U4-LENGUAJE SQL: DDL

1. El lenguaje SQL

¿Qué es el lenguaje SQL?

En las unidades didácticas anteriores has aprendido que mediante el **modelo Entidad-Relación** se pueden **modelizar** situaciones del mundo real, de manera que disponemos de una herramienta gráfica para trasladar los elementos de un **sistema de información y sus relaciones**, a un esquema manejable que puede ser fácilmente interpretado por cualquiera que conozca las reglas por las que se rige. Posteriormente has comprobado cómo se puede **trasladar ese modelo Entidad-Relación** a otro esquema de información más orientado a su tratamiento, como es el **modelo relacional**, basado en tablas, y debes saber normalizar las bases de datos resultantes, en base a las dependencias existentes en los datos, para mejorar su diseño evitando al máximo las redundancias de información.

No obstante, sólo tienes una visión parcial, sabes que hasta ahora todo lo que has hecho es **diseñar** la base de datos, pero que aún no has abordado la parte práctica del asunto: **Construir y explotar la base de datos.** Por eso te habrás preguntado: ¿Dónde acaba todo esto? **¿De qué manera podemos almacenar y tratar la información esquematizada en los modelos E-R y relacional por medio de un Sistema Gestor de Bases de Datos**? La respuesta a estas preguntas se llama **SQL**.

> **SQL** es un lenguaje que nos **permite interactuar con los SGBD Relacionales** para especificar las operaciones que deseamos realizar sobre los datos y su estructura.
> SQL son las siglas de **S**tructured **Q**uery **L**anguage (**Lenguaje de Consulta Estructurado).**
> Es un **lenguaje declarativo**, lo cual quiere decir que en él se especifica al Sistema Gestor de Base de Datos **qué queremos obtener** y no la manera de cómo conseguirlo.
> Es un **lenguaje no procedimental** porque no necesitamos especificar el procedimiento para conseguir el objetivo, sino el objetivo en sí. **No es un lenguaje de programación** como puede ser Java o C.

1.1. SQL, un poco de historia.

El **origen de SQL** está íntimamente ligado al de las **bases de datos relacionales**. En 1970 **Codd** propuso el modelo relacional que ya has estudiado en unidades anteriores. Basándose en esta idea en los **laboratorios de IBM** se definió el lenguaje **SEQUEL** (Structured English QUEry Language) que posteriormente sería implementado por el SGBD **System R**. Más tarde otros fabricantes de SGBD lo adoptarían, de forma que se acabó convirtiendo en un estándar de la industria informática.

En 1986 el ANSI (Instituto Americano de estándares) publicó la **primera versión** estándar del lenguaje, hoy conocido como **SQL-86**. Al año siguiente este estándar es adoptado por la **ISO** (Organismo Internacional de Estandarización), y a partir de entonces se han sucedido nuevas versiones que ampliaban en capacidad a las anteriores.

1.2. SQL en la actualidad.

Hoy día **SQL es el lenguaje de consulta y manipulación de datos más extendido** y utilizado por todos los **desarrolladores y fabricantes de SGBD**. A lo largo de los años se han ido ampliando sus características conforme se sucedían avances en la tecnología informática.

A continuación te presentamos una tabla que te explica la **evolución de este lenguaje desde el lejano 1986** hasta ahora.

Año	Versión	Alias	Características
1986	SQL-86	SQL-87	Primera publicación hecha por ANSI. Confirmada por ISO en 1987.
1989	SQL-89		Revisión menor.
1992	SQL-92	SQL2	Revisión mayor.
1999	SQL-99	SQL3	Entre otras características se añade la orientación a objetos.
2003	SQL:2003		Profundiza en la orientación a objetos e introduce características de XML.
2006	SQL:2006		Define las maneras en las cuales el SQL se puede utilizar conjuntamente con XML. Además, permite integrar dentro de su código SQL el uso de XQuery, lenguaje de consulta XML publicado por el W3C.
2008	SQL:2008		Permite el uso de la cláusula ORDER BY fuera de las definiciones de los cursores. Incluye los disparadores del tipo INSTEAD OF. Añade la sentencia TRUNCATE.

Como ves **SQL es un lenguaje** con cierta antigüedad que **ha tenido que evolucionar** para **adaptarse a nuevas características y requerimientos**. Es importante resaltar que **SQL no es propiedad de ningún fabricante, sino que es una norma a seguir**, desgraciadamente los fabricantes de software no suelen implementar SQL puro en sus productos, sino que a menudo incorporan pequeñas variaciones para conseguir funcionalidades concretas en sus desarrollos. Esto hace que lo qué debería ser un estándar no lo sea completamente en la realidad.

Como resumen de lo anterior, un concepto importante que debes asimilar es **que SQL es el lenguaje que usan los SGBDR en la actualidad**, y como tal, su estudio y práctica resulta imprescindible en tu formación como profesional informático.

2. ¿Cómo se usa SQL?

Bien, **ya sabemos qué es SQL**. **Ahora vamos a aprender cómo podemos usarlo para nuestros propósitos**. No olvides que estamos estudiando un lenguaje de interacción con bases de datos y que la herramienta para acceder y manipular esas bases de datos son los sistemas gestores de base de datos.

Los SGBDR permiten dos modos de acceso a las bases de datos:

Modo interactivo, destinado principalmente a los **usuarios finales, avanzados u ocasionales, en el que las diversas sentencias SQL se escriben y ejecutan en línea de comandos, utilizando un intérprete de órdenes**. Este modo es el que utilizaremos en las prácticas de esta unidad didáctica, porque es el que permite centrarnos exclusivamente en SQL y aprender el lenguaje de una forma directa y sin necesidad de otro tipo de conocimientos.

Modo embebido, destinado al uso por parte de los **programadores. En este caso las sentencias SQL se introducen en lenguajes de programación, llamados lenguajes anfitrión** (por ejemplo Java, lenguajes de la plataforma .NET de Microsoft, PHP, C++, etc.), de manera que el **resultado es una mezcla de ambos**. En este caso **el lenguaje anfitrión aporta lo que le falta a SQL, es decir la programación**. Aunque éste es el modo habitual en el que se utiliza SQL cuando se desarrollan aplicaciones, su estudio de esta forma requeriría el estudio simultáneo del lenguaje anfitrión, por lo qué pospondremos su estudio a una unidad didáctica posterior para no complicar el estudio de lo que realmente nos interesa en este momento.

3. Empezando a practicar SQL.

En este momento te estarás haciendo muchas preguntas, y la mejor manera de responderlas es empezando a **ver en la práctica qué es eso de SQL** y como usarlo.

Ya sabes que **a través de SQL interactuaremos con un SGBDR que a su vez gestiona una o más bases de datos relacionales**. Por lo tanto lo primero que debemos hacer para empezar a aprender y practicar SQL es disponer de un SGBDR en nuestro ordenador de prácticas.

