede crear un archivo de texto de parámetros a partir de un archivo SPFile (de hecho se usa mucho como copia de seguridad). Sintaxis:

```
CREATE PFILE [= 'rutaAlArchivoSPFILE']  
FROM [SPFILE= 'rutaAlArchivoDetTexto' | MEMORY]
```

Ejemplo

```
CREATE PFILE='copia.ora' FROM MEMORY;
```

crea un archivo de parámetros a partir de los parámetros en uso

### (1.9.5) modificación de parámetros en caliente

Los parámetros que se desea modificar, mientras se ejecuta la base de datos, se configuran con la instrucción **ALTER SYSTEM SET**; algunos son dinámicos (su efecto se produce inmediatamente) y otros estáticos (su efecto se produce tras el arranque). La sintaxis es:

```
ALTER SYSTEM SET parámetro=valor  
[COMMENTS = comentarios]  
[SCOPE={SPFILE | MEMORY | BOTH }]
```

**SCOPE** controla cuando se produce el efecto del parámetro:

- **SPFILE**. Significa que el comando modifica el archivo de parámetros SPFILE y sus efectos se verán en el siguiente arranque.
- **MEMORY**. El cambio se graba en memoria y se produce inmediatamente; pero como no toca el archivo SPFILE, su efecto no será inmediato.
- **BOTH**. Hace ambas cosas.

El comando **SHOW PARAMETERS** muestra los parámetros que actúan en la sesión actual; **SHOW SPPARAMETERS** muestra los del archivo SPFILE que sea el actual. La vista **V\$PARAMETER** contiene los parámetros actuales de la sesión, **V\$SYSTEM\_PARAMETER** contiene los parámetros del sistema y **V\$SPFILE** los del archivo SPFILE se usen o no.

## (1.10) ficheros LOG en Oracle

Hay múltiples archivos LOG para monitorizar los distintos aspectos de Oracle. Entre ellos:

- **alert LOG**. Registra sucesos de la gestión general de la BD.
- **LOG de procesos en background**. Registra los sucesos provocados por los procesos en ejecución de Oracle.
- **LOG de usuarios**. Monitoriza la actividad de los usuarios.

Se gestionan normalmente con el **Enterprise Manager**, pero es posible hacerlo con las vistas del diccionario de datos, como **V\$DIAG\_INFO** para el **alert log**. En temas posteriores se verán todos los archivos LOG y sus posibilidades.

## (1.11) APÉNDICE: instalación de MySQL

### (1.11.1) pasos previos

Antes de instalar **MySQL** (al igual que otro software cuya instalación sea crítica) se deben tomar estas decisiones:

- Comprobar hardware mínimo necesario
- Decidir la distribución. MySQL está disponible para numerosas plataformas, hay que decidir para cuál nos interesa, en base al precio del sistema, fiabilidad, buena integración, etc.
- Decidir el formato de la distribución. Hay dos posibilidades:
  - **Distribución binaria.** Se trata de una forma más fácil y rápida de instalar. Puede ser a través de un instalador preparado o bien a través de binarios genéricos comprimidos. En el último caso, simplemente con descomprimir y realizar unos cuantos ajustes, tenemos la instalación finalizada
  - **Código fuente.** Se trata de que debemos compilar el código fuente para que funcione el SGBD. Es más complicada de realizar pero permite un mayor control de todos los componentes a instalar, así como preparar un ejecutable óptimo para nuestro sistema o bien incluso modificar el código (que es C y C++).
- Obtener los archivos. De la dirección: <http://dev.mysql.com/downloads>.

### (1.11.2) documentación

MySQL dispone de manuales tanto en línea como descargables (por ejemplo en formato PDF) a través de la página <http://dev.mysql.com/doc/>

### (1.11.3) instalación en Windows

Es la versión más descargada de MySQL. Hay dos opciones:

- Distribución binaria, que permite iniciar el servidor inmediatamente. Es un archivo **msi**, con las ventajas que eso tiene en un entorno de trabajo Windows. Opciones:
  - Instalador **msi**
  - Instalación comprimida en formato ZIP
- Distribución de código fuente para ser compilada con el compilador **VC++ 6.0**.

## instalación con un paquete **msi**

### instalación con el asistente

El asistente de MySQL es el programa que permite facilitar la instalación del software. Los pasos a realizar con él son:

- (1) Aceptar el cuadro con la información inicial
- (2) Escoger el tipo de instalación. La opción personalizada (**Custom**) permite elegir más a fondo los componentes a instalar.
- (3) Confirmar la instalación

Tras finalizar estos pasos, automáticamente se modifica el registro de Windows, se añade un grupo para MySQL en el menú de Inicio y una carpeta (por ejemplo **MySQL 5.5**) dentro de la carpeta de los programas en el sistema de archivos de Windows (por ejemplo con la ruta **C:\Program Files\MySQL\MySQL 5.5**).

