

Gestión de bases de datos (Parte I)

Introducción a las bases de datos

Competencias

- Conocer el rol de las bases de datos relacionales
- Conocer las características de una RDBMS
- Aprender qué es el lenguaje estructurado de consultas (SQL).

Introducción

Hoy en día, estamos en constante generación de datos, cada compra que hacemos, cada búsqueda en internet, cada vez que tomamos el metro, estamos generando datos. Si estos datos los agrupamos y ordenamos, podemos obtener información de ellos.

Una herramienta que tenemos para ordenar estos datos, son las bases de datos y su importancia se basa en que a través de ellas podemos obtener información.

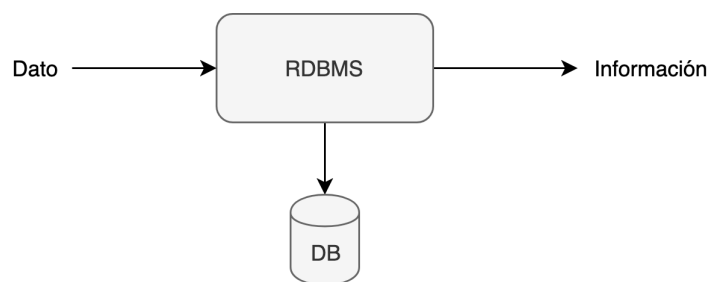


Imagen 1. RDBMS.

Como podemos ver en la imagen, tiene un dato, que luego de pasar por un gestor de bases de datos se obtiene cierta información.

Bases de datos

Las bases de datos se definen como un conjunto de información relacionada que se encuentra ordenada o estructurada. En el ámbito de la informática las bases de datos también son conocidas como sistemas de gestión de bases de datos relaciones o RDBMS (Relational Database Management System), las cuales nos permiten definir y manipular una o más bases de datos.

¿Qué es SQL?

Structured Query Language (Lenguaje estructurado de consultas) es un lenguaje creado para definición y la manipulación de bases de datos relacionales. El beneficio de este lenguaje es que facilita administración de datos almacenados.

Un caso cotidiano:

Imaginen un directorio telefónico con los datos de todas las personas de una ciudad, pero se requiere obtener el número de sólo una de ellas.

¿Qué es lo que hay que hacer para encontrarlo?

Naturalmente, buscar en orden alfabético hasta encontrar el nombre de la persona, lo que puede tomar bastante tiempo. Sin embargo, SQL entrega la facilidad de obtener rápidamente el número deseado.

¿Y qué pasaría si se quisiera obtener los números de todas las mujeres de la ciudad? ¿O de toda la gente que tenga edad entre los 20 y 30 años? ¿O incluso ambas condiciones juntas?

SQL da la posibilidad de hacer este tipo de consultas, reduciendo el tiempo de ejecución en comparación a un humano realizando la misma tarea.

Instalación y configuración de OracleSQL

Competencias

- Conocer las ventajas de utilizar Oracle SQL
- Instalar y configurar Oracle SQL en los distintos sistemas operativos
- Instalar SQL Developer

Introducción

En este capítulo se analizarán las ventajas y desventajas de utilizar las bases de datos proporcionadas por Oracle SQL, y a su vez se dejará el entorno de desarrollo listo para comenzar a trabajar con las bases de datos relacionales.

¿Por qué Oracle SQL?

A lo largo de este módulo se muestra la implementación de SQL mediante Oracle SQL y su entorno de desarrollo Oracle SQL Developer.

Además de Oracle Database, existen otros motores de gestión de bases de datos en el mercado, entre los que podemos mencionar:

- Db2 de IBM.
- SQL Server de Microsoft.
- MySQL de Oracle.
- PostgreSQL.

Oracle Database es un sistema de gestión de base de datos de tipo objeto-relacional (ORDBMS, por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation. Esta herramienta permite analizar datos y efectuar recomendaciones para mejorar el rendimiento y la eficiencia en el manejo de aquellos datos que se encuentran almacenados.

Algunas de las características más relevantes de Oracle SQL son:

- Multiplataforma: Puede ejecutarse en varios sistemas operativos, incluidos Windows Server, Unix y varias distribuciones de GNU / Linux.
- Es el motor de base de datos objeto-relacional más usado a nivel mundial.
- Permite el uso de particiones para hacer consultas, informes, análisis de datos, etc. Esto favorece su eficiencia.

Dentro de las desventajas podemos mencionar principalmente:

- El costo de la licencia en las versiones empresariales (EE, SE).
- Sus limitaciones en la edición express (XE).

Instalación de Oracle SQL

Un aspecto a considerar es que la instalación de Oracle SQL dependerá del sistema operativo que se esté ocupando. Para efectos prácticos, se añade la explicación de los procesos de instalación para Windows y Linux.

En este módulo, se utilizará la versión 11g de Oracle SQL.

Windows

Ir a la página de descarga de Oracle SQL, buscar la versión 11g, seleccionar la arquitectura del sistema operativo (Windows 32/64) y descargar ambas partes.









(11.2.0.1.0)		
Name	Download	Note
Microsoft Windows (32-bit)	 File 1 ,  File 2 (2GB)	See All
Microsoft Windows (x64)	 File 1 ,  File 2 (2GB)	See All
Linux x86	 File 1 ,  File 2 (2GB)	See All
Linux x86-64	 File 1 ,  File 2 (2GB)	See All

Imagen 2. Instalación en Windows.

Se deben extraer los ejecutables en una carpeta y hacer doble clic en el archivo setup.exe para iniciar el proceso de instalación.

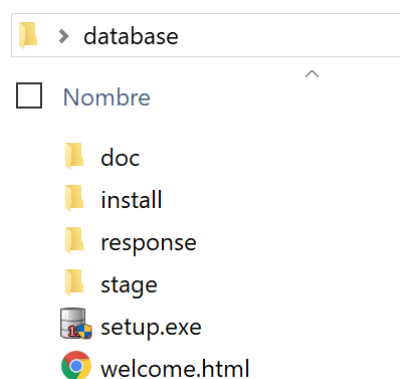


Imagen 3. Extraer ejecutables.

Pasos para la instalación en Windows

- **Paso 1.** El instalador solicita un correo electrónico por si se desea recibir información sobre seguridad y actualizaciones. Podemos ignorarlo haciendo clic en el botón siguiente, sin llenar nada en este cuadro:

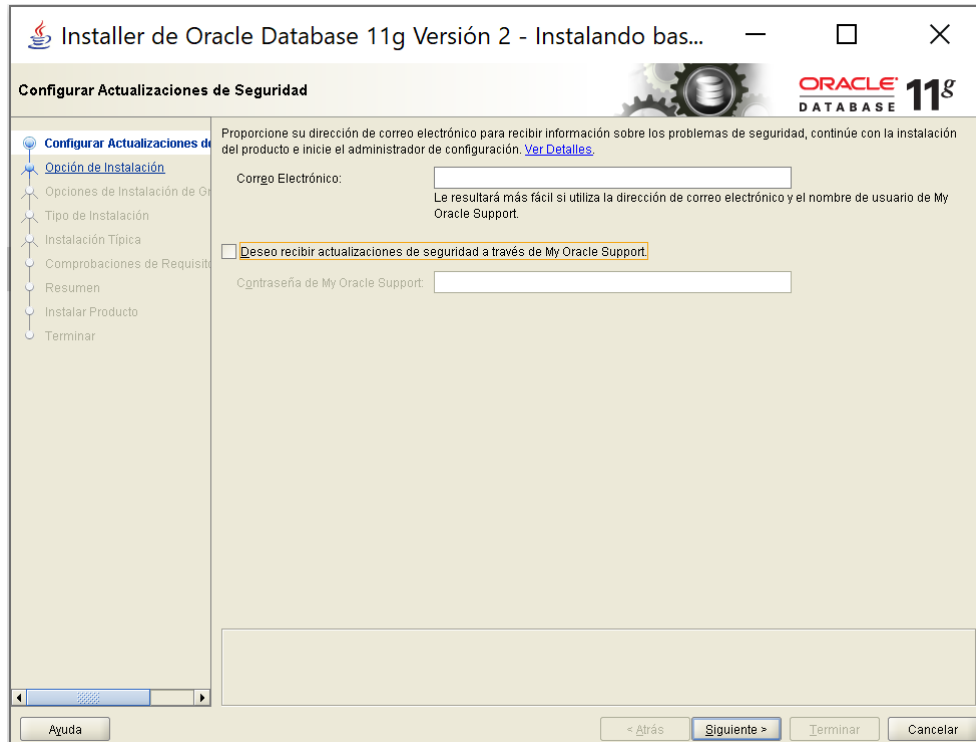


Imagen 4. Instalación en Windows - Paso 1.

- **Paso 2.** El instalador solicita escoger el tipo de instalación que se realizará. En este caso escogeremos la primera opción.