Se nos presenta aquí la primera decisión que adoptar **¿Qué SGBDR utilizar?** La respuesta es al mismo tiempo sencilla y complicada, veamos:

- Es sencilla porque nos vale **cualquier SGBDR que implemente al menos SQL-99.**
- Es complicada porque existen bastantes productos en el mercado que cumplen ese requisito.

Para las prácticas **utilizaremos como SGBD ORACLE.**
Por eso lo primero que debes tener es instalada en tu máquina el SGBD ORACLE, a ser posible la 11g, e imprescindible el un Cliente lo más cómodo posible para atacar y manipular dicho SGBD, en nuestro caso recomiendo el TOAD.

ORACLE **soporta muchas características de SQL-99**. Su obtención es fácil, **se puede descargar gratuitamente de la web (con ciertas características limitativas)** y existe mucha documentación accesible sobre él. Además es un producto de amplia implantación en el mercado actual de grandes compañías que deben mover enormes cantidades de datos con alta seguridad. Ni que decir tiene que **todo lo que se estudie aquí será válido en cualquier otro SGBDR**.

Oracle es un sistema de gestión de base de datos objeto-relacional (o ORDBMS por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando:

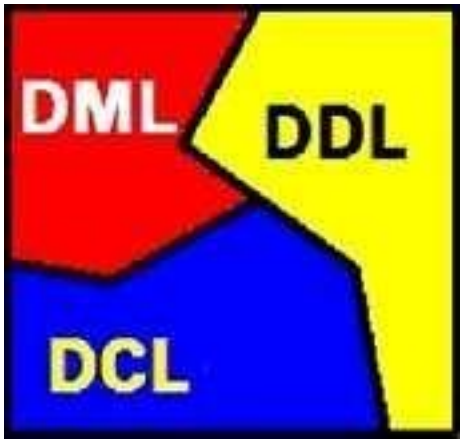
- Soporte de transacciones,
- Estabilidad,
- Escalabilidad
- Soporte multiplataforma.

Su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySQL o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

Lo primero que debes hacer es **descargar e instalar** el software subido al Aula Virtual.

4. Componentes del lenguaje SQL.

El lenguaje SQL está compuesto por **sentencias**. Esas sentencias se pueden clasificar en tres grupos:



Sentencias DDL (Lenguaje de Definición de Datos): Sirven para **crear, modificar y borrar elementos estructurales en los SGBDR**, como:

- bases de datos,
- tablas,
- índices,
- restricciones, etc.

Las definiciones de esos objetos quedan almacenadas en el diccionario de datos del sistema.

Sentencias DML (Lenguaje de Manipulación de Datos): Nos permiten indicar al sistema las **operaciones que queremos realizar con los datos almacenados en las estructuras creadas por medio de las sentencias DDL**. Por ejemplo son las sentencias que permitirán:

- generar consultas,
- ordenar,
- filtrar,
- añadir,
- modificar,
- borrar,
- etc.

Sentencias DCL (Lenguaje de Control): Un conjunto de sentencias orientado a gestionar todo lo relativo a:

- usuarios,
- permisos,
- seguridad,
- etc.

Con el tiempo han surgido **nuevas necesidades** en los SGBDR que han obligado a incorporar **nuevas sentencias** que no se pueden clasificar en los tres grupos clásicos anteriores**, cómo son las Sentencias para gestión de transacciones y bloqueos, las sentencias para replicación, etc.**

Las sentencias SQL a su vez se construyen a partir de:

Cláusulas: Que modifican el comportamiento de las sentencias. Constan de palabras reservadas y alguno de los siguientes elementos.
Operadores lógicos y de comparación: Sirven para ligar operandos y producir un resultado booleano (verdadero o falso).
Funciones de agregación: Para realizar operaciones sobre un grupo de filas de una tabla.
Funciones: Para realizar cálculos y operaciones de transformación sobre los datos.
Expresiones: Construidas a partir de la combinación de operadores, funciones, literales y nombres de columna.

A lo largo de estas unidades estudiaremos y practicaremos gran parte de las sentencias SQL, y por supuesto las más importantes y utilizadas. Aunque su estudio completo no sería posible aquí, por su elevadísimo número y casuística de cada una, y para completar tu formación te remitimos a los buenos manuales que sobre SQL circulan en la red y en las librerías. En especial **te recomendamos la documentación del SGBDR** que vas a utilizar, ORACLE. Puedes obtenerla de la misma web desde donde has descargado la aplicación.

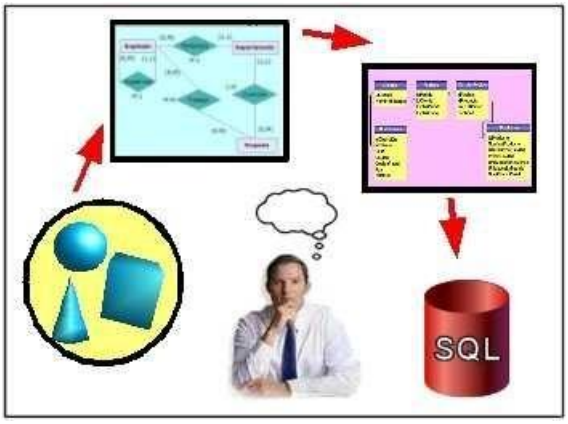
5. Un ejemplo para nuestras prácticas.

En esta unidad didáctica estudiaremos cómo SQL permite manipular las bases de datos. Comprobarás que la unidad que estás comenzando a estudiar está repleta de prácticas sobre lo estudiado. Y para hacer de hilo conductor vamos a plantear **un ejemplo** que nos permitirá probar todos los conceptos que vayamos introduciendo. De esta forma el ejemplo se irá enriqueciendo conforme avances en el estudio de los contenidos de esta unidad y su comprensión y asimilación desde el principio te permitirá concentrarte en los aspectos propios de SQL y no en los del caso concreto que se quiere modelizar.

Hemos escogido un ejemplo que es bastante conocido y que se ha utilizado ampliamente en los libros dedicados al tema de bases de datos relacionales.

Como ya has aprendido en unidades anteriores la implementación en un SGBD de una base de datos es la culminación de un proceso que pretende modelizar una situación del mundo real para poder sistematizarla y tratarla por medios informáticos. Ya sabes que los pasos que se siguen para diseñar una base de datos relacional son:

Estudio una situación del mundo real que se pretende modelizar.
Diseño de un modelo conceptual de la situación.
Diagrama Entidad-Relación.
Traslado del diseño conceptual a un diseño lógico**. Modelo relacional**.
Implementación del modelo relacional en el SGBDR que se vaya a utilizar (SQL).



