









- ▶ Instrucciones de Actualización
- Diccionario de datos de Oracle









1. LENGUAJE DE DEFINICIÓN DE DATOS

- El lenguaje **SQL** puede utilizarse como **Lenguaje de Definición de Datos** (LDD) o **Data Definición Language** (DDL). El LDD permite:
 - Definir y crear tablas
 - Suprimir tablas
 - Modificar la definición de las tablas
 - Definir tablas virtuales (vistas) de datos
 - Construir *indices* para hacer más rápido el acceso a las tablas
- **II Nota:** la notación utilizada para la especificación de los comandos de SQL es la siguiente:
 - Las palabras clave en se indican en mayúsculas
 - **Los corchetes** [] indican opcionalidad
 - Las llaves {} delimitan alternativas separadas por el símbolo |
 - Los puntos suspensivos (...) indica repetición



1.1 OPERACIONES SOBRE TABLAS

Las tres *operaciones* que pueden realizarse sobre *tablas* mediante el *Lenguaje de Definición de Datos* (LDD) son las de Crear, Eliminar y Modificar tablas.

1.1.1 CREACIÓN DE TABLAS

- Antes de realizar la creación de una tabla es conveniente planificar ciertos aspectos:
 - El nombre de la tabla, que deberá ser un nombre que permita identificar su contenido. Por ejemplo, llamamos a una tabla ALUMNOS porque contendrá datos sobre *alumnos*.
 - El nombre de cada columna de la tabla. Ha de ser un nombre autodescriptivo, que identifique su contenido. Por ejemplo, DNI, NOMBRE o APELLIDOS.
 - El tipo de dato y el tamaño que tendrá cada columna.
 - Las columnas obligatorias, los valores por defecto, las restricciones, etcétera.





Para crear una tabla usamos la orden CREATE TABLE, cuya sintaxis es la siguientes:

```
CREATE TABLE Nombre Tabla (
nombre_atributo tipo_dato [restricción_atributo] ... {
nombre_atributo tipo_dato [restricción_atributo] ... }
| restricción_tabla }] ...
);
```

Donde:

- En primer lugar, en la cláusula CREATE TABLE, especificamos el **nombre de la tabla**.
- En segundo lugar, entre paréntesis, se enumeran todas las columnas de la tabla separarlas con comas (,). Una definición de columna incluye el nombre de la columna seguido de su tipo de datos, por ejemplo, NUMBER, VARCHAR2, y una restricción de atributo tal como NOT NULL, clave primaria, chequeo de valor, etc.
- En tercer lugar, se agregan *restricciones de tabla* si corresponde, por ejemplo, *clave principal*, *clave externa* o *verificación*)





- ➡ Muchos atributos de tablas requieren valores limitados dentro de un rango o el cumplimiento de ciertas condiciones. Con una restricción de verificación de condiciones se puede expresar una condición que ha de cumplirse para todas y cada una de las filas de la tabla.
- La orden CREATE TABLE permite definir distintos tipos de restricciones sobre una tabla: claves primarias, claves ajenas, obligatoriedad, valores por defecto y verificación de condiciones.
- Para definir las **restricciones** en la orden CREATE TABLE usamos la cláusula **CONSTRAINT**. Ésta puede **restringir una sola columna** (restricción de atributo o columna) o puede **restringir un grupo de columnas** de una misma tabla (restricción de tabla).
- Existen dos modos de especificar restricciones:
 - como parte de la definición de columnas (restricción de columna) o
 - En la *parte final de la sentencia* CREATE TABLE, una vez especificadas todas las columnas (*restricción de tabla*).





Restricciones de atributo. Sintaxis

```
[CONSTRAINT nombre_restricción]
{ [NOT] NULL
| DEFAULT valor
| { UNIQUE | PRIMARY KEY }
| REFERENCES nombre_tabla [(nombre_atributo)] [ON DELETE CASCADE]
| CHECK (condición)
}
```

- Las restricciones **DEFAULT** y **[NOT] NULL** se suelen colocar *junto* al atributo, sin asociarle nombre de restricción.
- Tipos de restricciones de atributo
 - NOT NULL. El atributo no permite valores nulos.
 - DEFAULT valor. Permite asignar un valor por defecto en el atributo. Oracle no permite asignar un nombre a esta restricción cuando se utiliza como restricción de atributo.