### instalar la instancia de MySQL

El sistema MySQL en sí funcionará en cuanto lancemos el asistente para gestionar la instancia. La instancia es el proceso de base de datos que permite el acceso a la base de datos. Eso puede hacerse en el último paso del asistente anterior o bien lanzando el programa en la carpeta **bin** del directorio de MySQL, **MySQLInstanceConfig**.

Una vez lanzado la primera pantalla es esta:

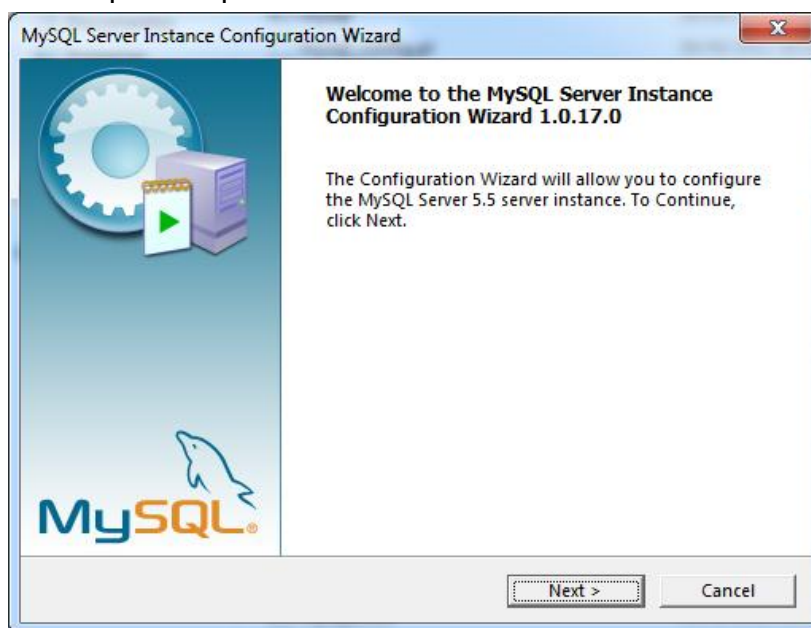


Ilustración 16, Imagen inicial del asistente de configuración de la instancia del servidor MySQL

Con él los pasos a realizar son:

- Elegir el tipo de instalación. La detallada permite más opciones de configuración.
- Elegir el tipo de servidor que deseamos:
  - **Developer Machine**. Máquina de Desarrollo. Para uso personal solamente, consumirá una cantidad mínima de recursos.

- **Server machine.** Servidor. Se entiende que comparte espacio con otros servidores (web, de correo, ftp). Consume una cantidad moderada de recursos.
  - **Dedicated Machine.** Servidor dedicado. Se elige si sólo se utiliza como servidor a MySQL.
- Uso de la base de datos. En realidad gestiona qué motores de la base de datos se usarán. MySQL dispone de dos opciones **InnoDB** (con buena capacidad de manejo de transacciones, pero más lenta en añadir datos) y **MyISAM**, más rápida pero con menos control de la integridad de los datos. Las opciones del cuadro son:
- **Multifuncional.** Instala ambos motores y les da la misma potencia.
  - **Transaccional.** Instala ambos, pero da preferencia a InnoDB en los recursos a fin de manejar correctamente las transacciones.
  - **No transaccional.** Desactiva InnoDB y activa sola MyISAM.
- Elección del directorio donde instalar los datos. Lo que se conoce como la ubicación del **tablespace** de datos. Sólo se usa con InnoDB y permite colocar los datos en otra ubicación respecto a los del programa MySQL o incluso en otra unidad de disco.
- Elección del número de conexiones concurrentes. La primera opción es para bases de datos con escaso número de conexiones concurrentes, la segunda pensada para transacciones en línea (OLTP) y la tercera permite editar a mano el número de conexiones concurrentes.
- Selección del número de puerto (normalmente el 3306) y elegir si deseamos el modo estricto de trabajo (es lo recomendable).
- Elección del juego de caracteres que se utilizará para codificar el texto de la base de datos.
- Elección del modo de trabajo del servicio. Normalmente MySQL se instala como un servicio de Windows más, pero podríamos no desearlo. Se puede cambiar el nombre al servicio.
- En este mismo paso se puede modificar el **PATH** de Windows para incluir el directorio **bin** en el **path** y así desde la línea de comandos utilizar los comandos mysql sin tener que modificar nuestra ruta.
- El siguiente paso es muy importante. Se DEBE de cambiar la contraseña del usuario **root**, del superadministrador, por seguridad. Además se puede elegir si permitimos los accesos de este usuario de forma remota (desde otro ordenador) y si activamos la cuenta anónima (acceso sin contraseña), que no es recomendable.
- Tras esos pasos comienza la instalación en sí. Si todo va bien aparece un mensaje como éste:

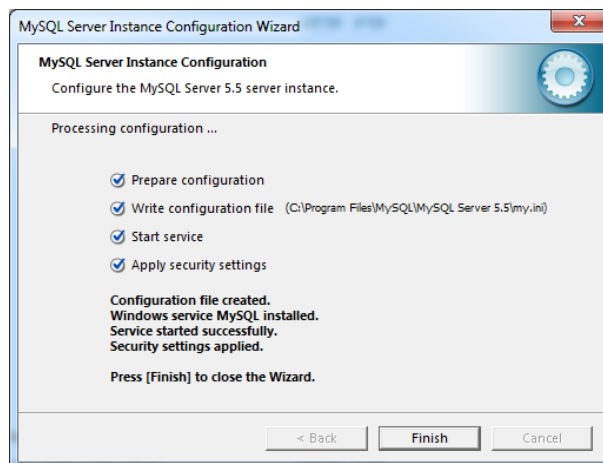


Ilustración 17, Cuadro de diálogo que aparece cuando la instalación ha concluido

En el cuadro último, se indica la ruta del archivo de configuración en el que se han guardado los parámetros de instalación. Es el archivo **my.ini** situado en la carpeta de instalación de MySQL. Examinar ese archivo es interesante y modificarle permitiría cambiar el funcionamiento de MySQL independientemente de lo que se haya especificado en la instalación.

### comprobar funcionamiento de MySQL

Necesitamos probar la instalación. En Windows, MySQL crea las tablas de privilegios y usuarios sin tener que configurar más. Además creará tres bases de datos y diversas tablas en ellas como ejemplo.

MySQL será un servicio que se iniciará automáticamente en el arranque. De no desearlo así debemos ir a la pantalla de **Servicios** de Windows (dentro de la carpeta de herramientas administrativas) y parar el servicio además de indicar que no deseamos su inicio automático (en las propiedades del servicio **MySQL**).

Para detener el servicio, basta ir a la línea de comandos e indicar el comando:

```
net stop MySQL
```

Si deseamos iniciar de nuevo el servicio:

```
net start MySQL
```

Si el servicio está funcionando, para probar el funcionamiento basta con conectar con MySQL desde la línea de comandos, por ejemplo con:

```
mysql -u root -p
```

Después se nos solicitará la contraseña del administrador. Y entonces estaremos viendo el Shell de MySQL (la pantalla que interpreta comandos MySQL). Por ejemplo ejecutando el comando **show databases**, se nos mostrarán las bases de datos ejemplo de MySQL.

En el caso de que el servidor no arranque con el comando net, podemos lanzar MySQL invocando al programa (demonio) **mysqld**

### desinstalar MySQL en el caso de instalación con el paquete msi

Para ello basta con ir al panel de control, localizar MySQL y desinstalar. Pero eso elimina sólo los archivos de MySQL que se crearon en la instalación, los archivos de



configuración y datos permanecerán donde se crearon y eso puede causar problemas en otra instalación. Para desinstalar completamente:

- (1) Desinstalar desde el panel de control
- (2) Eliminar el contenido de **C:\Documents and Settings\All Users\Application Data\MySQL**, o bien en Windows 7 o Windows Server 2008 **C:\ProgramData\MySQL** (**ProgramData** es una carpeta oculta que deberemos mostrar con la opción de mostrar carpetas y archivos ocultos de Windows)
- (3) Revisar la carpeta de instalación de MySQL y comprobar que no ha quedado ningún archivo sin borrar

### instalar MySQL desde un fichero ZIP

En ese caso tenemos un archivo comprimido que contiene la instalación de MySQL, simplemente habrá que configurar la instalación. Los pasos son:

- (1) Extraer el fichero ZIP en la ubicación deseada
- (2) Crear un fichero de opciones con al menos las líneas donde colocar la ubicación de MySQL y los datos. Aunque hay varias posibilidades de colocar el archivo de configuración (como se describirá más adelante), lo habitual es llamarle **my.ini** y colocarlo en la carpeta de instalación de Windows (por ejemplo **C:\Windows**).

Conviene copiar uno de los archivos de configuración que se encuentran en la carpeta raíz de MySQL (por ejemplo **my-medium.ini**), cambiarle el nombre a **my.ini** y modificar las líneas deseadas.

Ejemplo de ello es por ejemplo indicar cuál es el directorio raíz de MySQL y la carpeta donde se almacenará la base de datos. Para ello se añadirían las siguientes líneas a la sección **mysqld**:

**[mysqld]**

..... **otros parámetros de configuración**

*# coloca en basedir el directorio de instalación*

**basedir=E:/mysql**

*# coloca en datadir el directorio de datos*

**datadir=E:/mydata/data**

Como se ve en el ejemplo MySQL usa las barras de los directorios al estilo Unix (/ en lugar de \). Para que la ruta cuadre con el archivo de opciones es necesario asegurar que la raíz de los archivos de MySQL se encuentra donde se ha marcado en el archivo de opciones.