5.1. Planteamiento de la situación que queremos modelizar.

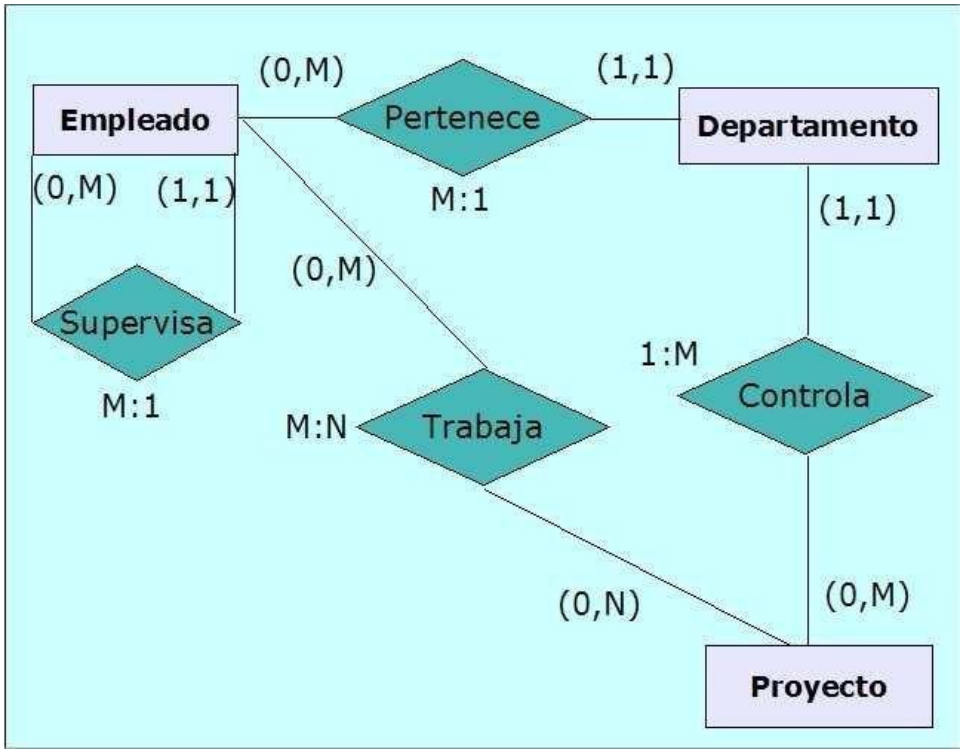
Piensa en la siguiente situación del mundo real:

Una **empresa** pretende desarrollar una base de datos de **empleados y proyectos**.
La empresa está estructurada en **departamentos**, en cada uno de los cuales se están desarrollando uno o varios **proyectos**, de forma que un proyecto sólo depende de un departamento.
Por otro lado, cada departamento consta de **uno o varios empleados** que trabajan de forma exclusiva para ese departamento, pero pueden trabajar simultáneamente en varios proyectos.
Cada empleado tiene un **jefe** encargado de **supervisar** su trabajo, pudiendo cada jefe supervisar el trabajo de varios empleados.
Tanto los empleados, como los departamentos y los proyectos se identifican por un código que es único para cada uno de ellos.
De cada **empleado** se quiere guardar su nombre y su fecha de ingreso en la empresa, así mismo todos los departamentos y los proyectos tienen un nombre descriptivo.
Por último, se quiere guardar en la base de datos **cuántas horas ha trabajado** cada empleado en los proyectos en los que está involucrado.

Hasta aquí el enunciado de nuestro caso práctico, veamos a continuación el diseño conceptual de este supuesto.

5.2. Diagrama E-R de nuestro caso práctico.

¿Has comprendido bien el enunciado de la situación planteada? Con las enseñanzas de las unidades didácticas anteriores **no tendrás problema en asimilar el siguiente diagrama E-R** que modeliza de forma conceptual la situación descrita.



Los atributos de cada entidad serian:

EMPLEADO: Código de empleado, nombre y fecha de ingreso en la empresa

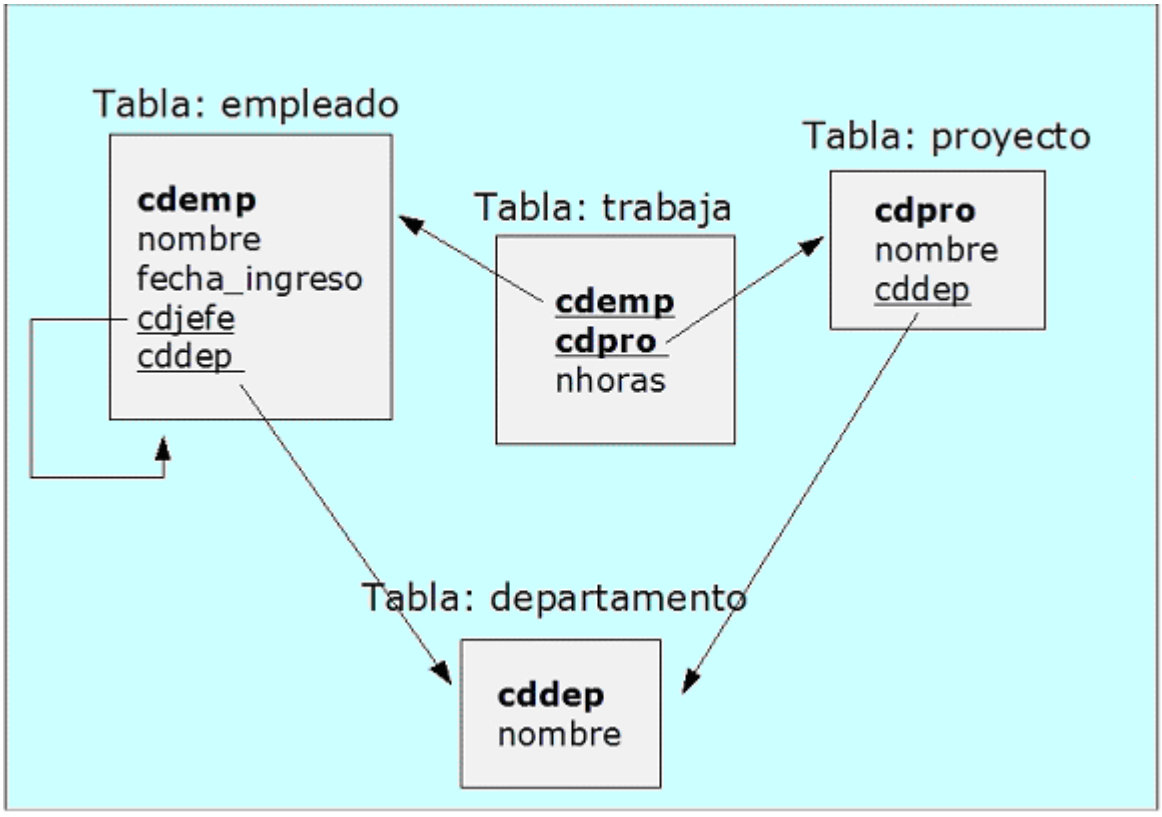
PROYECTO: Código de proyecto y nombre del mismo

TRABAJA: La relación M:M tiene un atributo de relación número de horas

DEPARTAMENTO: Un código único y el nombre del mismo

5.3. Modelo relacional de nuestro caso práctico.

Aplicando lo aprendido en las **unidades anteriores** no tendrás problemas en comprobar que el diseño lógico (esquema relacional) correspondiente al diagrama E-R anterior es el siguiente.