- PRIMARY KEY. Permite indicar que un atributo forma la *clave* primaria. Se utiliza cuando la clave primaria es simple.
- REFERENCES tabla [(atr)]. Es la manera de indicar que un atributo es *clave ajena* o *externa* y hace referencia a la clave primaria de otra tabla. Opcionalmente se puede poner, entre paréntesis, el nombre del atributo que hace de clave principal.
- UNIQUE. Obliga a que los valores del *atributo tomen valores* únicos (no puede haber dos filas con igual valor en ese atributo). Se implementa creando un índice para dicho atributo.
- CHECK (condición). Establece una condición que deben cumplir los valores introducidos para un atributo.
- La cláusula CONSTRAINT sirve para asignarle un nombre a una restricción





CREATE TABLE nombre_tabla (





```
Columna1 TIPO_DE_DATO
```

[CONSTRAINT nombrerestricción]

[NOT NULL] [UNIQUE] [PRIMARY KEY] [DEFAULT valor]

[REFERENCES Nombretabla [(columna)]

[ON DELETE CASCADE]]

[CHECK (condición)],

Columna2 TIPO_DE_DATO

[CONSTRAINT nombrerestricción]

[NOT NULL] [UNIQUE] [PRIMARY KEY] [DEFAULT valor]

[REFERENCES Nombretabla [(columna)]

[ON DELETE CASCADE]]

[CHECK (condición)],

) [TABLESPACE espacio de tabla];





```
CREATE TABLE CURSO (
 nomCurso varchar2(20) NOT NULL,
 codCurso char(3),
 profesor char(9),
 maxAlum number(2, 0),
fechalnic date,
fechaFin date,
 horas number(3, 0) NOT NULL,
 CONSTRAINT cursosClave PRIMARY KEY (codCurso),
 CONSTRAINT cursosFechasValidas CHECK (fechaFin > fechalnic),
 CONSTRAINT cursosCodigosValidos
              CHECK (codCurso IN ('C01','C02','C03','C04','C05')),
 CONSTRAINT cursosAjena FOREIGN KEY (profesor)
                           REFERENCES PROFESOR(dni),
CONSTRAINT cursosNomUnico UNIQUE (nomCurso)
```





CREATE TABLE EMPLEADO (





COD_PROVINCIA NUMBER(2) REFERENCES PROVINCIAS

ON DELETE CASCADE

);

Restricciones de tabla.

Las restricciones de tabla de la orden CREATE TABLE, que aparecen al final de la definición de las columnas, se diferencian de las de atributo en que pueden hacer referencia a varias columnas en una única restricción (por ejemplo, declarando dos columnas como clave primaria o ajena)







Restricciones de tabla. Sintaxis

- Sirven para definir *restricciones conjuntas sobre una combinación de atributos*.
- Se especifican *después* de haber descrito todos los campos de la tabla.
- Las restricciones más habituales son:
 - **UNIQUE (atr1 [, atr2] ...):** el(los) atributos(s) **NO** pueden contener *valores duplicados conjuntamente*.





- PRIMARY KEY (atr1 [, atr2] ...): el(los) atributos(s) forman la clave primaria. Por lo tanto, tienen valor ÚNICO y NO NULO.
- ▶ FOREIGN KEY (atr1 [, atr2] ...) REFERENCES tabla (atr1 [, atr2] ...): el (los) atributos(s) forman una clave ajena. La cláusula REFERENCES indica la tabla referenciada.
- Si los atributos clave de la tabla referenciada no tienen el mismo nombre que los atributos que forman la clave ajena, deben especificarse los nombres de los atributos clave.
- CHECK (condición): establece una condición que deben cumplir los valores introducidos para un atributo o entre atributos.









La cláusula REFERENCES en las restricciones de atributo y la cláusula FOREIGN KEY en las restricciones de tabla se utilizan cuando existen relaciones entre tablas, y se puede producir un incumplimiento de la restricción de integridad referencial al eliminar tuplas que estén siendo referenciadas.