**Nota:** Normalmente el directorio de datos (**data**), se encuentra debajo de la raíz de instalación de MySQL, **si deseamos utilizar la nueva ubicación deberemos copiar el contenido de data a esa nueva ubicación.**

- (3) Colocar la ruta al directorio **bin** de MySQL dentro de la variable PATH de Windows.
- (4) Lanzar la instancia de la base de datos desde la línea de comandos escribiendo:  
**mysql --console.**

Tras este comando (si la configuración está bien), se quedará la consola con el mensaje: **Version: '5.5.16' socket: " port: 3306 MySQL Community Server**

(GPL) (suponiendo que disponemos de MySQL Community Server versión 5.5.16). El servidor MySQL está lanzado y escuchando por el puerto 3306

- (5) Para el resto de veces bastará con lanzar el demonio de MySQL (mysqld), simplemente escribiendo **mysqld** en la línea de comandos.

### arrancar MySQL como un servicio

También es posible en este tipo de instalación arrancar MySQL como un servicio. Basta con:

```
mysqld --install
```

Si no funcionara es posible que el servidor MySQL esté en funcionamiento. Detener implica usar el comando:

```
mysqladmin -u root shutdown
```

Desde este instante tendremos MySQL colocado como servicio de Windows. Disponemos de las opciones de comandos:

- **mysql --install-manual**. Para instalar MySQL como servicio que debe iniciarse manualmente.
- **mysql --remove**. Para eliminar MySQL como servicio.

### desinstalar si la instalación es desde un ZIP

- (1) Desinstalar el servicio de MySQL si es tal. En todo caso detener la instancia definitivamente de MySQL con **sc delete mysql**
- (2) Si lo anterior no funciona (por ejemplo sino se configuró como servicio), entonces: **mysqladmin -u root shutdown**
- (3) Borrar el archivo de configuración **my.ini**
- (4) Borrar los archivos de MySQL

En realidad en este tipo de instalación, todo es más sencillo porque no se instala MySQL al estilo de las aplicaciones de Windows

El usuario administrador en este tipo de instalación no posee contraseña; habría que cambiarla más tarde.

## (1.11.4) instalación en Linux/Unix

### instalación binaria genérica

La instalación binaria genérica en los sistemas de tipo Unix es similar a la de Windows mediante archivo ZIP. Este tipo de instalación valdría para cualquier versión de Linux. La ventaja es la comodidad y el hecho de que se maneje igual en todo tipo de Unix/Linux. La desventaja es que esta instalación no está optimizada para la versión de Linux concreta en la que instalamos MySQL.

La instalación creará una carpeta raíz desde la que colgarán todos los directorios de MySQL, concretamente:

directorio	uso
<b>bin</b>	Ubicación de los programas MySQL
<b>data</b>	Ubicación de los datos de las bases de datos

directorio	USO
<b>docs</b>	Manual en formato <b>info</b> de Linux
<b>man</b>	Manual en formato del comando <b>man</b> de Unix
<b>include</b>	Cabeceras de código fuente
<b>lib</b>	Archivos de librería
<b>scripts</b>	Contiene el script <b>mysql_install_db</b>
<b>share</b>	Mensajes de error, juegos de caracteres, ejemplos,...
<b>sql-bench</b>	<i>Benchmarks</i>

Los pasos son:

- (1) Descargar el archivo comprimido con la instalación desde <http://dev.mysql.com/downloads/> Será algo así como **Linux Generix xxxx.tar.gz TAR Compressed** donde las equis indican la versión del sistema (por ejemplo versión 2.6 de 32 o 64 bits)
- (2) Abrir la consola de comandos (el **Shell**) con privilegios administrativos y ejecutar la orden:  
**groupadd mysql.**  
Esto añade un grupo para el usuario relacionado con MySQL
- (3) Ejecutar la orden:  
**user add -g mysql mysql.**  
Esto crea un usuario llamado mysql que pertenece al grupo anterior.
- (4) Colocarnos en el directorio que contendrá la instalación de MySQL. Lo habitual es el comando: **cd /usr/local.**
- (5) Descomprimir el archivo con:  
**tar zxvf rutaCompletaAlArchivoTAR.tar.gz.**  
Necesitamos indicar la ruta al archivo **tar** para descomprimirlo en la carpeta en la que nos posicionamos en el punto 4. Es decir si hemos descargado MySQL en la carpeta de descargas, podría ser: **tar zxvf \$HOME/Downloads/MySQL-5.5.16-1.Linux.2.6.x86\_64.tar.gz.** El resultado es MySQL descomprimido en **/usr/local** (o en el directorio que hayamos indicado en el punto 4).
- (6) Modificar el nombre del directorio, para que sea más entendible. Por ejemplo:  
**mv /usr/local/mysql-5.5.16-linux.2.6-x86\_64 /usr/local/mysql**
- (7) Ir al directorio raíz de **MySQL** (con **cd /usr/local/mysql** usando el nombre creado en el punto anterior) desde ahí dar propiedad de los archivos de MySQL al usuario root y grupo **mysql** creados anteriormente. Al usuario **mysql** se le da la propiedad de la carpeta data donde se encuentran los datos:  
**chown -R root .**  
**chown mysql data**  
**chgrp -R mysql .**