El esquema anterior está en 3FN (tercera forma normal) y consta de cuatro tablas. A partir de ahora y en lo que resta de esta unidad didáctica utilizaremos este esquema relacional para **ejemplificar los contenidos teóricos** relativos al lenguaje SQL. No olvides que el vehículo para realizar las prácticas será el gestor de base de datos MySQL, que debes tener instalado y en funcionamiento. También debes tener a mano la **documentación de MySQL** que se instaló junto con la aplicación, o en línea a través de la página web oficial del producto.

Más exactamente las tablas serían :

EMPLEADO (nombre, fecha_ingreso, cdjefe, cddep)
CP: cdemp
Caj: cdjefe → EMPLEADO
Caj: cddep → DEPARTAMENTO
VNN: cdjefe
VNN: cddep

DEPARTAMENTO (cddep, nombre)
CP:cddep

PROYECTO (cdpro, nombre, cddep)
CP: cdpro
Caj: cddep → DEPARTAMENTO
VNN: cddep

TRABAJA (cdemp, cdpro, nhoras)
CP: cdemp, cdpro
Caj: cdemp→ EMPLEADO
Caj: cdpro→ PROYECTO

6. Lenguaje de Definición de datos (DDL)

Ya sabes que **el lenguaje SQL se compone de sentencias**, y que éstas **se pueden clasificar en tres grupos**.

Empezaremos estudiando **aquéllas que nos permiten crear, modificar y borrar las estructuras de nuestra base de datos.**

Este grupo de sentencias **constituyen el Lenguaje de Definición de Datos (DDL).** Las sentencias del DDL suelen ser bastante dependientes del SGBD que estemos utilizando, por lo cuál tendremos que consultar su documentación para conocer todos los detalles.

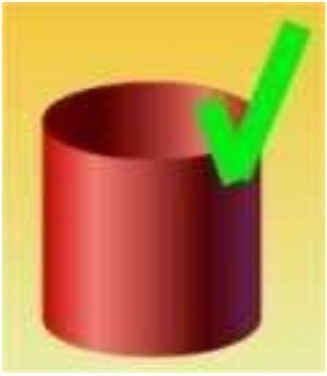
Veremos las sentencias DDL que por ahora vamos a necesitar para crear la estructura de la base de datos para nuestro ejemplo. La lista es relativamente corta:

Para bases de datos: CREATE DATABASE, DROP DATABASE
Para tablas: CREATE TABLE, ALTER TABLE y DROP TABLE
Para índices: CREATE INDEX y DROP INDEX

Como parece, es bastante sencillo, porque los nombres de las sentencias son bastante autoexplicativos de su finalidad. Además, CREATE no puede ser más que crear, ALTER no puede ser más que alterar o modificar y DROP no puede ser más que borrar o eliminar. Basta con poner detrás el tipo de elemento sobre el que quieres que se actúe: DATABASE, TABLE o INDEX. Parece lógico, ¿no?

Pero lo que le parece complicado no es la lista de sentencias, sino la lista de cláusulas y la sintaxis completa de cada una de ellas, que puede ser bastante larga, y no siempre tan evidente. Se debe practicar mucho para trabajar con soltura con todas ellas.

6.1. Creación y borrado de una base de datos.



Lo primero que debes aprender es a crear la base de datos, puesto que éste es el objeto que contiene a todos los demás elementos.

Cuando se crea una base de datos, en el SGBD ocurren muchos **procesos internos** que culminan con la asignación de **espacio físico** de almacenamiento para contener las estructuras de datos y acceso a los mismos de la base de datos, y también el registro en el diccionario de datos del SGBD de las características de la base de datos creada.

A continuación te presentamos la sintaxis de la sentencia para crear bases de datos en SQL, `CREATE DATABASE`.

```
CREATE DATABASE <nombre_bd>
```

En ORACLE nada más instalar el SGBD nos obliga a crearlo con una Base de Datos inicial, ORCL por defecto, podemos crear más o usar esta, si queremos crear más necesitamos cambiar el tsnames y darles espacio, estos serian los pasos necesarios, añado url:

http://www.dba-oracle.com/oracle_create_database.htm#standard_create_database_syntax

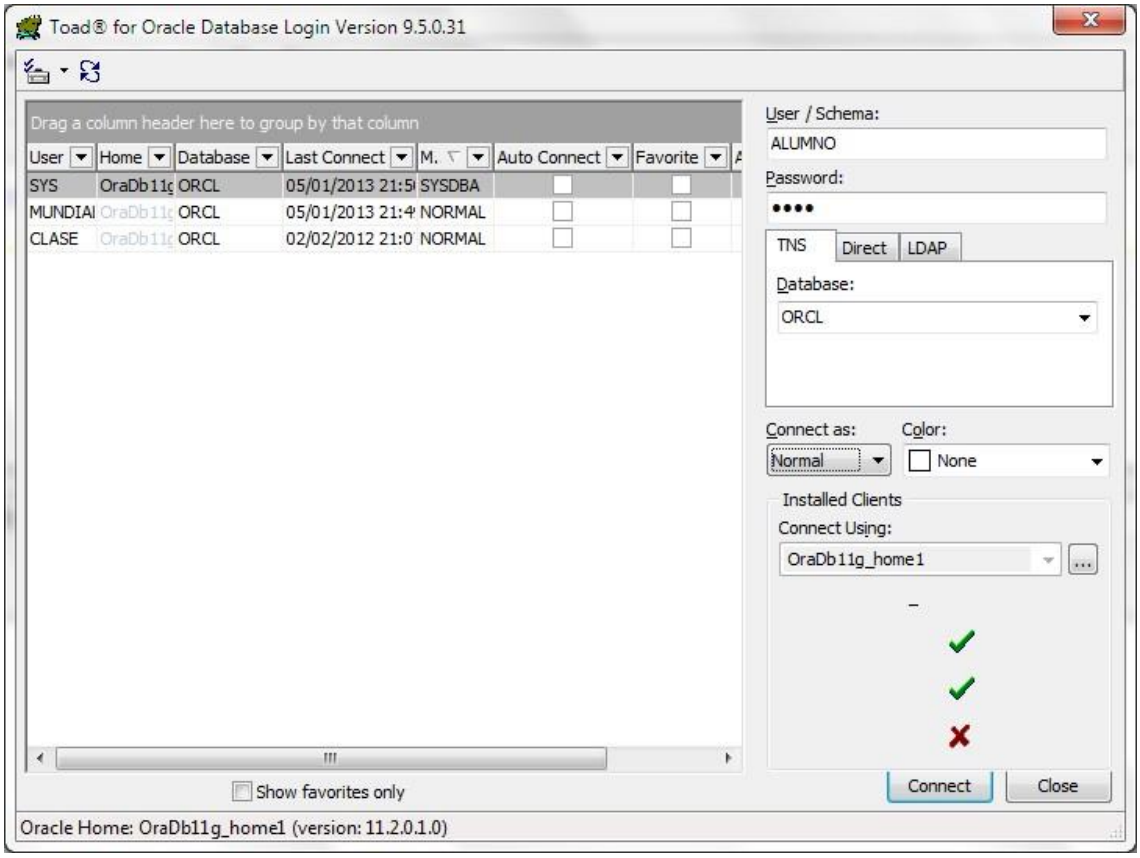
Nosotros vamos a trabajar con la BD por defecto, ORCL pero ORACLE tiene algo muy potente que son los ESQUEMAS, dentro de una misma BD podemos tener diferente ESQUEMAS que permiten dar seguridad a la BD incluso BD dentro de otras. Como inicialmente trabajamos nosotros con SYS, no es conveniente trabajar desde un superusuario, por seguridad y porque tiene muchos objetos del sistema, mejor crear un Usuario limpio y darles permisos, esto se realiza con el lenguaje DCL que daremos el 3 trimestre, pero para crearlo deberemos realizar:

CREACIÓN DE USUARIO/SCHEMA (Lenguaje DCL)

CREATE USER "ALUMNO" PROFILE "DEFAULT" IDENTIFIED BY "1234"; -- Crea el Usuario

GRANT ALL PRIVILEGES TO "ALUMNO"; -- Le da todos los permisos de acción

Una vez hecho, nos conectaremos con el nuevo usuario, y nos aseguraremos de conectar como NORMAL.



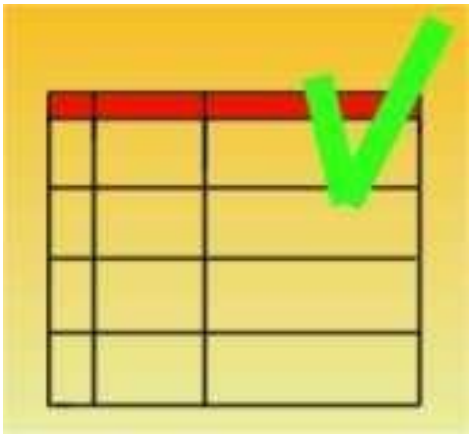
Para eliminar Usuarios o Bases de Datos está:
DROP USER:

Para borrar una base de datos se utiliza la sentencia `DROP DATABASE`:

```
shutdown abort;
startup mount exclusive restrict;
drop database;
exit;
```

Hay que tener cuidado ya que borraremos la base de datos y todo lo que contiene. Los SGBD introducen medidas de seguridad de forma que sólo los usuarios autorizados (normalmente el administrador y el propietario de la base de datos) pueden ejecutar este tipo de sentencias. Veremos la gestión de la seguridad y los usuarios más adelante, dentro de las funciones de Administración.

6.2. Creación, modificación y borrado de tablas.



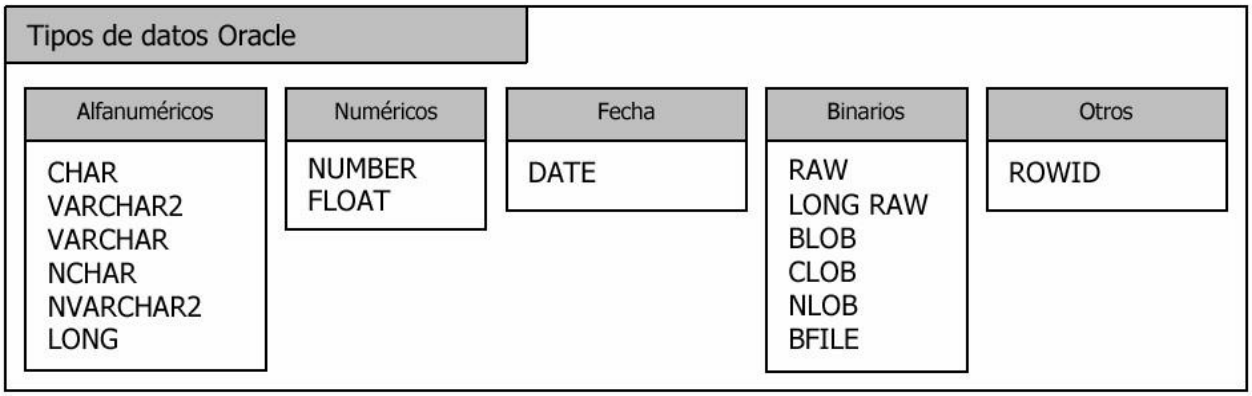
En este momento dispones de un SGBD funcionando y una base de datos recién creada, pero **esa base de datos no es más que una estructura vacía. Ya sabes que los objetos que se utilizan para albergar datos en un SGBD relacional son las tablas** y a continuación vamos a aprender a crearlas, modificarlas y borrarlas.

Para **crear una tabla** en SQL se utiliza la sentencia `CREATE TABLE`.

A continuación te presentamos una **sintaxis reducida de esta sentencia**:

```
CREATE TABLE nombre_tabla(
      COLUMNA TIPO
            [NOT NULL]
            [UNIQUE]
            [PRIMARY KEY]
            [DEFAULT valor]
            [REFERENCES Nomtabla][columna[,columna]][ON DELETE
                  CASCADE]]
            [CHECK condicion],
      COLUMNA TIPO [NOT NULL],
      . . .
{CONSTRAINT nombre_clave_primaria PRIMARY KEY (columnas_clave)}
{CONSTRAINT nombre_clave_foránea
      FOREIGN KEY(columnas_clave) REFERENCES tabla_detalle(
      columnas_clave )
      {ON DELETE CASCADE} }
)
{TABLESPACE tablespace_de_creación}
{STORAGE( INITIAL XX{K|M} NEXT XX{K|M} )}
```

- `ALTER TABLE nombretabla`
  - `[ADD (columna[,columna]...)`
  - `[MODIFY (columna[,columna]...)`
  - `[DROP COLUMN (columna[, columna]...)`
  - `[ADD CONSTRAINT restriccion]`
  - `[ADD CONSTRAINT restriccion];`



Habrá que escoger el dominio más adecuado para los valores que debe contener cada columna. **ORACLE**, como todos los SGBDR, **ofrece una amplia variedad de tipos de dato diferentes**. La mayoría de ellos pertenecen al estándar SQL, pero como siempre habrá que consultar la documentación del sistema que estamos utilizando.