Si se incumple una restricción de integridad referencial al eliminarse una tupla referenciada, existen tres posibles acciones a realizar:

- ▶ **RESTRICT.** Operación restringida. Es la opción por defecto en Oracle (no hace falta especificarla)
- CASCADE. Operación en cascada que afecta a todas las tuplas asociadas.
- SET NULL. Establece a nulo todas las tuplas asociadas.







CREATE TABLE ASIGNATURA



• • •

CONSTRAINT asig-prof **FOREIGN KEY** (prof) **REFERENCES** PROFESOR (nPr) **ON DELETE SET NULL**);

Esta sentencia indica que si eliminamos un profesor de la tabla PROFESOR, a las tuplas en la tabla ASIGNATURA que hacían referencia a dicho profesor se les asignan un valor nulo en el atributo *prof*.



CREATE TABLE ASIGNATURA (



•••

CONSTRAINT asig-prof **FOREIGN KEY** (prof) **REFERENCES** PROFESOR (nPr) **ON DELETE CASCADE**);

Esta sentencia indica que si eliminamos un profesor, de la tabla PROFESOR, responsable de alguna asignatura, se eliminarán todas las asignaturas que tenga asociadas en la tabla ASIGNATURA.

Cadena de Caracteres





⚠ Restricciones en los atributos del tipo fecha

- Los campos de tipo **fecha** requieren un **tratamiento especial** cuando se emplean en la cláusula CHECK.
- Puesto que las **fechas** se escriben como *cadenas de caracteres*, para poder *comparar* un campo de tipo *fecha* (date) *con* un *valor* concreto, habrá que *convertir la cadena que contiene la fecha a un valor de tipo fecha*. Para ello se utiliza la función de Oracle to_date(). Sin embargo, no será necesario usar esta función cuando se comparen dos campos de tipo fecha.

Ejemplos:

CONSTRAINT nombre CHECK (fechal \$11/10/1999)

CHECK (fecha1 > to_date('11/10/1999','dd/mm/yyyy'))

▶ CONSTRAINT nombre CHECK (fecha1 = fecha2)



1.1.1.1 TIPOS DE DATOS

- Cuando se crea una tabla instrucción CREATE TABLE se debe especificar el *tipo de dato* para cada una de sus *columnas*, estos tipos de datos definen el *dominio de valores* que cada columna puede contener.
- Un campo de tabla podrá almacenar *distintos tipos de datos*: fecha, texto, número y otros tipos. los tipos más usados son **number**, **varchar2**, **date** y **Boolean**.
- Los tipos de datos soportados por Oracle se agrupan en las siguientes categorías:

Cadenas de Caracteres

SQL	TiPO	ORACLE
CHAR(tam)	Almacena caracteres con una de longitud fija	CHAR(tam)
CHARACTER(tam)	Almacena caracteres con una de longitud fija especificada en <i>tam</i> . La longitud máxima es de 2.000 bytes. Ejemplo: CODIGO CHAR(6)	CHAN(tarri)





SQL	TiPO	ORACLE	
VARCHAR(tam)	Almacena cadenas de caracteres de longitud variable , la longitud se expresa en <i>tam</i> . La longitud máxima es de 4.000 caracteres (bytes), y la mínima es 1 byte. Ejemplo: DIRECCION VARCHAR2 (40)	VARCHAR2(tam)	

Numéricos

	SQL	TiPO	ORACLE
	INTEGER		
	INT		NUMBER(0,38)
	SMALLINT	Almacena datos numéricos, tanto <i>enteros</i> como <i>reales</i> , con o sin signo.	
	FLOAT	• Precisión representa el número total de dígitos que va	
-	REAL	 a tener el dato que se define; el rango va de 1 a 38. Escala representa el número de dígitos a la derecha del punto decimal. Ejemplo: SALARIO NUMBER(7,2) define la columna – SALARIO con 7 dígitos, 2 de ellos decimales 	NUMBER (precisión,
	DOUBLE PRECISION		
	DECIMAL(p,e)		
	DEC(p,e)		escala)
	NUMERIC(p,e)		





Cadenas de bits

SQL	TiPO	ORACLE	
BIT(tam)	Almacena datos binarios . Es similar al tipo VARCHAR2, con la diferencia de que maneja cadenas de bytes en lugar de cadenas de caracteres. El tamaño máximo es de 2.000 bytes.	RAW(tam)	
BIT VARYING(tam)	Almacena datos binarios . Se emplea para el almacenamiento de gráficos, sonidos, etc. El tamaño máximo es de 2 gigabytes.	LONG RAW	