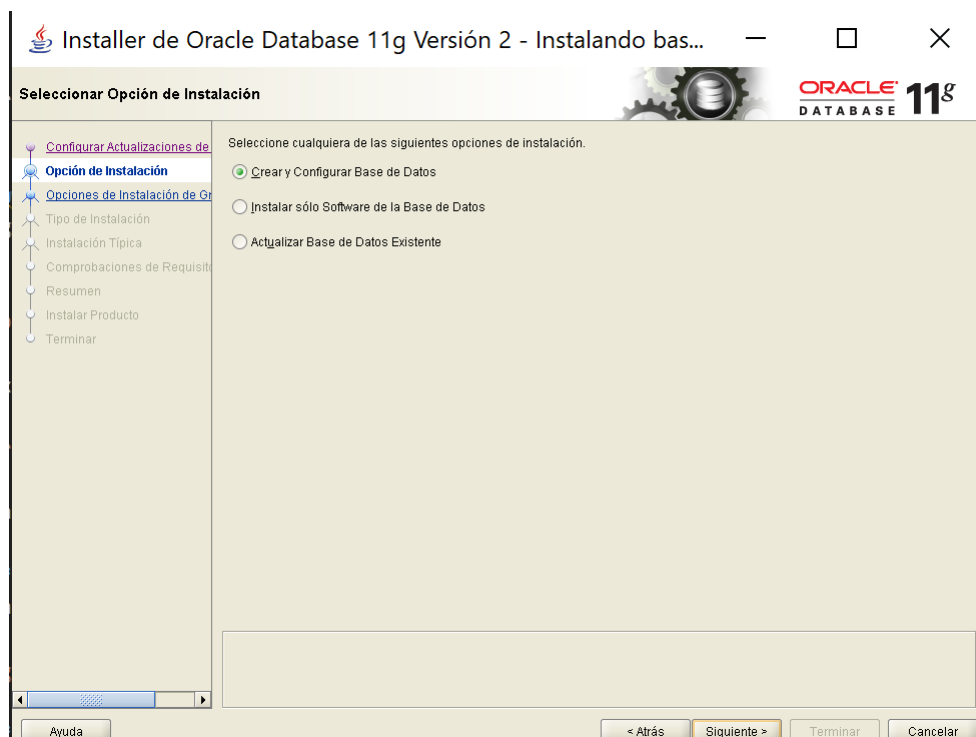


Imagen 5. Instalación en Windows - Paso 2.

- **Paso 3.** El instalador permite elegir la clase de sistema donde se instalará la base de datos. Escogeremos la primera opción: escritorio.

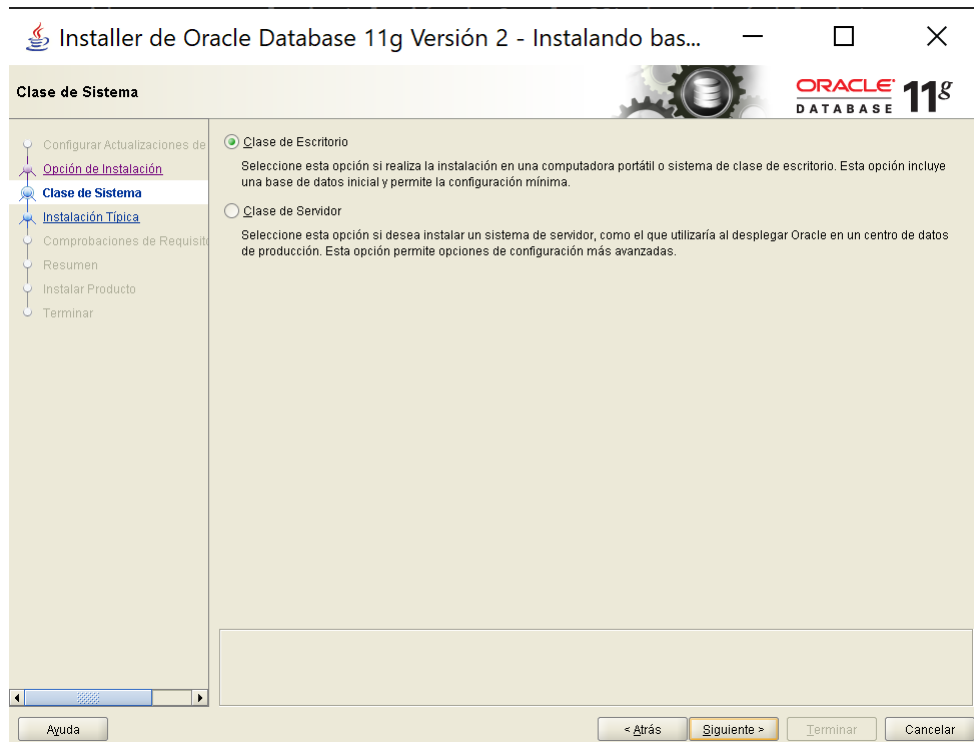


Imagen 6. Instalación en Windows - Paso 3.

- **Paso 4.** Permite escoger la ruta donde queremos instalar Oracle. Dejamos la configuración por defecto y fijamos una contraseña para el administrador.

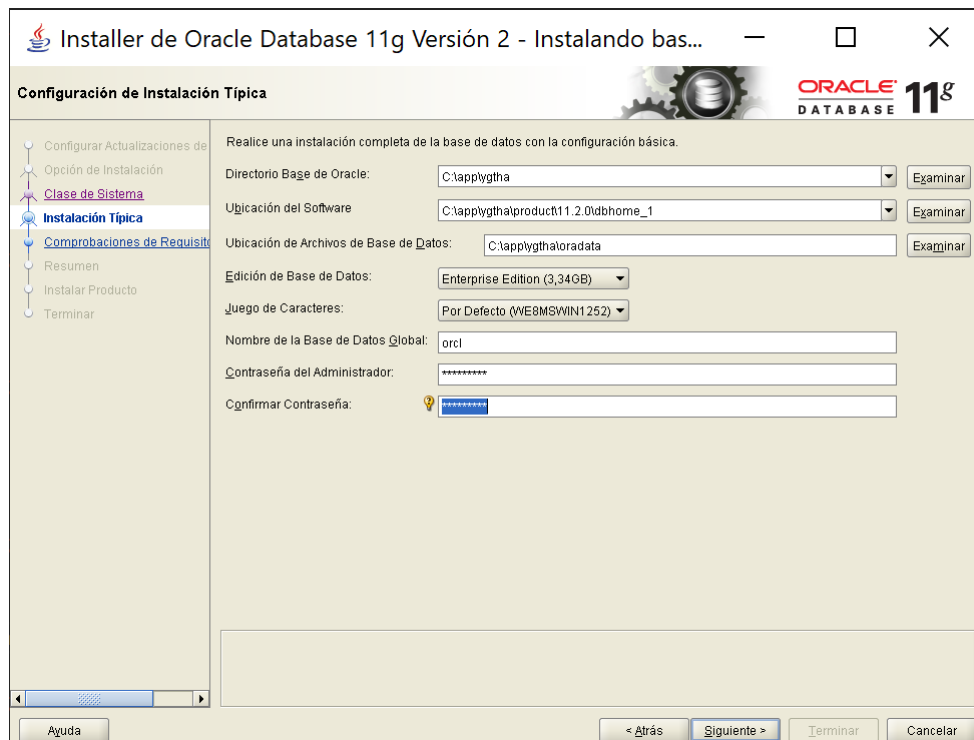


Imagen 7. Instalación en Windows - Paso 4.

- **Paso 5.** Oracle verifica los requisitos y dependencias del software y entrega un resumen de la instalación. Presionamos finalizar.

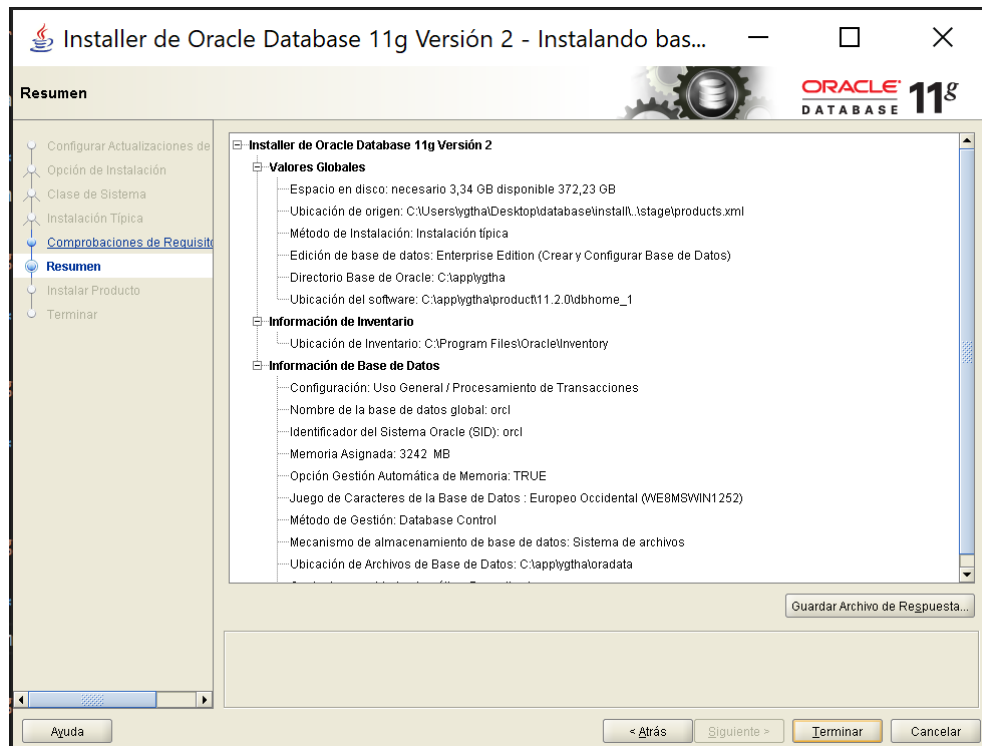


Imagen 8. Instalación en Windows - Paso 5.