Cualquier tipo de dato utilizado como dominio de una columna puede contener el valor NULL. Este valor hay que interpretarlo como **ausencia de valor**, no hay que confundirlo, por ejemplo, con el valor 0 de una columna de tipo `NUMBER` la cadena vacía en una columna de tipo `VARCHAR2`. Una columna que forme parte **de la clave primaria de una tabla no puede contener el valor NULL**, y **tampoco aquellas columnas definidas como NOT NULL**, será el propio SGBDR el encargado de mantener esta restricción e impedirá cualquier intento contrario a la regla.

En nuestro ejemplo la creación de tablas quedaría así:

```
CREATE TABLE departamento (
cddep VARCHAR2(2) NOT NULL,
nombre VARCHAR2(30),
ciudad VARCHAR2(20),
CONSTRAINT pk_departamento primary key (cddep)
);

CREATE TABLE empleado (
cdemp VARCHAR2(3) NOT NULL,
nombre VARCHAR2(30),
fecha_ingreso DATE,
cdjefe VARCHAR2(3) NOT NULL,
cddep VARCHAR2(2) NOT NULL,
constraint pk_empleado primary key (cdemp),
constraint fk_empleado_dep FOREIGN KEY (cddep)
REFERENCES departamento(cddep) ON delete CASCADE,
constraint fk_empleado_jefe FOREIGN KEY (cdjefe)
REFERENCES empleado(cdemp)
ON DELETE SET NULL
);

CREATE TABLE proyecto (
codpro VARCHAR2(3) NOT NULL,
nombre VARCHAR2(30),
cddep VARCHAR2(2) NOT NULL,
constraint pk_proyecto primary key (codpro),
constraint fk_proyecto_dept FOREIGN KEY (cddep)
REFERENCES departamento(cddep)
);

CREATE TABLE trabaja (
cdemp VARCHAR2(3) NOT NULL,
cdpro VARCHAR2(3) NOT NULL,
nhoras NUMBER DEFAULT 0
);

alter table trabaja add constraint pk_trabaja PRIMARY KEY (cdemp,cdpro);

alter table trabaja add constraint fk_trabaja_empl FOREIGN KEY (cdemp)
REFERENCES empleado(cdemp) ON DELETE CASCADE;

alter table trabaja add constraint fk_trabaja_proy FOREIGN KEY (cdpro)
REFERENCES proyecto(cdpro) ON DELETE CASCADE;
```

BASES DE DATOS. Diseño Físico

Algunas consideraciones al respecto:

Nombres válidos: Al igual que en el caso de los nombres válidos de base de datos**, los nombres de tablas, índices y columnas deben seguir unas reglas que dependen del SGBDR**. En ORACLE básicamente deben seguir:

1. Debe comenzar con una letra.
2. No puede tener más de 30 caracteres.
3. Debe ser compuesto por los siguientes caracteres especiales o caracteres alfanuméricos: $ _ y #.
4. No puede ser una palabra reservada.

Restricciones de integridad y referencia: Observa que **hemos definido junto con las columnas de cada tabla las restricciones para hacer cumplir la reglas de integridad de la entidad (**`PRIMARY KEY`**) y de referencia (**`FOREIGN KEY`**)**. Faltaria las UK, que se colocan como UNIQUE. Ej:

```
alter table alumnos
  add constraint UQ_alumnos_documento
  unique (documento);
```

En cuanto a como poner las Constraint (constraint entendemos como restricciones), vemos que se pueden poner de dos maneras, o bien directamente en la creación de la tabla, o bien, fuera de la creación de la tabla con un ALTER TABLE.

Foreign Key

Por ejemplo si queremos que se cumpla **la integridad de referencia en la columna cddep de la tabla empleado,** de manera que todos los empleados tengan un código de departamento que exista en la tabla departamento debemos utilizar la estructura:

```
FOREIGN KEY (<columnas que forman la clave externa>)
 REFERENCES tabla(<columnas que forman la clave primaria>)
```

Funciona de la siguiente manera: las **columnas indicadas en** `FOREIGN KEY` **hacen referencia a las columnas indicadas en** `REFERENCES`. Se puede indicar la acción a realizar cuando se modifican (`ON UPDATE`) o se borran (`ON DELETE`) los valores especificados en `REFERENCES`, esta acción puede ser: impedir, arrastrar el cambio, poner a nulo o no hacer nada (respectivamente: `CASCADE, SET NULL, NO ACTION`). ORACLE no permite por su gran complejidad y peligro en ON UPDATE, otros SGBD si lo contemplan.

En SQL quedaría así:

```
  FOREIGN KEY (cddep)
   REFERENCES departamento(cddep)
   ON DELETE CASCADE
```

Lo que se interpreta como: La columna cddep de la tabla empleado hace referencia (FOREIGN KEY (cddep)) a la columna cddep de la tabla departamento (REFERENCES departamento(cddep)). No puede existir una fila en empleado que contenga un valor de cddep que no exista previamente en la tabla departamento. En el hipotetico caso de poder poner ON UPDATE, ssi se modifica el cddep de una fila en la tabla departamento (ON UPDATE) el cambio se transmite automáticamente a todas las filas de la tabla empleado que tuviesen ese departamento (CASCADE). Si se intenta borrar una fila de la tabla departamento (ON DELETE), se borraran también en la tabla de empleados con ese valor de departamento (RESTRICT).

Es posible **modificar la estructura de una tabla** por medio de la sentencia `ALTER TABLE`. Su sintaxis es muy parecida a la de `CREATE TABLE`.

El **borrado de una tabla** es muy simple, aunque hay que tener cuidado en esta operación, porque un borrado indebido puede tener consecuencias graves, si se borran datos que no se debían borrar. Asegúrate dos veces siempre antes de usar esta sentencia.

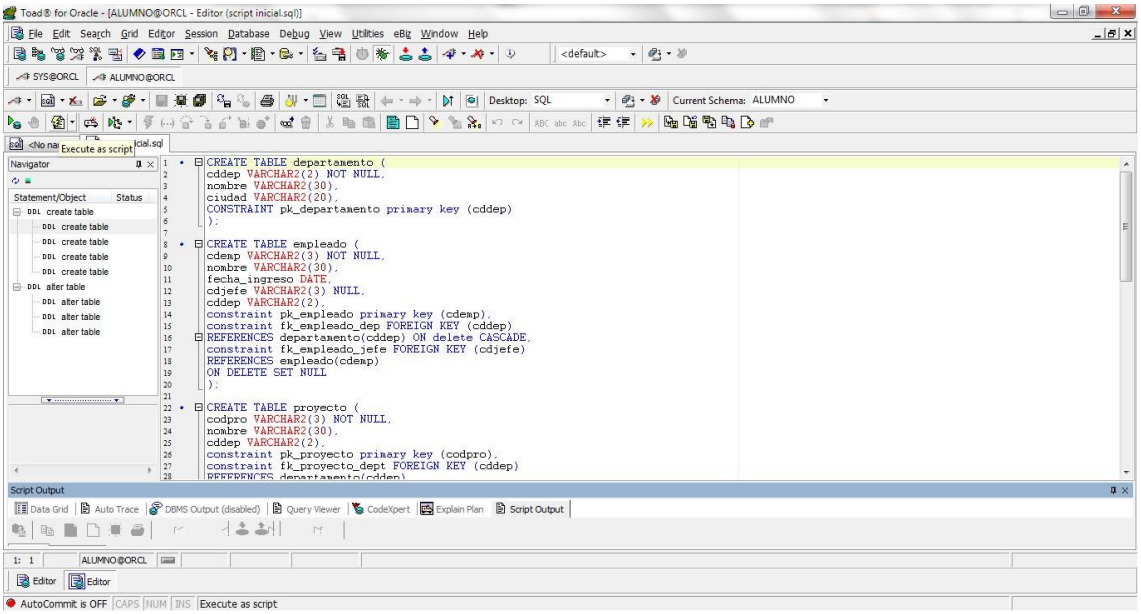
La sentencia a utilizar es:

```
DROP TABLE <nombre_tabla>
```