Fecha y Hora

SQL	TiPO	ORACLE
DATE	Almacena información de fechas y horas. Para cada tipo DATE se almacena la siguiente información: Siglo/Año/Mes/Día/Hora/Minutos/Segundos. El formato de la fecha se especifica con el parámetro NLS_DATE_FORMAT, que es una cadena de caracteres, del tipo 'DD/MM/YY'. El formato de la fecha se puede cambiar mediante la orden ALTER SESSION y variando el parámetro NLS_DATE_FORMAT. Ejemplo: FECHA DATE.	DATE
TIMESTAMP	Es una extensión del tipo DATE . Se utiliza para guardar fechas con una <i>mayor precisión</i> . Para su definición se utiliza la instrucción TIMESTAMP [(<i>precision</i>)] donde precisión es un valor de 0 a 9 que indica el número de dígitos en la parte fraccional del segundo.	TIMESTAMP(p)





1.1.2 ELIMINACIÓN DE TABLAS

- La orden SQL: **DROP TABLE** *suprime* una tabla de la base de datos.
- Cada usuario *puede borrar sus propias tablas*; sólo el *administrador* de la base de datos o algún *usuario* con el privilegio DROP ANY TABLE puede borrar las tablas de otro usuario.
- Al suprimir una tabla también se suprimen los índices y los privilegios asociados a ella. El formato de la orden DROP TABLE es:



DROP TABLE [usuario].nombretabla [CASCADE CONSTRAINTS];

CASCADE CONSTRAINTS elimina las restricciones de integridad referencial que remitan a la clave primaria de la tabla borrada. Si se omite esta cláusula, la tabla no se borra si existen restricciones sobre sus atributos en otras tablas y Oracle devuelve un mensaje de error.







Supongamos una tabla PROVINCIAS, que tiene definida clave primaria en la columna CODPROVINCIA, y la tabla PERSONAS, que tiene definida una clave ajena (CODPROVIN) referenciando a la tabla PROVINCIAS.

Si intentamos borrar la tabla PROVINCIAS, Oracle nos dará un mensaje de error:





ERROR en línea 1:



ORA-02449: claves únicas/primarias en la tabla referidas por claves ajenas

El error se debe a que existe una restricción de clave ajena en la tabla PERSONAS que referencia a la clave primaria de la tabla PROVINCIAS.

Para poder borrar esta tabla tendremos que usar la opción CASCADE CONSTRAINTS, que suprime todas las restricciones de integridad referencial que se refieran a claves de la tabla borrada:



DROP TABLE PROVINCIAS CASCADE CONSTRAINTS;







1.1.3 ELIMINACIÓN DEL CONTENIDO DE TABLAS

La orden SQL: TRUNCATE permite suprimir todas las filas de una tabla y liberar el espacio ocupado para otros usos sin que desaparezca la definición de la tabla de la base de datos.

Es una orden del lenguaje de definición de datos que no genera información de retroceso (ROLLBACK); es decir, una sentencia TRUNCATE no se puede anular, como tampoco activa disparadores DELETE. Por eso, la eliminación de filas con la orden TRUNCATE es más rápida que con DELETE *.

Su formato es:



TRUNCATE TABLE [usuario].nombretabla [{ DROP | REUSE } STORAGE];

La siguiente sentencia borra todas las filas de la tabla EJEMPLO:







- De forma opcional, TRUNCATE permite liberar el espacio utilizado por las filas suprimidas:
 - Con la opción **DROP STORAGE** se libera todo el espacio, excepto el especificado mediante el parámetro MINEXTENTS de la tabla; se trata de la opción por defecto.
 - Con **REUSE STORAGE** se *mantendrá reservado el espacio* para nuevas filas de la tabla.
 - No se puede truncar una tabla cuya clave primaria sea referenciada por la clave ajena de otra tabla. Antes de truncar la tabla hay que desactivar la restricción.
 Ejemplo: Si intentamos truncar la tabla PROVINCIAS, Oracle nos dará el siguiente mensaje de error:

IJ.