- **Paso 6.** Comenzará la instalación y debemos continuar sin realizar ninguna configuración posterior.
- **Paso 7.** Para validar si la instalación se ha realizado correctamente, debemos ejecutar CMD como administrador e ingresamos los siguientes comandos:

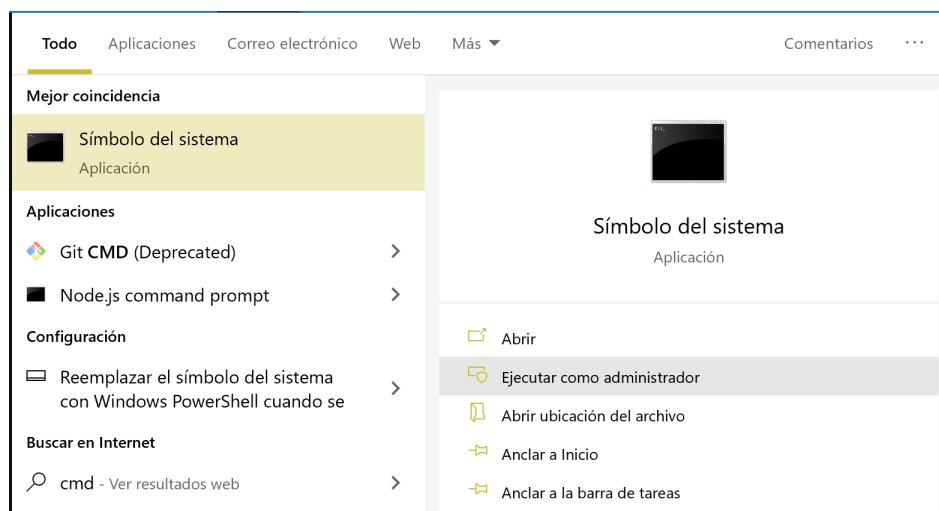
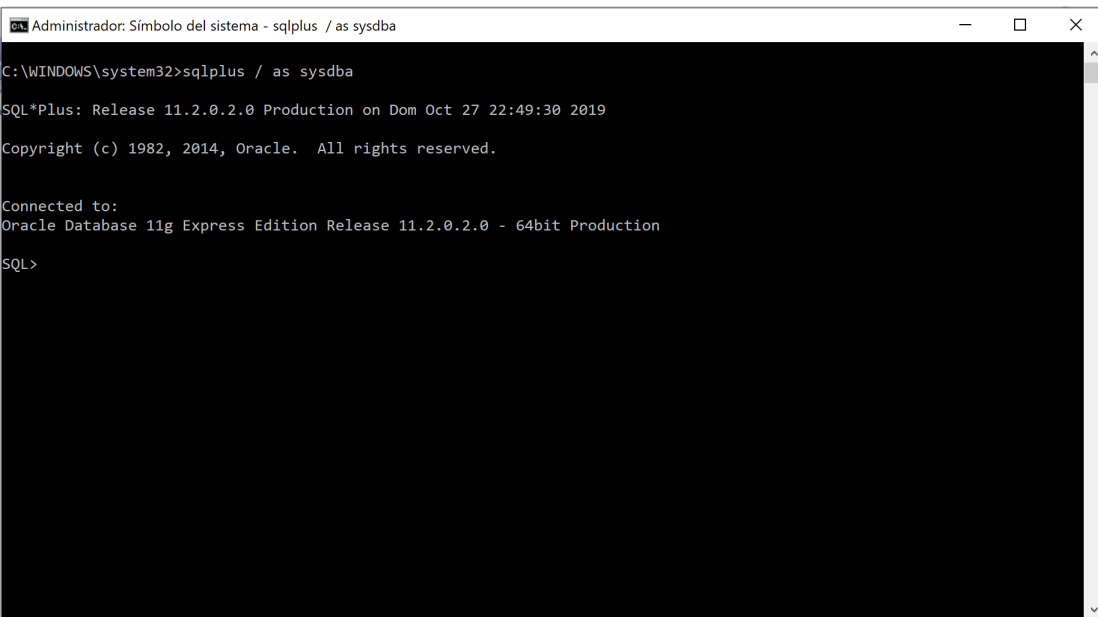


Imagen 9. Instalación en Windows - Paso 7.

```
SET oracle_sid = orcl
```

```
SQLPLUS / AS SYSDBA
```



```
Administrador: Símbolo del sistema - sqlplus / as sysdba
C:\WINDOWS\system32>sqlplus / as sysdba
SQL*Plus: Release 11.2.0.2.0 Production on Dom Oct 27 22:49:30 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
SQL>
```

Imagen 10. CMD.

Pasos para la instalación en Linux

Pasos previos

Se debe descargar la versión Oracle 11g express para Linux.

- **Paso 1.** Copiamos el archivo descargado y lo pegamos en el directorio de inicio.

- **Paso 2.** Descomprimos el archivo usando el comando:

```
unzip oracle-xe-11.2.0-1.0.x86_64.rpm.zip
```

- **Paso 3.** Instalamos los paquetes requeridos usando el comando:

```
sudo apt-get install alien libaio1 unixodbc
```

- **Paso 4.** Ingresamos a la carpeta Disco 1 y convertimos el formato del paquete RPM al formato del paquete DEB (utilizado por Ubuntu) usando el comando:

```
sudo alien --scripts -d oracle-xe-11.2.0-1.0.x86_64.rpm
```

- **Paso 5.** Creamos el script chkconfig requerido:

```
sudo pico /sbin/chkconfigsudo pico /sbin/chkconfig
```

Una vez iniciado el editor de texto, copiamos esto en el archivo y guardamos:

```
#!/bin/bash
# Oracle 11gR2 XE installer chkconfig hack for Ubuntu
file=/etc/init.d/oracle-xe
if [[ ! `tail -n1 $file | grep INIT` ]]; then
    echo >> $file
    echo '### BEGIN INIT INFO' >> $file
    echo '# Provides: OracleXE' >> $file
    echo '# Required-Start: $remote_fs $syslog' >> $file
    echo '# Required-Stop: $remote_fs $syslog' >> $file
    echo '# Default-Start: 2 3 4 5' >> $file
    echo '# Default-Stop: 0 1 6' >> $file
    echo '# Short-Description: Oracle 11g Express Edition' >> $file
    echo '### END INIT INFO' >> $file
fi
update-rc.d oracle-xe defaults 80 01
```

- **Paso 6.** Cambiamos el permiso del archivo chkconfig:

```
sudo chmod 755 /sbin/chkconfig
```

- **Paso 7.** Se establecen los parámetros del kernel (Oracle 11gR2 XE requiere parámetros de kernel adicionales):

```
sudo pico /etc/sysctl.d/60-oracle.conf
```

Una vez abierto el editor, copiamos lo siguiente en el archivo y guardamos:

```
# Oracle 11g XE kernel parameters
fs.file-max=6815744
net.ipv4.ip_local_port_range=9000 65000
kernel.sem=250 32000 100 128
kernel.shmmax=536870912
```

Verificamos el cambio con el comando:

```
sudo cat /etc/sysctl.d/60-oracle.conf
```

Deberíamos ver lo que ingresado anteriormente. Ahora se deben cargar los parámetros del kernel:

```
sudo service procps start
```

Verificamos que los nuevos parámetros se carguen:

```
sudo sysctl -q fs.file-max
```

Deberíamos ver el valor máximo de archivo que ingresado anteriormente.

- **Paso 8.** Configuramos / dev / shm punto de montaje para Oracle. Pasa eso, debemos crear un archivo:

```
sudo pico /etc/rc2.d/S01shm_load
```

Una vez abierto, copiamos lo siguiente en el archivo y guardamos.

```
#!/bin/sh
case "$1" in
start)
    mkdir /var/lock/subsys 2>/dev/null
    touch /var/lock/subsys/listener
    rm /dev/shm 2>/dev/null
    mkdir /dev/shm 2>/dev/null
*)
    echo error
    exit 1
;;
esac
```

- **Paso 9.** Cambiamos los permisos del archivo creado:

```
sudo chmod 755 /etc/rc2.d/S01shm_load
```

Y ejecutamos los siguientes comandos:

```
sudo ln -s /usr/bin/awk /bin/awk
sudo mkdir /var/lock/subsys
sudo touch /var/lock/subsys/listener
```

Finalmente, reiniciamos el sistema.