- (8) Para el paso siguiente necesitamos la librería **libaio1**, si no disponemos de ella hay que instalarla. En los Linux tipo Debian (como Ubuntu) con:

```
apt-get install libaio1 libaio-dev
```

En los Linux tipo Red Hat (como Fedora) con:

```
yum install libaio1
```

- (9) Instalar las tablas iniciales mediante (suponiendo que seguimos en la raíz de MySQL):

```
scripts/mysql_install_db --user=mysql
```

- (10) Iniciar MySQL, pero teniendo cuidado de hacerlo sin ser el usuario **root**. La forma de hacerlo (situados desde la raíz de MySQL):

```
bin/mysqld_safe --user=mysql &
```

Si sale bien el servidor estará funcionando correctamente. En la pantalla de consola, no aparece el **prompt** del sistema ya que se queda esperando la finalización del proceso lanzado; algo que no ocurrirá hasta que nosotros le paremos.

- (11) Comprobar que realmente MySQL está funcionando con (situados desde la raíz de MySQL):

```
bin/mysqladmin version
```

Si todo ha ido bien, el resultado sería algo como:

```
bin/mysqladmin Ver 8.42 Distrib 5.5.16, for linux2.6 on x86_64
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
```

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

```
Server version          5.5.16
Protocol version 10
Connection              Localhost via UNIX socket
UNIX socket             /tmp/mysql.sock
Uptime:                 1 min 27 sec
```

```
Threads: 1 Questions: 1 Slow queries: 0 Opens: 33 Flush tables: 1 Open tables:
26 Queries per second avg: 0.011
```

- (12) Comprobar que podemos cortar MySQL (situados desde la raíz de MySQL):

```
bin/mysqladmin -u root shutdown
```

Si volvemos a comprobar si MySQL está en pie mediante el comando **mysqladmin versión**, debería salir algo parecido a:

```
bin/mysqladmin: connect to server at 'localhost' failed  
error: 'Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)'  
Check that mysqld is running and that the socket: '/tmp/mysql.sock' exists!
```

- (13) Lanzar de nuevo el servidor (situados desde la raíz de MySQL):

```
bin/mysqld_safe --user=mysql
```

- (14) Entrar en MySQL (situados desde la raíz de MySQL):

```
bin/mysql
```

No hay contraseñas para ningún usuario, por lo que es conveniente hacerlo (se explica más adelante)

### preparar MySQL para el inicio automático

Si se desea que MySQL se inicie automáticamente en cada encendido del ordenador, bastará con copiar el script **mysql.server** que se encuentra en el directorio **support-files** que, a su vez, está en la raíz de MySQL. La copia se realiza al directorio que contiene los scripts de inicio del sistema, normalmente **/etc/init.d**

El comando a ejecutar sería (estando en la raíz de MySQL):

```
cp support-files/mysql.server /etc/init.d/mysqld  
chmod +x /etc/init.d/mysqld
```

Eso le coloca en los scripts de inicio con el nombre **mysql** y le da permiso de ejecución. Desde ese momento el servicio mysql ya existe. Y es posible lanzar y parar mysql con los comandos:

```
service mysqld start
```

y para detener:

```
service mysqld stop
```

Para lanzar automáticamente en cada reinicio existen comandos en los diversos Linux. El más popular:

```
chkconfig --level 345 mysqld on
```

Que iniciará el servidor en los niveles 3,4 y 5 del sistema, con el código de secuencia 50.

En el caso de Ubuntu (y de otros Linux tipo Debian), ese comando no existe y actualmente se puede utilizar:

```
update-rc.d mysqld defaults
```

que coloca el servicio mysql en los niveles 2,3,4 y 5 y le asigna el código de secuencia 20. Este otro comando es equivalente al [chkconfig](#) anterior:

```
update-rc.d mysqld start 50 2 3 4 5 . stop 50 0 1 6
```

Ambos comandos tienen más opciones que serán fáciles de utilizar para quienes tengan conocimientos sobre cómo funciona el arranque en Linux.

Para que estas opciones funcionen, el archivo de configuración **my.cnf**, situado normalmente en **/etc/my.cnf** debería contener estas líneas al menos (suponiendo que mysql se ha instalado en la ruta **/usr/local/mysql**):

```
[mysqld]
....
datadir=/usr/local/mysql/data
[mysql.server]
basedir=/usr/local/mysql
```

La ruta **/usr/local/mysql** en el ejemplo es la que apunta a la raíz de MySQL.