TRUNCATE TABLE PROVINCIAS;



*

ERROR en línea 1:



ORA-02266: claves únicas/primarias en la tabla referidas por claves ajenas activadas







1.1.3 MODIFICACIÓN DE TABLAS

- La definición de una tabla se puede modificar mediante el comando ALTER TABLE (*modificar tabla*), permitiendo cambiar tanto sus *columnas* como *restricciones*.
- Las posibles acciones de modificar tablas serán:
 - la inserción o eliminación de una columna (atributo),
 - la *modificación* de la definición de una *columna*
 - la inserción o eliminación de restricciones de tabla.
 - la activación o desactivación de restricciones de tabla.

1.1.3.1 AÑADIR, MODIFICAR O ELIMINAR COLUMNAS

Add: Se utiliza ADD para añadir columnas a una tabla con un valor NULL inicial para todas las tuplas.



ALTER TABLE nombre_tabla

ADD nuevo_nombre_atributo tipo [NOT NULL]
[CONSTRAINT restricción]





A la hora de añadir una columna a una tabla debemos tener en cuenta varios factores:

Si la columna **NO está definida como NOT NULL**, se le puede añadir en cualquier momento.

Si la columna **está definida como NOT NULL**, la tabla debe estar *vacía* o debe especificarse un *valor por defecto*.

EJ.

En el siguiente ejemplo agregamos el campo "numero" a la tabla "mitabla", de tipo number(3), con el valor por defecto cero y que NO acepta valores nulos:

ALTER TABLE mitabla
ADD numero number(3) default 0 not null;

ALTER TABLE mitabla ADD numero number(3) not null;







ERROR en línea 1:

ORA-01758: la tabla debe estar vacía para agregar la columna (NOT NULL) obligatoria





- Modify: Modifica una o más columnas existentes en la tabla. Al modificar una columna de una tabla se han de tener en cuenta estos aspectos:
 - Si la **columna es NULL** en todas las filas de la tabla, se puede **disminuir la longitud** y **modificar el tipo** de dato.
 - Se puede **aumentar la longitud** de una columna en cualquier momento.
 - Al disminuir la longitud de una columna que tiene datos no se puede asignar un tamaño menor del máximo tamaño del valor almacenado.
 - Es posible aumentar o disminuir el número de posiciones decimales en una columna de tipo NUMBER.
 - Un atributo ya existente sólo se puede hacer NOT NULL si todas las tuplas tienen en ese atributo un valor distinto de NULL.
 - ho se puede cambiar el tipo de dato de un campo que es 'foreign key' o que es referenciado por una 'foreign key', a menos que el cambio no afecte la restricción.







En el siguiente ejemplo modificamos el campo 'precio' de la tabla 'libros' para que tome valores de 6 dígitos incluyendo 2 decimales y no acepte valores nulos:

ALTER TABLE libros
MODIFY precio number(6,2) not null;

Rename Column: Se utiliza para renombrar columnas en tablas.

ALTER TABLE nombre_tabla RENAME COLUMN nombre_antiguo to nombre_nuevo;



Drop Column: Se utiliza para borrar una columna de una tabla. Hay que tener en cuenta que no se pueden borrar todas las columnas de una tabla y tampoco se pueden eliminar claves primarias referenciadas por claves ajenas.





Sintaxis:

ALTER TABLE nombre_tabla

DROP COLUMN nombre_columna [CASCADE CONSTRAINT];

Para borrar varias columnas a la vez omitimos COLUMN:

ALTER TABLE nombre_tabla

DROP (nombre_columna1, nombre_columna2,...);

Eliminamos las columnas SEXO e IMPORTE de la tabla EJEMPLO:











1.1.3.2 AÑADIR Y ELIMINAR RESTRICCIONES

- Podemos *añadir* y *eliminar* las siguientes restricciones de una tabla: CHECK, PRIMARY KEY, NOT NULL, FOREIGN KEY y UNIQUE.
- Para *añadir restricciones* usamos la orden:

ALTER TABLE nombre_tabla
ADD CONSTRAINT nombre_restricción restricción;

Para *eliminar restricciones* usamos la orden:

ALTER TABLE nombre_tabla

DROP CONSTRAINT nombre_restricción [CASCADE];