Pasos para la instalación de Oracle

- **Paso 1.** Instalamos el DBMS de Oracle:

```
sudo dpkg --install oracle-xe_11.2.0-2_amd64.deb
```

- **Paso 2.** Configuramos Oracle usando el comando:

```
sudo /etc/init.d/oracle-xe configure
```

Configuramos las variables de entorno editando su archivo `.bashrc`:

```
pico ~/.bashrc
```

Una vez iniciado el editor, copiamos las siguientes líneas al final del archivo:

```
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/xe
export ORACLE_SID=XE
export NLS_LANG=`$ORACLE_HOME/bin/nls_lang.sh`
export ORACLE_BASE=/u01/app/oracle
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
export PATH=$ORACLE_HOME/bin:$PATH
```

Cargamos los cambios ejecutando el perfil:

```
. ~/.profile
```

- **Paso 3.** Iniciamos el Oracle 11gR2 XE:

```
sudo service oracle-xe start
```

Administrar usuarios y bases de datos

Competencias

- Crear usuarios y asignar permisos
- Crear bases de datos

Introducción

A continuación veremos como administrar usuarios dentro de una base de datos, viendo qué permisos pueden tener al momento de enfrentarse a una base de datos.

Algunas características de Oracle SQL

En primer lugar, debemos dejar claro algunos puntos con respecto a la sintaxis de Oracle SQL:

- Las instrucciones deben ser terminadas con el símbolo " ; " (punto y coma)
- Hay que tener en cuenta que existen nombres restringidos para variables, ya que estos suelen ser comandos y pasa a ser fácil de confundir.

Algunos comandos son:		
ALIAS	AND	AS
CREATE	CREATEDB	CREATEUSER
DATABASE	FROM	INNER
JOIN	LARGE	PASSWORD
WHERE		

- No es sensible a las letras mayúsculas/minúsculas, por lo que se pueden escribir los comandos de cualquiera de estas formas y Oracle SQL lo reconocerá de igual manera.

Administración de usuarios en Oracle SQL

A la hora de trabajar, más de una persona ocupa la información de una misma base de datos. Oracle SQL da la posibilidad de crear usuarios mediante la definición del rol administrador de base de datos, quien tenga este rol será el encargado de asignar roles, los que van a definir el rango de acciones posibles para usuarios ordinarios; así como la creación/modificación/eliminación de tablas, revocación de roles/usuarios por nombrar algunas de las funcionalidades y solo los usuarios registrados van a poder acceder a la base de datos.

Crear usuarios

Primero hay que entrar mediante el terminal a la base de datos con el comando

```
sqlplus / as sysdba
```

Para poder crear un usuario, se debe aplicar el siguiente comando:

```
CREATE USER nombre_usuario IDENTIFIED BY contraseña;
```

Por ejemplo, crearemos el usuario 'usuario_prueba' con la contraseña 'prueba123' de la siguiente manera:

```
CREATE USER usuario_prueba IDENTIFIED BY prueba123;
```

Proporcionar roles

Con nuestra nueva cuenta 'usuario_prueba' creada, ahora podemos comenzar a agregar privilegios a la cuenta utilizando el comando 'GRANT'. GRANT es una declaración con muchas opciones posibles, pero su funcionalidad principal es administrar los privilegios de los usuarios y roles en toda la base de datos.

Le otorgaremos los permisos asociados al inicio de sesión a nuestro usuario:

```
GRANT CONNECT TO usuario_prueba;
```

Asignar privilegios

A continuación, asignaremos los privilegios a nuestro usuario, de la siguiente manera:

```
GRANT CREATE SESSION, GRANT ANY PRIVILEGE TO usuario_prueba;
```

Estos son algunos de los privilegios que se pueden otorgar:

Privilegio	Descripción
SELECT	Permite hacer SELECT sobre la tabla
INSERT	Permite hacer INSERT sobre la tabla
UPDATE	Permite hacer UPDATE sobre la tabla
DELETE	Permite hacer DELETE sobre la tabla
REFERENCES	Otorga privilegios para establecer CONSTRAINT en la tabla
ALTER	Otorga privilegios para hacer ALTER TABLE
INDEX	Otorga privilegios para crear index sobre la tabla
ALL	Otorga todos los privilegios anteriores

Luego, le otorgaremos un espacio en disco para permitirle crear o modificar tablas y datos:

```
GRANT UNLIMITED TABLESPACE TO usuario_prueba;
```

Aprovechando que estamos conectados como 'SYSDBA', activaremos el usuario 'HR' que contiene información de tablas y datos de prueba para hacer los ejercicios:

```
ALTER USER HR IDENTIFIED BY password ACCOUNT UNLOCK;
```

Cambiar de usuario

Ahora que sabemos cómo crear y otorgar privilegios a nuestro usuario, es normal preguntarse cómo se puede acceder a la base de datos con éste. A continuación veremos como como cambiar de usuario.

Primero debemos salir de Oracle SQL con el comando `exit`, e ingresar usando lo siguiente:

```
sqlplus usuario_prueba/prueba123
```

Ahora, para revisar que hemos ingresado de manera correcta, usaremos

```
SELECT user FROM dual;
```

y verán que les aparece el nombre de usuario con el que han ingresado a la base de datos.

```
USER
-----
USUARIO_PRUEBA
```


Herramientas para consultar una base de datos

Competencias

- Conocer herramientas para consultar una base de datos.
- Instalar Oracle SQL Developer.
- Crear una conexión a la Base de datos.

Introducción

Oracle SQL Developer es un entorno de desarrollo gratuito proporcionado por Oracle, que simplifica el desarrollo y la administración de las bases de datos Oracle. Este software nos proporciona una interfaz de trabajo para ejecutar consultas y scripts, herramientas de administración, reportes, modelamiento, entre otros.

Descarga e instalación

Para descargar SQL Developer, vamos a la siguiente [dirección](#) y escogemos nuestro sistema operativo.

En el caso de Windows, escogeremos la versión que incluye el JDK:



Plataforma	Descargar	Notas
Windows de 64 bits con JDK 8 incluido	 Descargar (490 MB)	<ul style="list-style-type: none">• MD5: 8ddbc6663eb774e179b33f702ecff101• SHA1: b1b08c57eb0ba95713a0e42f9ab58d9a6446442f• Notas de instalación
Windows de 32 bits / 64 bits	 Descargar (410 MB)	<ul style="list-style-type: none">• MD5: ec986f454d747b742830284e6cd46fb0• SHA1: f250ec93895f7b3fb4ae240ef32705cc5392e1b1• Notas de instalación• Se requieren JDK 8 u 11

Imagen 11. Descarga de SQL Developer.

Una vez descargado, extraemos los archivos en C:\Archivos de Programa

Este equipo > Disco local (C:) > Archivos de programa > sqldeveloper				
<input type="checkbox"/> Nombre	Fecha de modificación	Tipo	Tamaño	
dropins	07/11/2019 17:48	Carpeta de archivos		
dvt	07/11/2019 17:48	Carpeta de archivos		
equinox	07/11/2019 17:48	Carpeta de archivos		
external	07/11/2019 17:48	Carpeta de archivos		
ide	07/11/2019 17:48	Carpeta de archivos		
javavm	07/11/2019 17:48	Carpeta de archivos		
jdbc	07/11/2019 17:48	Carpeta de archivos		
jdev	07/11/2019 17:48	Carpeta de archivos		
jdk	07/11/2019 17:48	Carpeta de archivos		
jlib	07/11/2019 17:48	Carpeta de archivos		
jviews	07/11/2019 17:48	Carpeta de archivos		
module	07/11/2019 17:48	Carpeta de archivos		
modules	07/11/2019 17:48	Carpeta de archivos		
netbeans	07/11/2019 17:48	Carpeta de archivos		
rdbms	07/11/2019 17:48	Carpeta de archivos		
sleepycat	07/11/2019 17:48	Carpeta de archivos		
sqldeveloper	07/11/2019 17:48	Carpeta de archivos		
sqlj	07/11/2019 17:48	Carpeta de archivos		
svnkit	07/11/2019 17:48	Carpeta de archivos		
icon.png	04/09/2019 22:12	Archivo PNG	2 KB	
sqldeveloper.exe	04/09/2019 22:25	Aplicación	89 KB	
sqldeveloper.sh	04/09/2019 22:12	Shell Script	1 KB	

Imagen 12. Extraer archivos.

Iniciamos el programa con el archivo 'sqldeveloper.exe'.

Se sugiere generar un acceso directo en el escritorio, ya que este programa no lo crea por defecto.



Imagen 13. Iniciar Oracle SQL Developer.

En el caso de Linux, se deberá descargar el 'JDK 8' a través del siguiente [link](#).


Linux RPM	 Descargar (401 MB)	<ul style="list-style-type: none">• MD5: ad7fc15627151461f6a9b2affe8025c9• SHA1: ead77e3ee035deb7116ba0916b2a18081dd403ab• Notas de instalación• Se requieren JDK 8 u 11
-----------	--	---

Imagen 14. Descargar para Linux.