### desinstalar MySQL si la instalación es binaria ZIP

Si deseamos desinstalar MySQL del sistema; en este caso basta con parar el servidor, eliminar los servicios relacionados con MySQL (si es necesario) que se inician con el sistema y eliminar la carpeta raíz y todos los subdirectorios (comando **rm -R**)

### instalación mediante gestores de paquetes

Casi todos los Linux tienen predefinidos paquetes para instalar MySQL. Es el caso de los paquetes rpm que se pueden instalar fácilmente en cualquier Linux de tipo **Red Hat** (como **Fedora**).

Es la instalación recomendada desde el fabricante, la cuestión es que la localización de los programas varía respecto a la instalación mediante binarios genéricos. Por paquetes MySQL se organiza de la siguiente manera:

directorio	uso
<b>/usr/bin</b>	Ubicación de los programas MySQL y de sus scripts
<b>/usr/sbin</b>	Ubicación del servidor <b>mysqld</b>
<b>/var/lib/mysql</b>	Archivos log y bases de datos
<b>/usr/share/info</b>	Manual en formato <b>info</b> de Linux
<b>/usr/share/man</b>	Manual en formato del comando <b>man</b> de Unix
<b>/usr/include/mysql</b>	Cabeceras de código fuente
<b>/usr/lib/mysql</b>	Archivos de librería
<b>/usr/share/mysql</b>	Raíz de MySQL. Mensajes de error, juegos de caracteres, ejemplos,...
<b>/usr/share/sql-bench</b>	<b>Benchmarks</b>
<b>/etc/apache2</b>	Archivos de configuración y soporte



### instalación en Linux tipo RedHat (como Fedora)

Se suele usar la utilidad **yum** (o bien la utilidad gráfica de instalación de paquetes):

```
yum install mysql-server
```

Esto descarga los archivos necesarios y cambia la contraseña del administrador de la base de datos (la pide por teclado).

Una vez instalado se usaría el comando:

```
service mysqld start
```

Lanza el servidor MySQL y lo coloca como servicio del sistema. Si deseamos que sea un servicio que funcione desde el arranque del sistema, se haría:

```
chkconfig --levels 235 mysqld on
```

De esa forma en los niveles de ejecución 2,3 y 5; el demonio de MySQL (el proceso servidor) se arranca desde el inicio. El código:

```
chkconfig --levels 235 mysqld off
```

Elimina del arranque al servicio.

### instalación en Linux tipo Debian (como Ubuntu)

En los Linux derivados de **Debian** (como **Ubuntu**) pueden instalar el servidor MySQL con:

```
sudo apt-get install mysql-server
```

También se puede utilizar la herramienta gráfica **Synaptic** para instalar el paquete de servidor de MySQL.

Se pedirá una contraseña administrativa y se habrá arreglado el PATH, se habrá colocado el archivo de configuración en **/etc/mysql/my.conf** y se colocará como servicio de arranque el script **/etc/init.d/mysql** que permite que el servidor puede lanzarse automáticamente durante el arranque.

En todo momento podemos parar el servidor con:

```
service mysqld stop
```

y lanzar con:

```
service mysqld start
```

Además podremos colocar el servicio mysql para que se inicie automáticamente mediante:

```
update-rc.d mysqld defaults
```

## (1.11.5) asegurando la instalación

### contraseñas de usuarios

Hay que intentar desde el primer momento que la instalación de MySQL no permita que los usuarios puedan sobrepasar sus privilegios y dañar las bases de datos. Por ello hay que asegurar que todos los usuarios (y en especial los administrativos) tienen contraseña.

En las instalaciones en Windows ya existen las cuentas de usuario y se las pone contraseña durante la instalación. Sin embargo algunas instalaciones en Windows (las genéricas) no tienen los usuarios preconfigurados. Para configurar inicialmente los

usuarios se lanza el script (comentado anteriormente) `mysql_install_db`. Con ello en la tabla `mysql.user` dispondremos de los usuarios iniciales.

En Windows hay un usuario `root` (superusuario) que tienen privilegios totales pero que solo puede acceder a la máquina local. Para permitir el acceso con esos privilegios desde otra máquina, se crea otro usuario `root` (durante la instalación de MySQL se pregunta esa posibilidad). En Linux los usuarios `root` permiten el acceso local.

Hay cuentas anónimas (no tienen nombre de usuario) algunas para acceder de forma local y otras no. Para asegurar el acceso se deben poner contraseñas a las cuentas anónimas o bien eliminarlas.