ALTER TABLE ASIGNATURA DROP CONSTRAINT CKCURSO;



ALTER TABLE ASIGNATURA ADD CONSTRAINT CKCURSO CHECK (curso IN ('1','2','3','4','5'));







ALTER TABLE EMPLEADOS ADD CONSTRAINT APELLIDO_UQ UNIQUE(APELLIDO);



ALTER TABLE EMPLEADOS **ADD CONSTRAINT** PK_EMPLE **PRIMARY KEY(**EMP_NO);



ALTER TABLE EMPLEADOS ADD CONSTRAINT FK_EMPLE FOREIGN KEY (DEPT_NO) REFERENCES DEPART(DEPT_NO) ON DELETE CASCADE;



Por defecto, las restricciones se activan al crearlas, pero pueden desactivarse añadiéndole la cláusula DISABLE al final de la restricción.

El siguiente ejemplo añade una restricción, pero la desactiva:

ALTER TABLE EMPLE ADD CONSTRAINT APELLIDO_UQ UNIQUE(APELLIDO) DISABLE;







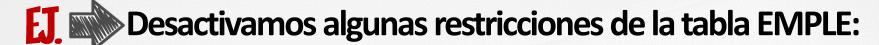
Para *desactivar una restricción* usamos la orden:

ALTER TABLE nombre_tabla

DISABLE CONSTRAINT nombre_restricción [CASCADE];

Para *activar una restricción* usamos la orden:

ALTER TABLE nombre_tabla ENABLE CONSTRAINT nombre_restricción [CASCADE];



- ALTER TABLE EMPLE DISABLE CONSTRAINT PK EMPLE;
- ALTER TABLE EMPLE DISABLE CONSTRAINT FK_EMPLE;





ESQUENA GENERAL:

```
ALTER TABLE nombre tabla
{ADD nuevo nombre atributo tipo [NOT NULL] [CONSTRAINT restricción]
   | MODIFY nombre_atributo tipo_nuevo [CONSTRAINT restricción]
   DROP COLUMN nombre atributo [CASCADE CONSTRAINT]
   DROP (nombre atributo1, nombre atributo2, ...)
   ADD CONSTRAINT nombre restricción restricción
   | { DROP | ENABLE | DISABLE } CONSTRAINT nombre restricción [CASCADE]
```



NO se pueden MODIFICAR restricciones, habría que borrarlas y crearlas



2. INSTRUCCIONES DE ACTUALIZACIÓN EN SQL

Los tres comandos que nos ofrece SQL para actualizar las bases de datos son: **INSERT**, **DELETE** y **UPDATE**.

2.1 EL COMANDO INSERT

- En su forma más simple, sirve para añadir una sola tupla a una tabla. Debemos especificar el nombre de la tabla y una lista de valores para la tupla. Los valores deberán listarse en el mismo orden en que se especificaron los atributos cuando se creó la tabla.
- La **sintaxis** del comando es la siguiente:

```
INSERT INTO nombre_tabla
  [(nombre_atributo [, nombre_atributo] ...)]
  { VALUES (valor [, valor] ...) }
;
```





- [(nombre_atributo [, nombre_atributo] ...)] representa la columna o columnas donde se van a introducir valores. *Si las columnas no se especifican* en la cláusula INSERT, se consideran, por defecto, *todas las columnas de la tabla*.
- (valor [, valor] ...) representa los valores que se van a asignar a las columnas.
 - Éstos se deben corresponder con cada una de las columnas que aparecen; la asociación columna-valor es posicional.
 - Deben coincidir con el *tipo de dato* definido para cada *columna*.
 - Cualquier columna que no se encuentre en la lista de columnas recibirá el valor NULL, siempre y cuando no esté definida como NOT NULL, en cuyo caso INSERT fallará, o su valor por defecto si lo tuviera.
 - Si no se da la lista de columnas, se han de introducir valores en todas las columnas.