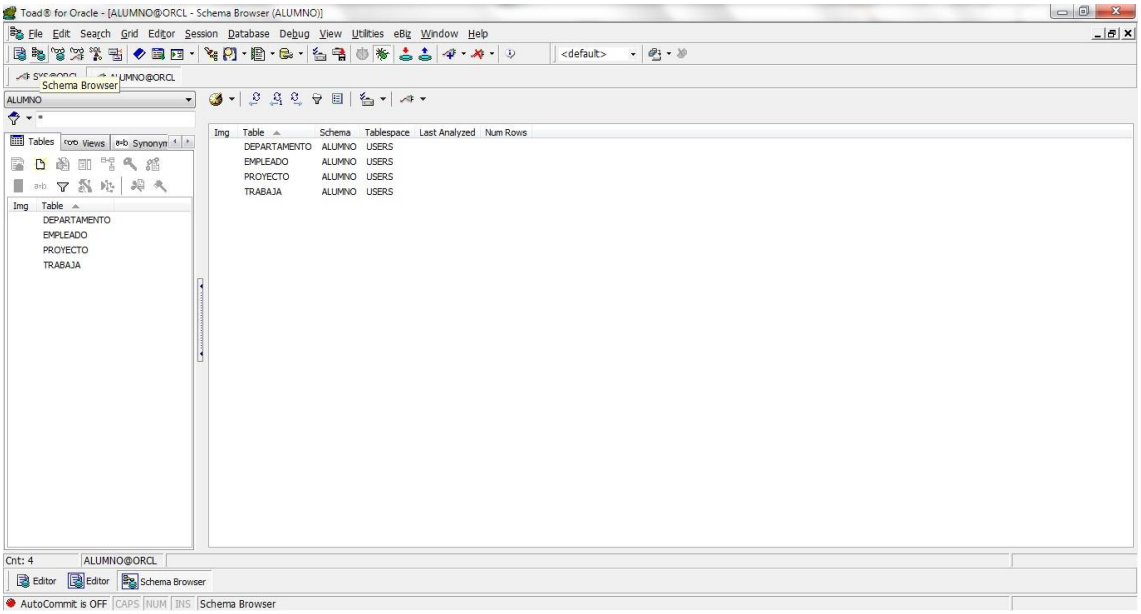
Por ejemplo, para borrar la tabla empleado:

```
DROP TABLE empleado
```

Gráficamente usando el TOAD tendremos:



Y una vez ejecturado, bien sentencia a sentencia bien con el "rayo" Amarillo que permite lanzar como un script todas las sentencias a la vez, para ellos toda setencia de be acabar en F9. Si vamos una a una poniendo el cursor sobre la sentencia y pulsando F9 ejectuta dicha sentencia, también seleccionando todo o con Sifht + F9, entonces si nos vamos al Schema Browser, podremos ver las tablas creadas y todos los elementos:



ACTUALIZACIÓN o BORRADO de componentes

Bueno hasta aquí hemos visto como crear las tablas y sus restricciones (constraint de clave primaria, clave única o clave ajena). Pero qué pasa si con el tiempo una vez creadas las tablas necesito modificar o actualizar las mismas. Para ellos necesitamos alterar la tabla por lo tanto se realiza con una ALTER TABLE.

Imagina que a los empleados queremos añadirles el atributo edad.

```
ALTER TABLE EMPLEADOS ADD EDAD NUMBER(3);
```

Con esto se añadiría a la tabla este campo como numérico de 3. Imaginar que ahora quiero que sea VNN y que tenga hasta 5 dígitos.

```
ALTER TABLE EMPLEADOS MODIFY EDAD NUMBER(5) NOT NULL;
```

Y si quiero volver a quitárselo

```
ALTER TABLE EMPLEADOS MODIFY EDAD NUMBER(3);
```

Y ahora voy a borrar el campo, quitarlo para siempre de la tabla (no borrar su contenido que eso es DML):

```
ALTER TABLE EMPLEADOS DROP COLUMN EDAD;
```

Otras CONSTRAINT

Aparte de las CONSTRAINT de PK, UK y FK, tenemos otras constraint para indicar los posibles valores de un campo, en realizada cuando le ponemos NOT NULL estamos haciendo este tipo de CONSTRAINT de forma implícita. Pero se pueden hacer más complejas, por ejemplo, imaginemos que el campo EDAD anterior queremos restringirlo que este entre 10 y 99.

```
ALTER TABLE EMPLEADOS ADD CONSTRAINT check_edad
  CHECK (EDAD BETWEEN 10 and 99)
```

O que queremos que sea una edad de 20, 25 o 30 o ninguna.

```
ALTER TABLE EMPLEADOS add CONSTRAINT check_edad2
  CHECK (EDAD IN (20,25,30) OR EDAD IS NULL);
```

Si es alfanumérico debe ir entre comillas simples:

```
ALTER TABLE EMPLEADOS add CONSTRAINT tipo_empleado
  CHECK (TIPO IN ('INGENIERO', 'ADMINISTRATIVO', 'DIRECTIVO') OR EDAD IS NULL);
```

ELIMINAR o DESABILITAR CONSTRAINT:

Si deseamos eliminar la restricción:

```
ALTER TABLE EMPLEADOS DROP CONSTRAINT chech_edad;
```

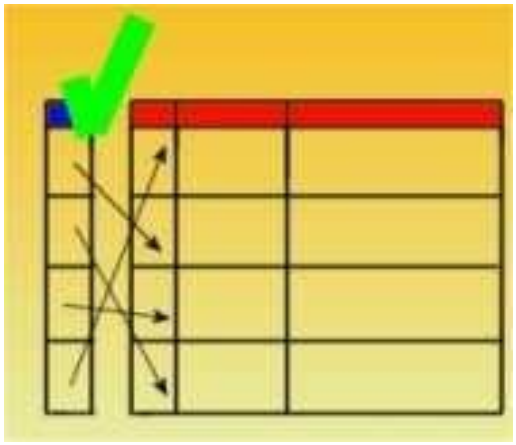
Pero a veces no queremos eliminarla para siempre solo que no se aplique temporalmente:

```
ALTER TABLE EMPLEADOS DISABLE CONSTRAINT chech_edad;
```

Y volverla a activar:

```
ALTER TABLE EMPLEADOS ENABLE CONSTRAINT chech_edad;
```

6.3. Creación y borrado de índices.



Cuando tenemos que buscar **manualmente** un dato en una lista que es muy larga, solemos tardar bastante más que si la lista es pequeña, incluso aunque hagamos la búsqueda aplicando algún **criterio** para no tener que comprobar todos los datos uno a uno desde el principio.