Una vez instalado el JDK, descomprimir el archivo e instalar el SQL Developer con el siguiente comando:

```
rpm -ihv sqldeveloper-xxx.noarch.rpm (reemplazar xxx con la versión exacta del archivo descargado)
```

Una vez instalado nos vamos a la carpeta de SQL Developer y ejecutamos el archivo .sh

```
./sqldeveloper.sh
```

Nos pedirá el path del JDK instalado (por ejemplo usr/java/jdk1.8.0_181). Se iniciará SQL Developer.

Configurar SQL Developer

Para crear una nueva conexión presionaremos en la cruz verde de la esquina superior izquierda.

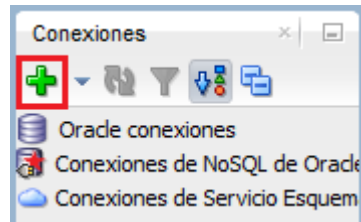


Imagen 15. Crear nueva conexión.

Utilizaremos el usuario HR que configuramos anteriormente, de acuerdo a la siguiente configuración:

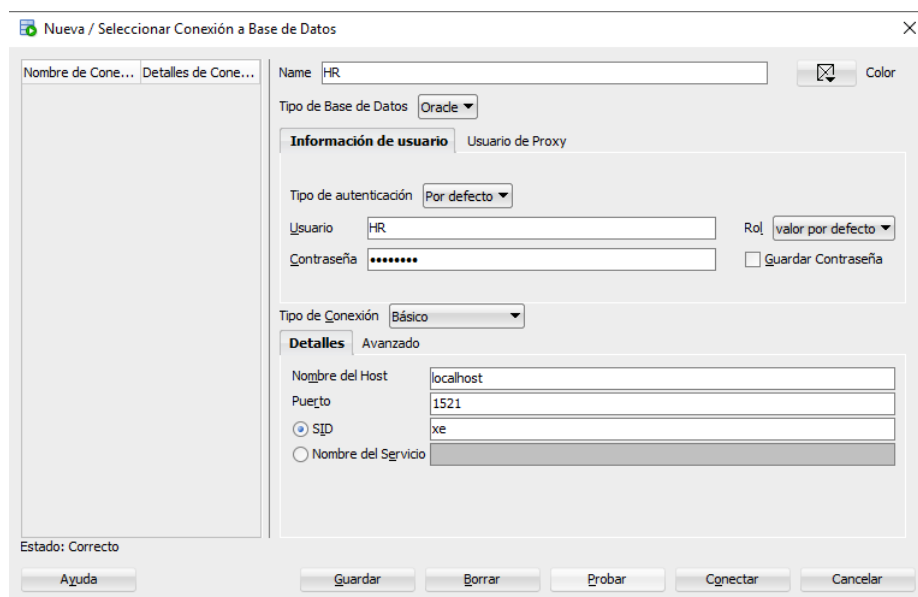


Imagen 16. Configuración.

Antes de guardar la conexión, presionamos probar. Si todo está correcto, guardamos y finalmente presionamos en 'conectar'. Esto abrirá una interfaz de consulta.

Otra forma de acceder a esta interfaz en adelante, será haciendo clic en el icono superior izquierdo:

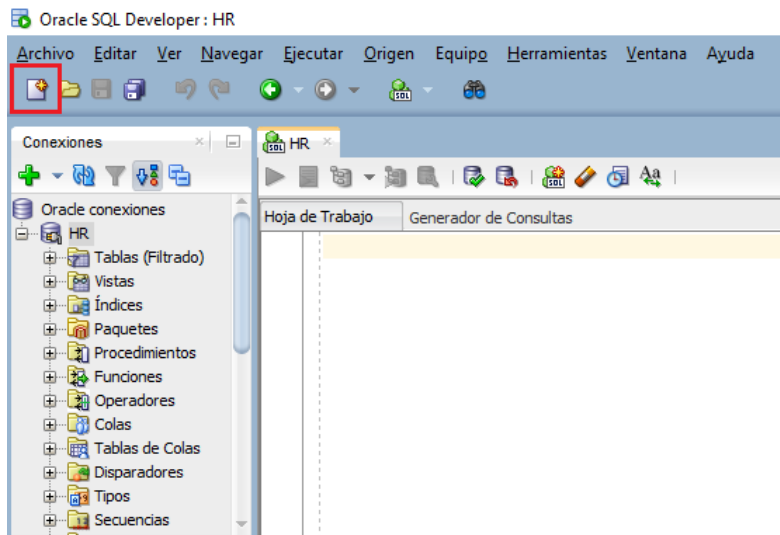


Imagen 17. Acceder a la interfaz.

Haremos una pequeña consulta para probar esta interfaz, escribiendo lo siguiente:

```
SELECT user FROM dual;
```

Para ejecutar la sentencia presionaremos la flecha verde que se observa en la imagen:

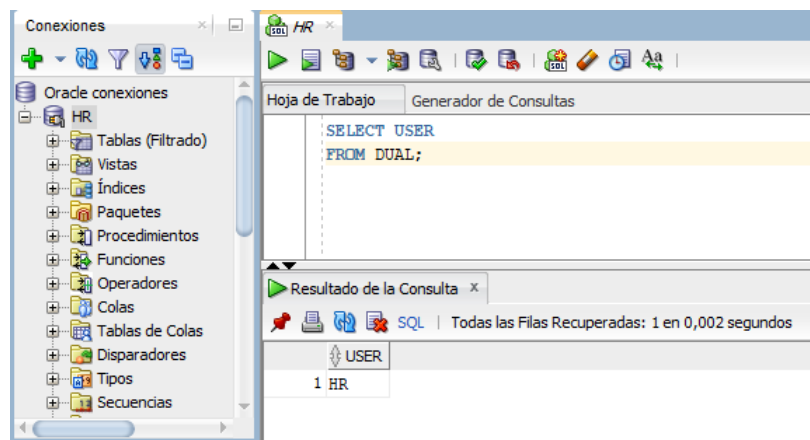


Imagen 18. Ejecutar sentencia.

Elementos de una base de datos

Competencias

- Conocer el concepto de tabla en una base de datos.
- Conocer como se relacionan las tablas mediante claves primarias y foraneas.
- Conocer los tipos de datos que se pueden utilizar.

Introducción

Otro aspecto a considerar es el hecho que existen dos tipos de bases de datos: las Relacionales y las No Relacionales:

- Las bases de datos **Relacionales** son aquellas compuestas por varias tablas donde se almacena la información y posteriormente se relacionan entre sí.
- Las bases de datos **No Relacionales** son aquellas que siguen esquemas más flexibles de organización, donde no necesariamente todas las entradas tienen la misma estructura.

Un aspecto relevante a tomar en cuenta es que para implementar bases de datos relacionales, es necesario tener conocimiento previo sobre qué es lo que vamos a almacenar. Ante la falta de información, el mejor enfoque es implementar una base de datos no relacional.

En esta unidad veremos como operar con bases de dato de tipo relacional.

Continuación Ejemplo: Directorio telefónico

Supongamos que deseamos crear un registro telefónico donde alojaremos el nombre, apellido, número telefónico, dirección y edad de una serie de individuos. Resulta que el registro en sí será la tabla `directorio_telefonico`, donde tendremos los campos `nombre`, `apellido`, `numero_telefonico`, `direccion` y `edad`.

Para empezar a crear nuestra base de datos utilizaremos tablas, donde alojaremos esta información.

Tablas

Una base de datos se compone de múltiples **tablas**. Cada una de éstas presentarán dos dimensiones:

- **filas**, que representan a los registros en la tabla
- **columnas**, que van a representar los atributos ingresados en cada registro, definiendo el tipo de dato a ingresar.

Claves primarias y foráneas

De manera adicional a las filas y columnas, las tablas también cuentan con claves primarias y foráneas. Estas buscan generar identificadores para cada registro de estas tablas mediante algún valor específico de una columna o atributo.

Clave primaria (primary key)

Cuando hacemos referencia a esta columna dentro de su tabla de origen, hablaremos de una clave primaria. Esta clave siempre será de **carácter único**.

Clave foránea (foreign key)

Cuando hacemos referencia a una columna identificadora en otra tabla a la cual hacemos **referencia**, hablamos de una clave foránea.

Ejemplo claves primarias y foráneas

Supongamos que en base a nuestra tabla `directorio_telefonico`, deseamos incorporar información de otra tabla llamada `agenda` que tiene las columnas `nick` y `numero_telefonico`. Ante esta situación, vamos a identificar que la columna `numero_telefonico` en la tabla `directorio_telefonico` será la **clave primaria** y la columna `numero_telefonico` en la tabla `agenda` corresponderá a la **clave foránea**. Este comportamiento es posible dado que el número registrado será congruente en ambas tablas.