Para examinar la lista de usuarios se usa:

```
SELECT user, host, password FROM mysql.user;
```

Para cambiar la contraseña, por ejemplo:

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('Atrezzo');
```

En el ejemplo se cambia la contraseña para el superadministrador `root` cuando accede de forma local.

Otra forma (mejor puesto que modifica de golpe la contraseña de todos los root) es:

```
UPDATE mysql.user SET password=PASSWORD('Atrezzo')
WHERE user='root';
FLUSH PRIVILEGES;
```

La última instrucción es la que hace que el cambio se lleve a cabo.

### eliminación de la base de datos ejemplo

En muchas instalaciones de MySQL se dispone de una base de datos llamada `test` que, en realidad, es para pruebas. Dejarla puede proporcionar un agujero de seguridad, por lo que si no deseamos usarla para aprender se debe de eliminar con:

```
DELETE FROM mysql.db WHERE Db LIKE 'test%';
FLUSH PRIVILEGES;
```

La última orden (`FLUSH PRIVILEGES`) confirma los cambios.

## (1.11.6) diccionario de datos de MySQL

MySQL dispone de una instrucción fuera del estándar SQL para consultar el diccionario de datos, es la instrucción `SHOW`. Tras esa palabra simplemente se indica lo que deseamos consultar. Por ejemplo:

```
SHOW DATABASES;
```

Muestra la lista de bases de datos accesibles actualmente.

```
SHOW TABLES;
```

Muestra la lista de las tablas del esquema actual de trabajo.

Sin embargo también dispone de una forma más estándar de consulta de los metadatos a través de la base de datos `INFORMATION_SCHEMA`. Dentro de `INFORMATION_SCHEMA` se encuentran vistas de toda la estructura de metadatos de MySQL. Por ejemplo:

```
SELECT table_name, table_schema FROM INFORMATION_SCHEMA.TABLES;
```

muestra la lista de tablas en MySQL indicando el esquema en el que se encuentran alojadas. Las Vistas más usadas son:

Vistas del INFORMATION_SCHEMA	Uso
<b>TABLES</b>	Muestra todas las tablas accesibles desde nuestro usuario.
<b>COLUMNS</b>	Muestra las columnas de las tablas.
<b>USER_PRIVILEGES</b>	Lista de privilegios de usuario
<b>SCHEMA_PRIVILEGES</b>	Privilegios de esquema
<b>TABLE_PRIVILEGES</b>	Muestra a los privilegios asignados a las tablas.
<b>COLUMN_PRIVILEGES</b>	Privilegios de columna
<b>CHARACTER_SETS</b>	Juegos de codificación de caracteres disponibles
<b>TABLE_CONSTRAINTS</b>	Restricciones
<b>KEY_COLUMN_USAGE</b>	Columnas clave
<b>ROUTINES</b>	Procedimientos y funciones almacenados en la base de datos
<b>VIEWS</b>	Vistas
<b>TRIGGERS</b>	Lanzadores de código

### (1.11.7) configuración de MySQL. Uso de opciones

#### uso de opciones en la línea de comandos

Se trata de acompañar a los comandos MySQL de opciones en la misma línea en la que se lanza el comando a fin de modificar su funcionamiento.

Así por ejemplo el comando **mysql** que permite conectar con la base de datos, admite diversas opciones, por ejemplo la opción **--execute** permite ejecutar un comando al comunicarse con la base de datos. Ejemplo:

```
mysql -u root -p --execute="SHOW DATABASES;"
```

Muestra por pantalla directamente el resultado de la instrucción SQL.

Muchas opciones admiten un formato corto que realiza exactamente esa operación. Para el ejemplo, vale:

```
mysql -u root -p -e "SHOW DATABASES;"
```

Casi todos los comandos MySQL admiten opciones, pero el más utilizado es **mysqld**, para verlas todas se usa:

```
mysqld --help --verbose
```

#### uso de opciones desde el archivo de configuración

Se puede configurar el funcionamiento de MySQL gracias al archivo **my.cnf** o **my.ini**. Como consejo, se debería de llamar **my.cnf** y así tanto en Windows como en Linux manejamos el mismo nombre.

En la mayoría de instalaciones se instala solo (bien en **C:\Windows** o bien **/etc** o en **/etc/mysql**); en otras tendremos que crearlo y eso es mejor hacerlo copiando los archivos de muestra de configuración de la carpeta **support-files** (Linux) o de la raíz de MySQL (Windows); el más usado para es el **medium**. Luego se modifican las opciones deseadas.

El archivo de configuración en **Windows** se puede usar de esta forma (en orden de cuál se ejecuta primero):

Ruta al fichero	Uso
<b>DirectorioWindows</b> \my.ini	Opciones globales (se aplican a todos)
c:\my.cnf	Opciones globales
<b>DirectorioRaízMYSQL</b> \my.ini	Opciones globales
---defaultExtraFile=ruta	Usa como archivo de configuración el de la ruta indicada

Mientras que en **Linux**

Ruta al fichero	Uso
/etc/my.cnf	Opciones globales
/etc/mysql/my.cnf	Opciones globales
<b>\$MYSQL_HOME</b> /my.cnf	Opciones que se aplican al servidor entero (pero sólo a él)
---defaultExtraFile=ruta	Usa como archivo de configuración el de la ruta indicada
~/.my.cnf	Opciones específicas del usuario

En ese archivo de configuración (y en cualquier otro de MySQL) hay que tener en cuenta que:

- Cualquier línea que comience con el signo **#** indica un comentario (texto que no se tiene en cuenta y que solo sirve para documentar el propio archivo).
- El documento se divide en secciones, las cuales se indican mediante el nombre de la sección encerrada entre corchetes (por ejemplo **[mysqld]**). La sección se suele corresponder con una de las aplicaciones de MySQL, por ello hay sección **mysqld**, **mysqladmin**,...
- El resto son instrucciones que constan de una variable de sistema (también llamada simplemente opción) seguida del signo "=" y del valor de dicha opción.

En la raíz de instalación del sistema (o en **support-files** en Linux) de MySQL hay una serie de archivos de configuración que pueden servir de plantilla. Es decir están pensados para ser copiados y modificados como archivo principal de configuración. El nombre de estos archivos (**my-small.cnf**, **my-medium.cnf**, **my-large.cnf** y **my-huge.cnf**) hace referencia al tamaño de las bases de datos que ha de manejar el servidor.

### algunas opciones del archivo my.cnf (o my.ini)

Se comentan las más utilizadas de la sección **mysqld** (afectan al funcionamiento del servidor MySQL)

- **basedir=ruta**. Ruta a la raíz MySQL
- **datadir=ruta**. Ruta al directorio de datos
- **tmpdir=ruta**. Ruta al directorio para archivos temporales
- **user=usuario**. Indica el nombre de usuario con el que se iniciará sesión en MySQL.
- **port=puerto**. Puerto de escucha de MySQL
- **socket=ruta**. Fichero o nombre de socket a usar en las conexiones locales.
- **default-table-type=tipo**. Tipo de la Tabla **InnoDB**, **MyISAM** o **BDB** normalmente
- **general-log=valor**. Con valor **uno**, permite que funcione el archivo LOG para almacenar las consultas realizadas.
- **general-log-file=ruta**. Indica la ruta al registro general de consultas
- **log-error=ruta**. Permite indicar la ruta al registro de errores.
- **log=ruta**. Indica la ruta al registro de consultas
- **log-bin=ruta**. Permite indicar la ruta al registro binario.
- **slow-query-log=0|1**. Indica si se hace LOG de las consultas lentas.
- **slow-query-log-file=ruta**. Ruta al archivo que hace LOG de las consultas lentas
- **long-query-time=n**. Segundos a partir de los cuales una consulta que tardes más, se considerará una consulta lenta.
- **pid-file=ruta**. Ruta al archivo que almacena el identificador de proceso de MySQL
- **console**. Muestra los errores por consola independientemente de lo que se configure para **log\_error**.
- **flush**. Graba en disco todos los comandos SQL que se ejecuten (modo de trabajo, sin transacción)
- **language**. Especifica el idioma de los lenguajes de error, normalmente esots archivos de lenguaje, están bajo /usr/local/share
- **skip-grant-tables**. Entra al servidor saltándose las tablas de permisos, es decir todo el mundo tiene privilegios absolutos
- **skip-networking**. El acceso a MySQL se hará solo desde el servidor local.
- **standalone**. Para Windows, hace que el servidor no pase a ser un servicio.

### (1.11.8) ficheros LOG en MySQL

Hay cuatro registros (**logs**):

- **Registro de errores (Error Log)**. Indica cuando arrancó y se detuvo el servidor. Se graba por defecto en la carpeta de datos de MySQL (archivo **host\_name.err**, donde host\_name es el nombre del servidor), pero la variable de sistema **log\_error** permite indicar otra ruta si fuera necesario.

- **Registro general de consultas (General Log File).** Está en la carpeta de datos de MySQL, salvo que se indique la variable **general-log-file**. Contiene las consultas realizadas. Es el archivo **host\_name.log**.
- **Registro binario (Binary Log).** Registra instrucciones DML. Los archivos binarios se almacenan por defecto en el directorio de datos. Sirve para intentar restaurar una base de datos en caso de desastre. Es binario, por lo que su manejo es complicado, para ver el contenido se usa la utilidad **mysqlbinlog** de esta forma:

```
mysqlbinlog archivoLOG
```

- **Registro de consultas lentas (Slow Query Log File).** Registra las consultas que tardaron un poco del tiempo mínimo establecido. El archivo está (salvo que se especifique **slow-log-file** como parámetro) en la carpeta de datos de MySQL con el nombre **host\_name-slow.log**.