¿Ocurre algo parecido en las bases de datos?
¿El tiempo que tarda el SGBD en encontrar un dato es independiente del número de datos que contengan las tablas?

La respuesta es que **no**. Cuando las tablas de una base de datos van aumentando de tamaño, es decir van teniendo más y más filas, las operaciones que se realizan sobre ellas van siendo cada vez más lentas.

Los índices son unas estructuras gestionadas por el SGBDR para agilizar el acceso a datos y los cálculos. El uso de índices es gestionado automáticamente por el motor del SGBDR, en concreto por el optimizador de consultas, encargado de decidir en cada momento qué índice usar y cómo hacerlo.

Cuando creamos una tabla y definimos columnas como `PRIMARY KEY` o utilizamos la cláusula `FOREIGN KEY` el SGBDR crea de forma automática los índices adecuados. También podemos crear índices de manera explícita utilizando la siguiente sintaxis:

```
CREATE INDEX <índice> ON <tabla>(<columna1>,<columna2>,…)
```

Por **ejemplo**, para crear un índice de la tabla empleado llamado emp_nombre, y basado en la columna nombre del empleado utilizaremos la siguiente sentencia.

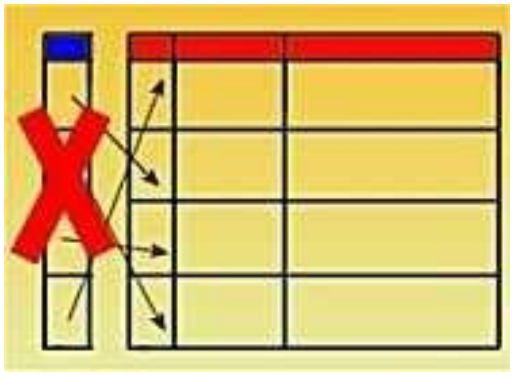
```
CREATE INDEX empleado_nombre ON empleado (nombre)
```

A partir de ese momento el motor de datos del SGBDR usará el índice para agilizar las operaciones sobre la tabla empleado que impliquen a la columna nombre. Hay que hacer notar que en tablas pequeñas el uso de índices puede tener un impacto negativo en el rendimiento.

Creación de índices para agilizar acceso a datos y cálculos…

```
CREATE INDEX
  empleado_nombre ON
  empleado (nombre)
```

Los índices tienen gran importancia para manejar con agilidad tablas con muchas filas. **Una animación te muestra cómo podemos comprobar qué índices existen sobre una determinada tabla.**



Para borrar un índice simplemente usaremos `DROP INDEX <índice> ON <tabla>`.

En el ejemplo anterior.

```
DROP INDEX empleado_nombre ON empleado
```

7. Creación de la base de datos para prácticas y ejercicios.

Vamos a aplicar todo lo anterior a la **creación de la base de datos sobre Gestión de Proyectos** que utilizaremos en algunas prácticas, de manera que realizar el diseño físico de este modelo en ORACLE.

Ejecutarlo como script (con el rayito). Un script SQL es una agrupación de sentencias SQL que pueden ser editadas, almacenadas y recuperadas para hacer más productivo nuestro trabajo.

Es conveniente grabar nuestro script para poder utilizarlo siempre que lo necesitemos sin tener que volverlo a escribir. Observa que los comentarios son útiles y están delimitados entre "**/***" y "***/**", si queremos comentar muchas líneas, si solo queremos comentar 1 línea antecediendo el doble signo menos, esa línea se ignora --. En el TOAD podemos hacerlos de forma automática en el menú EDIT con sus opciones COMMENT BLOCK ó UNCOMMENT BLOCK para descomentar.

Por lo tanto debemos de crearnos el usuario ALUMNO que es el que usaremos a partir de ahora, no entraremos más como SYS, y realizaréis el diseño físico de la gestión de proyectos, que el script lo tenéis hecho en la página 11.

Y a parte como ejercicios realizaremos los siguientes, quiero que se realice en el Ordenador el diseño FISICO de cada uno de estos supuestos (me tendréis que entregar el script de cada supuesto):

Para todos los ejercicios usaremos los siguientes atributos:
ALUMNOS: DNI, NOMBRE Y EDAD
ASIGNATURAS: CODIGO, NOMBRE
MATRICULAR: FECHA
EXAMENES: CODIGO, FECHA, NUMERO_PREGUNTAS
EXAMINAR: NOTAS

1) **Una relación llamada MATRICULAR con tipo de correspondencia 1:1 y cardinalidades (0,1) (0,1) entre ALUMNOS y ASIGNATURA**
2) **Una relación llamada MATRICULAR con tipo de correspondencia 1:1 y cardinalidades (1,1) (1,1) entre ALUMNOS y ASIGNATURA**
3) **Una relación llamada MATRICULAR con tipo de correspondencia 1:M y cardinalidades (0,1) en el lado del 1 entre ALUMNOS y ASIGNATURA (1 Alumno Muchas asignaturas, 1 asignatura 1 alumno)**
4) **Una relación llamada MATRICULAR con tipo de correspondencia 1:M y cardinalidades (1,1) entre ALUMNOS y ASIGNATURA (1 Alumno Muchas asignaturas, 1 asignatura 1 alumno)**
5) **Una relación llamada MATRICULAR con tipo de correspondencia 1:M y entre ALUMNOS y ASIGNATURA con DEBILIDAD de IDENTIDAD de ASIGNATURAS sobre ALUMNOS (1 Alumno Muchas asignaturas, 1 asignatura 1 alumno)**
6) **Una relación llamada MATRICULAR con tipo de correspondencia M:M entre ALUMNOS y ASIGNATURA**
7) **Una relación ternaria M:M:M entre ALUMNOS, ASIGNATURAS y EXAMENES**
8) **Una relación ternaria M:M:1 entre ALUMNOS, ASIGNATURAS y EXAMENES, el lado del 1 a EXAMENES**
9) **Una relación ternaria M:1:1 entre ALUMNOS, ASIGNATURAS y EXAMENES, el lado del M a ALUMNOS**
10) **Una agregación a una relación MATRICULAR M:M entre ALUMNOS y ASIGNATURAS, y esta agregación se une a EXAMENES con otra relación M:M llamada EXAMINAR.**