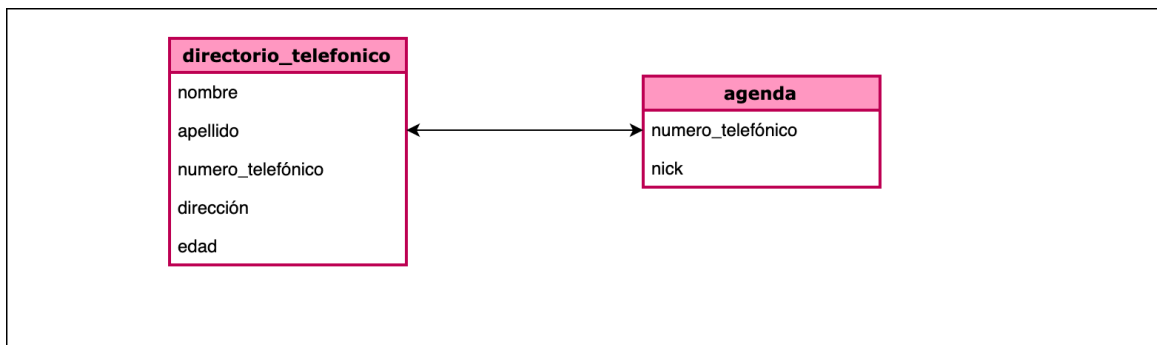


Imagen 19. Ejemplo clave primaria y foránea.

Tipos de datos

Una de las principales características del trabajo con bases de datos, es la necesidad de declarar los distintos tipos de datos existentes en cada campo a completar. Estos aplican restricciones sobre lo que puede ingresar a los registros. No podemos ingresar caracteres cuando piden un número, ni sobrepasar el límite de caracteres posibles.

Los tipos de datos más comunes son:

- `INT` : Números enteros de 4 bytes que pueden tomar valor desde -2147483648 hasta +2147483647.
- `NUMBER(p, s)` : Números decimales de 22 bytes que pueden tener una precisión (p) entre 1 y 38 y una escala (s) entre -84 y 127.
- `CHAR(s)` : Cadena de hasta 2000 bytes de longitud fija.
- `VARCHAR(s)` : Cadena de hasta 4000 bytes de longitud variable. A diferencia de CHAR, si no se ocupa toda la memoria, esta queda libre. CHAR ocupará toda la memoria solicitada.
- `DATE` : Almacena fecha según formato definido en los parámetros de la base de datos. Permite un rango entre el 01-01-4172 AC hasta 31-12-9999 DC.
- `TIMESTAMP` : Almacena fecha y hora juntos: yyyy-mm-dd hh:mm:ss

Para consultar en detalle los tipos de datos, podemos ir a la documentación oficial de Oracle.

Instrucciones de creación, inserción, actualización y eliminación de datos.

Competencias

- Construir bases de datos.
- Crear tablas con sus atributos y tipos de datos correspondientes.
- Crear claves primarias y foráneas.
- Crear relaciones entre tablas.
- Cargar datos en formato SQL.

Introducción

Hasta el momento hemos instalado, configurado y aprendido ciertos comandos de Oracle SQL, y ya es momento que empecemos a crear nuestra primera base de datos con las primeras tablas.

El proceso de vida de una tabla en una base de datos parte con el proceso de Crear, Insertar, Actualizar y Eliminar, que responde a las operaciones elementales que podemos realizar en una tabla.

Realizaremos estos procesos a través del entorno SQL Developer, que nos facilitará de forma visual la creación y administración de nuestros datos.

Creación de tablas

La primera tarea que debemos realizar es la creación de la tabla que permitirá almacenar la información en filas y columnas.

Para crear una tabla dentro de nuestro motor, debemos utilizar el comando `CREATE TABLE` acompañado del nombre de la tabla y declarar los atributos con su respectivo tipo de dato, ingresándolos entre paréntesis. La forma canónica de creación es la siguiente:

```
CREATE TABLE nombre_tabla(  
columna1 tipo_de_dato1,  
columna2 tipo_de_dato2,  
columna3 tipo_de_dato3  
);
```

Volviendo a nuestro ejemplo, vamos a crear la tabla `directorio_telefonico` con los campos `nombre`, `apellido`, `numero_telefonico`, `direccion` y `edad`.

Comentarios

Las líneas precedidas por `--` representan comentarios específicos sobre cada línea del código.

```
-- Creamos una tabla con el nombre directorio_telefonico
CREATE TABLE Directorio_telefonico(
-- Definimos el campo nombre con el tipo de dato cadena con un largo de 25 caracteres.
nombre VARCHAR(25),
-- Definimos el campo apellido con el tipo de dato cadena con un largo de 25 caracteres.
apellido VARCHAR(25),
-- Definimos el campo numero_telefonico con el tipo de dato cadena con un largo de 8 caracteres.
numero_telefonico VARCHAR(8),
-- Definimos el campo dirección con el tipo de dato cadena con un largo de 255 caracteres.
direccion VARCHAR(255),
-- Definimos el campo edad con el tipo de dato entero
edad INT,
-- Definimos que el campo numero_telefonico representará la clave primaria de la tabla.
PRIMARY KEY (numero_telefonico)
);
```

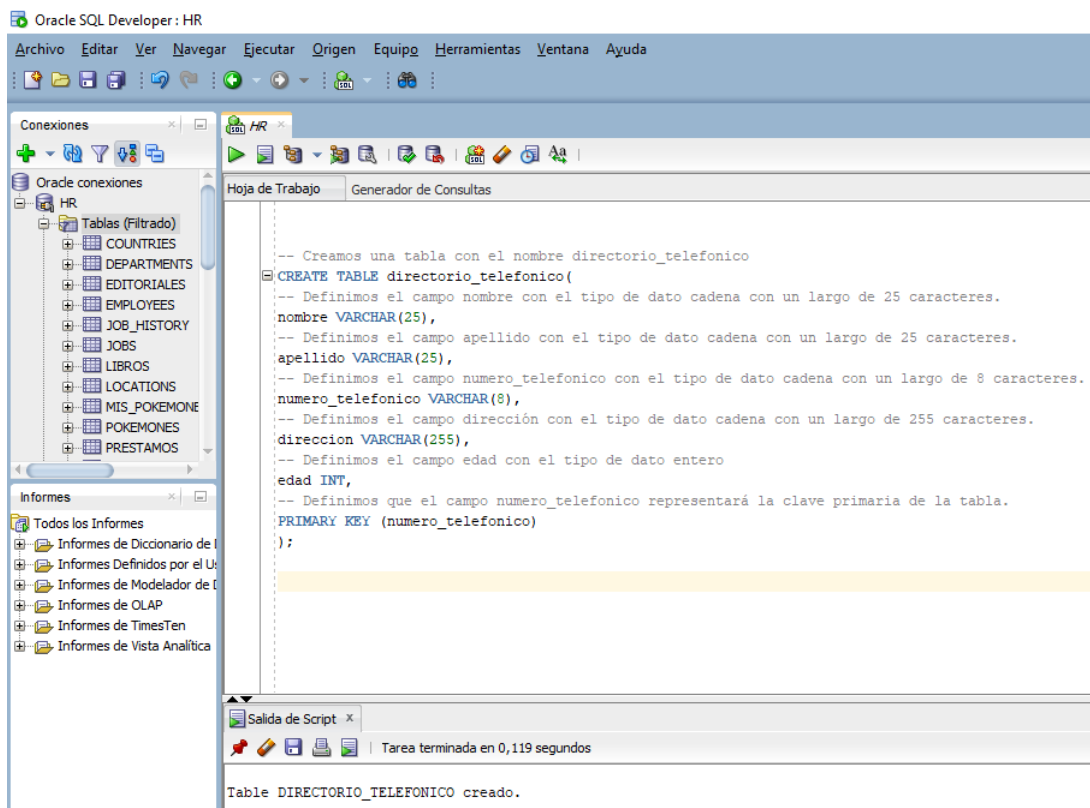


Imagen 20. Comentarios.

Para poder ver la estructura que tiene nuestra tabla, podemos usar el comando

```
SELECT * FROM directorio_telefonico;
```

donde mostrará la siguiente estructura

nombre	apellido	numero_telefonico	direccion	edad

Creando una tabla con clave foráneas

Posterior a la creación de ésta, definiremos los componentes de la segunda tabla `agenda` con el campo `numero_telefonico` y asignaremos una clave foránea proveniente de la tabla `directorio_telefonico`.

Profundizemos sobre la última instrucción. La forma canónica de implementar una clave foránea responde a los siguientes componentes:

```
-- Creamos una tabla con el nombre agenda
CREATE TABLE Agenda(
-- Definimos el campo nick con el tipo de dato cadena con un largo de 25 caracteres
nick VARCHAR(25),
-- Definimos el campo numero_telefonico con el tipo de dato cadena con un largo de 8 caracteres.
numero_telefonico VARCHAR(8),
-- Vinculamos una clave foránea entre nuestra columna numero_telefonico y su simil en la tabla
directorio_telefonico
FOREIGN KEY (numero_telefonico) REFERENCES Directorio_telefonico(numero_telefonico)
);
```

De manera similar, nuestra tabla `Agenda` se ve de la siguiente manera, sin registros de momento.

nick	numero_telefonico

Inserción de datos en una tabla

Para que una base de datos sea útil, como dice su nombre, debe contener datos. Es por esto que existe el comando `INSERT`, que indica la tabla a la que se agregarán datos, los tipos de datos, y los valores que queremos ingresar en el registro.

Esto es un proceso ordenado, y debemos especificar a qué fila pertenecen los datos que estamos ingresando.

Es necesario usar un orden claro, ya que sino la tabla pierde su estructura, haciendo imposible obtener la información buscada y la base de datos perdería su real utilidad.

La forma canónica de la instrucción `INSERT` es la siguiente:

```
INSERT INTO nombre_tabla (columna1, columna2, columna3) VALUES (valor1,
valor2, valor3);
```

Por ejemplo, ingresaremos un registro a las tablas `Directorio_telefonico` y `Agenda`:

```
-- Definimos qué tabla vamos a insertar datos
INSERT INTO Directorio_telefonico
-- Explicitamos cuáles son las columnas a insertar
(nombre, apellido, numero_telefonico, direccion, edad)
-- Con la instrucción VALUES logramos asociada cada columna con un valor específico
VALUES ('Juan', 'Perez', '12345678', 'Villa Pajaritos', 21);
```

Ingreseemos otros registros más:

```
INSERT INTO Directorio_telefonico
(nombre, apellido, numero_telefonico, direccion, edad) VALUES
('Fabian', 'Salas', '32846352', 'Playa Ancha', 21);

INSERT INTO Directorio_telefonico
(nombre, apellido, numero_telefonico, direccion, edad) VALUES
('John', 'Rodriguez', '23764362', 'Constitución', 21);

INSERT INTO Directorio_telefonico
(nombre, apellido, numero_telefonico, direccion, edad) VALUES
('Braulio', 'Fuentes', '23781363', 'Rancagua', 19);
```

Hasta el momento tenemos 4 registros, para poder verlos, se utiliza el siguiente comando, el cque se explicará más adelante:

```
SELECT * FROM Directorio_telefonico;
```

nombre	apellido	numero_telefonico	direccion	edad
Juan	Perez	12345678	Villa Pajaritos	21
Fabian	Salas	32846352	Playa Ancha	21
John	Rodriguez	23764362	Constitución	21
Braulio	Fuentes	23781363	Rancagua	19

¿Qué ocurre si tratamos de ingresar el siguiente registro?

```
INSERT INTO Directorio_telefonico  
(nombre, apellido, numero_telefonico, direccion, edad) VALUES  
('Marta', 'Fuentes', '23781363', 'Santiago', 24);
```

Aparece el siguiente error

```
Error que empieza en la línea: 40 del comando :  
INSERT INTO Directorio_telefonico  
(nombre, apellido, numero_telefonico, direccion, edad) VALUES  
('Marta', 'Fuentes', '23781363', 'Santiago', 24)  
Informe de error -  
ORA-00001: unique constraint (HR.SYS_C007001) violated
```

Donde especifica que se está intentado agregar un valor duplicado, violando la restricción de que una clave primaria debe ser de carácter único.

Ahora, ingresemos un par de registros en en la tabla `Agenda` de la siguiente manera:

```
-- Realicemos el mismo procedimiento en la tabla Agenda  
INSERT INTO Agenda (nick, numero_telefonico) VALUES ('Juanito', '12345678');
```

Recordemos que hicimos una relación entre la tabla `Directorio_telefonico` y `Agenda`, indicando que la clave foránea de la tabla `Agenda` es el `numero_telefonico` de la tabla `Directorio_telefonico`. Entonces, ¿Qué ocurrirá si tratamos de ingresar otro nick con el mismo número?

```
INSERT INTO Agenda (nick, numero_telefonico) VALUES ('Juancho', '12345678');
```

No hay problema, ya que el número telefónico existe en la tabla `directorio_telefonico`. Ahora intentemos agregar un número que no existe en la tabla de `directorio_telefonico`:

```
INSERT INTO Agenda (nick, numero_telefonico) VALUES  
('Fabi', '32846351');
```

y nos aparece el siguiente error:

```
Error que empieza en la línea: 62 del comando :  
INSERT INTO Agenda (nick, numero_telefonico) VALUES  
('Fabi', '32846351')  
Informe de error -  
ORA-02291: integrity constraint (HR.SYS_C007002) violated - parent key not found
```

Especificando que se viola la restricción de la llave foránea, ya que no existe en la tabla de `Directorio_telefonico`.

Ingresemos un par de datos para tener las siguientes registros en las tablas:

nombre	apellido	numero_telefonico	direccion	edad
'Juan'	'Perez'	'12345678'	'Villa Pajaritos'	21
'Fabian'	'Salas'	'32846352'	'Playa Ancha'	21
'John'	'Rodriguez'	'23764362'	'Constitucion'	21
'Braulio'	'Fuentes'	'23781363'	'Rancagua'	19
'Pedro'	'Arriagada'	'38472940'	'Valdivia'	23
'Matias'	'Valenzuela'	'38473623'	'Nogales'	22
'Cristobal'	'Missana'	'43423244'	'Con Con'	20
'Farid'	'Zalaquett'	'32876523'	'La Florida'	20
'Daniel'	'Hebel'	'43683283'	'San Bernardo'	20
'Javiera'	'Arce'	'94367238'	'Quilpue'	20

Agregamos registros en la tabla de `Agenda` , quedando los siguientes valores.

<code>nick</code>	<code>numero_telefonico</code>
'Juanito'	'12345678'
'Juancho'	'12345678'
'Peter'	'38472940'
'Mati'	'38473623'
'Cris'	'43423244'
'Javi'	'94367238'
'Farid'	'32876523'
'Dani'	'43683283'

Actualización de registros

A veces nos vemos en la necesidad de cambiar los registros ya existentes, sin embargo, es riesgoso borrarlos e ingresarlos nuevamente (más adelante se profundizará en esto), por lo que si queremos actualizar los datos de un registro, debemos usar el comando `UPDATE` de la siguiente forma:

```
UPDATE nombre_tabla SET columna1=valor_nuevo WHERE condicion;
```

Juan se cambió de casa a Villa Los Leones, por lo que tenemos que actualizar la tabla `Directorio_telefonico`:

```
UPDATE Directorio_telefonico  
SET direccion='Villa Los Leones'  
WHERE nombre='Juan';
```

nombre	apellido	numero_telefonico	direccion	edad
'Fabian'	'Salas'	'32846352'	'Playa Ancha'	21
'John'	'Rodriguez'	'23764362'	'Constitucion'	21
'Braulio'	'Fuentes'	'23781363'	'Rancagua'	19
'Pedro'	'Arriagada'	'38472940'	'Valdivia'	23
'Matias'	'Valenzuela'	'38473623'	'Nogales'	22
'Cristobal'	'Missana'	'43423244'	'Con Con'	20
'Farid'	'Zalaquett'	'32876523'	'La Florida'	20
'Daniel'	'Hebel'	'43683283'	'San Bernardo'	20
'Javiera'	'Arce'	'94367238'	'Quilpue'	20
'Juan'	'Perez'	'12345678'	'Villa Los Leones'	21

y vemos que la dirección ha sido actualizada.

Eliminación de registros

Cuando estamos completamente seguros de que queremos eliminar un registro de una tabla o incluso la tabla completa, y que de verdad estamos seguro de que la condición que usamos es la correcta, podemos utilizar el comando DELETE. Este lo que hace es eliminar uno, varios o todos los registros de la tabla según la condición dada. La sintaxis para eliminar toda la tabla es:

```
DELETE FROM tabla;
```

Para poder seleccionar que registros queremos borrar debemos hacerlo de la siguiente forma:

```
DELETE FROM tabla WHERE condicion;
```

En nuestro ejemplo eliminaremos a John de la tabla de `directorio_telefonico` de la siguiente forma:

```
DELETE FROM Directorio_telefonico WHERE nombre='John';
```

y en caso que quisieramos borrar todos los datos de una tabla usaremos:

```
DELETE FROM Agenda;
```

nick	numero_telefonico

Agreguemos un registro a la tabla 'Agenda':

```
INSERT INTO agenda  
VALUES('Juanito','12345678');
```

y ahora eliminemos a Juan de la tabla de `Directorio_telefonico`

```
DELETE FROM Directorio_telefonico WHERE nombre='Juan';
```

obtenemos un error recordándonos que no se puede violar la restricción de clave foránea del número telefónico

Error que empieza en la línea: 74 del comando :

```
DELETE FROM Directorio_telefonico WHERE nombre='Juan'
```

Informe de error -

ORA-02292: integrity constraint (HR.SYS_C007002) violated - child record found

Lo pernicioso del método **DELETE**

Si bien el comando DELETE existe para ser usado, debe usarse con mucha precaución, es más, sólo el administrador de la base de datos debería ser capaz de eliminar datos de ella.

Este comando es susceptible a errores, ya que si no implementamos correctamente la condición en el comando WHERE, podemos eliminar datos que no teníamos pensado borrar y no hay posibilidad de recuperarlos.

Oracle SQL les da a todos permisos por defecto, sin embargo, utilizar el comando DELETE no es parte de ellos, el administrador de la base de datos es el encargado de otorgar ese poder.

Añadiendo o eliminado columnas

Utilizando la definición anterior de nuestra tabla `Agenda`, se busca agregar ahora una nota. Ante la eventualidad que deseemos añadir/remover una columna específica de una tabla, podemos implementar la siguiente sintaxis:

```
ALTER TABLE nombre_tabla  
ADD nueva_columna tipo_de_dato;
```

```
-- Declaramos que alteraremos la tabla Agenda  
ALTER TABLE Agenda  
-- Definimos el campo nick con el tipo de dato cadena con un largo de 25 caracteres.  
ADD nota VARCHAR(100);
```

Si deseamos eliminar alguna columna en específico, podemos implementar la instrucción `DROP` de manera análoga a como lo hicimos con `ADD`.

```
ALTER TABLE nombre_tabla  
DROP nueva_columna tipo_de_dato;
```

Restricciones

Puede que existan casos en los que nuestras columnas necesiten reglas que cumplir, como que no hayan valores repetidos o que no existan campos vacíos. Es por eso que aparece el concepto de restricciones. en este listado se muestran las principales con su respectiva definición:

- **NOT NULL:** La columna no puede tener valores `NULL`.
- **UNIQUE:** Todos los valores de la columna deben ser diferentes unos a otros.
- **PRIMARY KEY:** Aplica la clave primaria.
- **FOREIGN KEY:** Aplica la clave foránea.
- **CHECK:** Todos los valores de una columna deben satisfacer una condición en específico.
- **DEFAULT:** Le da un valor por defecto a aquellos registros que no tengan un valor asignado.
- **INDEX:** Sirve para crear y recuperar data de forma rápida.

Estos se aplican de la siguiente forma:

```
-- Creamos una tabla  
CREATE TABLE nombre_tabla(  
-- Declaramos una serie de restricciones a cada campo de dato creado  
columna1 tipo_de_dato1 restriccion,  
columna2 tipo_de_dato2 restriccion,  
columna3 tipo_de_dato3 restriccion  
);
```

PRIMARY KEY y FOREIGN KEY se manejan de forma distinta, los explicaremos en la siguiente sección. Usaremos de ejemplo las mismas tablas que creamos antes, asignándoles restricciones:

Volveremos a crear las tablas `Directorio_telefónico` y `Agenda` :

```
CREATE TABLE Directorio_telefonico(  
  nombre VARCHAR(25) NOT NULL,  
  apellido VARCHAR(25),  
  numero_telefonico VARCHAR(8) UNIQUE,  
  direccion VARCHAR(255),  
  edad INT );
```

```
CREATE TABLE Agenda(  
  nick VARCHAR(25) NOT NULL,  
  numero_telefonico VARCHAR(8) UNIQUE  
);
```

En este caso, hicimos que `numero_telefonico` sea un valor único que no pueda repetirse, que nombre y nick no puedan tener valores NULL.

Restricciones a nivel de PRIMARY KEY y FOREIGN KEY

Como mencionamos antes, las claves primarias sirven para identificar un registro único en una tabla, y la clave foránea sirve para referenciar esa columna desde otra tabla.

La forma de aplicarlas es la siguiente:

```
CREATE TABLE tabla1(  
  columna1 tipo_de_dato1,  
  columna2 tipo_de_dato2,  
  columna3 tipo_de_dato3,  
  PRIMARY KEY (columna1)  
);  
  
CREATE TABLE tabla2(  
  columna4 tipo_de_dato4,  
  columna5 tipo_de_dato5,  
  FOREIGN KEY (columna4) REFERENCES tabla1(columna1)  
);
```

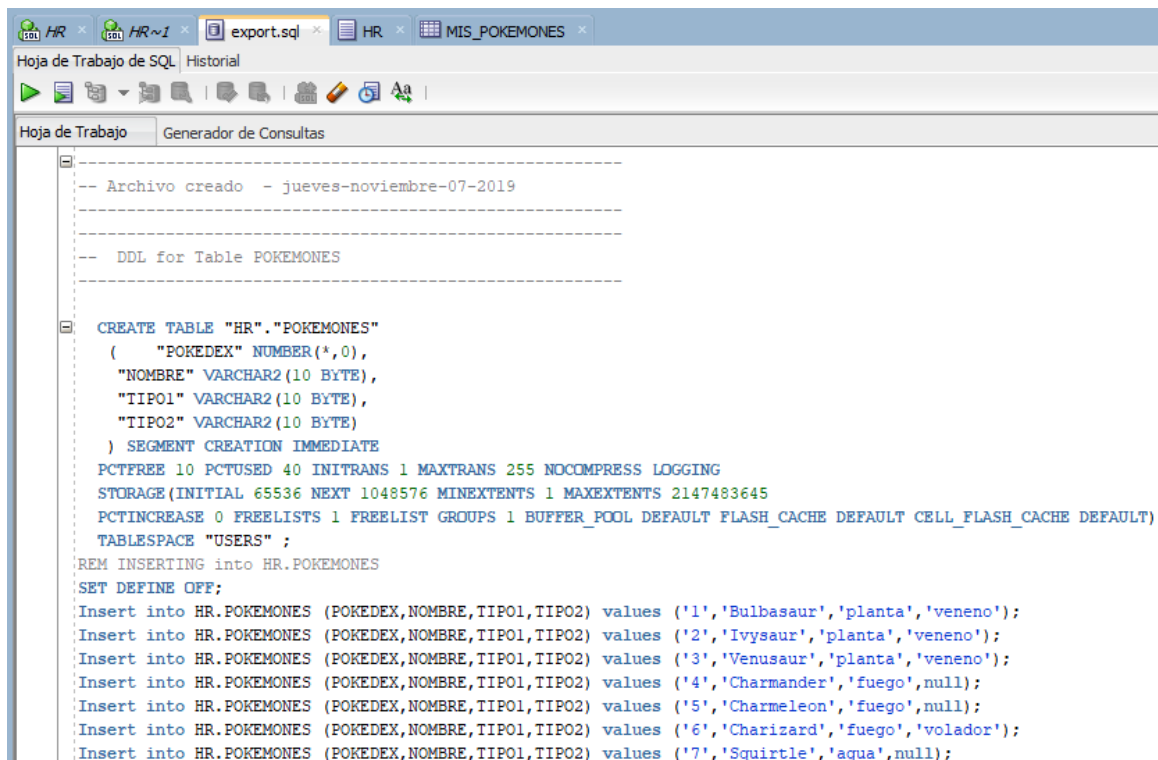
Volvamos al ejemplo de `Directorio_telefonico` y `Agenda` , ambos usan los mismos números telefónicos para referirse a las mismas personas.

La diferencia aquí, es que obtenemos los números desde `Directorio_telefonico` para llevarlos a `Agenda` .

Es por esto, que `numero_telefonico` será clave primaria (PRIMARY KEY) en `Directorio_telefonico` y clave foránea (FOREIGN KEY) en `Agenda`.

Esto se puede ver reflejado así:

```
CREATE TABLE Directorio_telefonico(  
  nombre VARCHAR(25),  
  apellido VARCHAR(25),  
  numero_telefonico VARCHAR(8),  
  direccion VARCHAR(255),  
  edad INT,  
  PRIMARY KEY (numero_telefonico)  
);  
  
CREATE TABLE Agenda(  
  nick VARCHAR(25),  
  numero_telefonico VARCHAR(8),  
  nota VARCHAR(100),  
  FOREIGN KEY (numero_telefonico) REFERENCES  
  Directorio_telefonico(numero_telefonico)  
);
```



The screenshot shows a SQL IDE window with a script editor. The script contains SQL commands for creating a table and inserting data. The table is named 'POKEMONES' and has columns 'POKEDEX', 'NOMBRE', 'TIPO1', and 'TIPO2'. The script also includes a comment about the file creation date and a series of INSERT statements for various Pokémon.

```
-- Archivo creado - jueves-noviembre-07-2019  
  
-- DDL for Table POKEMONES  
  
CREATE TABLE "HR"."POKEMONES"  
(  
  "POKEDEX" NUMBER(*,0),  
  "NOMBRE" VARCHAR2(10 BYTE),  
  "TIPO1" VARCHAR2(10 BYTE),  
  "TIPO2" VARCHAR2(10 BYTE)  
) SEGMENT CREATION IMMEDIATE  
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING  
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645  
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)  
TABLESPACE "USERS" ;  
  
REM INSERTING into HR.POKEMONES  
SET DEFINE OFF;  
Insert into HR.POKEMONES (POKEDEX,NOMBRE,TIPO1,TIPO2) values ('1','Bulbasaur','planta','veneno');  
Insert into HR.POKEMONES (POKEDEX,NOMBRE,TIPO1,TIPO2) values ('2','Ivysaur','planta','veneno');  
Insert into HR.POKEMONES (POKEDEX,NOMBRE,TIPO1,TIPO2) values ('3','Venusaur','planta','veneno');  
Insert into HR.POKEMONES (POKEDEX,NOMBRE,TIPO1,TIPO2) values ('4','Charmander','fuego',null);  
Insert into HR.POKEMONES (POKEDEX,NOMBRE,TIPO1,TIPO2) values ('5','Charmeleon','fuego',null);  
Insert into HR.POKEMONES (POKEDEX,NOMBRE,TIPO1,TIPO2) values ('6','Charizard','fuego','volador');  
Insert into HR.POKEMONES (POKEDEX,NOMBRE,TIPO1,TIPO2) values ('7','Squirtle','agua',null);
```

Imagen 21. Restricciones.