Por ejemplo, para añadir una nueva tupla completa a la tabla ORDENADOR, podemos hacer:



INSERT INTO ORDENADOR VALUES ('Ord220', 'Ordenador Multimedia', 'Aula 8');



Si quisiéramos introducir un profesor, pero no conocemos todos sus datos, podemos realizar la siguiente instrucción:

INSERT INTO PROFESOR (nPr, dni, nombre)
VALUES ('30', '29.555.555', 'Antonio Díaz Sotelo');



Inserción con SELECT. Una variación de la instrucción INSERT inserta múltiples tuplas en una relación y la carga con el resultado de una consulta. El formato de INSERT con SELECT es el siguiente:

INSERT INTO NombreTabla1 [(columna [, columna] ...)]
SELECT { columna [, columna] ... | * }
FROM NombreTabla2 [CLÁUSULAS DE SELECT];







Disponemos de la tabla EMPLE30, cuya <u>descripción es la</u> misma que la de la tabla EMPLE. Insertamos los datos de los empleados del departamento 30:

SQL> INSERT INTO EMPLE30

- 2 (EMP_NO, APELLIDO, OFICIO, DIR, FECHA_ALT, SALARIO, COMISION, DEPT_NO)
- **3 SELECT**
- 4 EMP NO, APELLIDO, OFICIO, DIR, FECHA ALT, SALARIO, COMISION, DEPT NO
- **5 FROM** EMPLE
- 6 WHERE DEPT_NO=30;

6 filas creadas.

Como las tablas EMPLE y EMPLE30 <u>tienen la misma</u> <u>descripción</u>, no es preciso especificar las columnas, siempre y cuando queramos dar valores a todas las columnas. La siguiente sentencia generaría el mismo resultado:

SQL> INSERT INTO EMPLE30 **SELECT * FROM** EMPLE **WHERE** DEPT_NO=30;











Insertar un empleado de apellido 'ROMERO', con número de empleado 1112, en la tabla EMPLE. Los restantes datos del nuevo empleado serán los mismos que los de 'ARROYO' y la fecha de alta será la fecha actual del sistema:



- 2 SELECT 1112, 'ROMERO', OFICIO, DIR, SYSDATE, SALARIO,
- 3 COMISION, DEPT NO
- 4 FROM EMPLE WHERE APELLIDO='ARROYO';

1 fila creada.



Las columnas cuyos valores desconocemos (OFICIO, DIR, SALARIO, COMISION, DEPT_NO) son las que devolverá la sentencia SELECT; en el resto de las columnas ponemos directamente sus valores.





2.2 EL COMANDO DELETE

- Para eliminar una o varias tuplas de una tabla se usa la orden SQL **DELETE**.
- La **sintaxis** del comando es la siguiente:

DELETE FROM nombre_tabla [WHERE condición];



Está formada por una cláusula WHERE, similar a la de las consultas SQL, para seleccionar las tuplas que se van a eliminar. La *omisión* de la cláusula WHERE indica que *deben eliminarse todas las tuplas* de la tabla.

Borrar de la tabla ORDENADOR todos los ordenadores que sean estaciones SUN.

DELETE FROM ORDENADOR **WHERE** tipo = 'Estación Sun';



2.3 EL COMANDO UPDATE

Para actualizar los valores de las columnas de una o varias filas de una tabla utilizamos la orden UPDATE, cuyo formato es el siguiente:

```
UPDATE Nombre Tabla

SET {nombre_atr = valor [, nombre_atr = valor, ...] | nombre_atr [, nombre_atr, ...] = (subconsulta) [WHERE condición];
```

Donde:

- Nombre Tabla: nombre de la tabla cuyos valores de columnas se van a actualizar.
- **SET:** pares *columna-valor* que indican los valores que se les asociarán a cada columna en la actualización.
- **WHERE:** selecciona las filas que se van a actualizar. Si se omite, la actualización afectará a todas las filas de la tabla.







Sea la tabla CENTROS, actualizar la dirección del centro con COD_CENTRO 22 a 'C/Galaroza 13' y el número de plazas a 295:

UPDATE CENTROS

SET DIRECCION = 'C/ Galaroza 13', NUM_PLAZAS = 295 WHERE COD_CENTRO = 22;

2.3.1 EL COMANDO UPDATE CON SELECT

- Podemos incluir una **subconsulta** en una sentencia UPDATE que puede estar contenida en la cláusula **WHERE** o puede formar parte de **SET**.
- Cuando la subconsulta forma parte de SET, debe seleccionar una única fila y el mismo número de columnas (con tipos de datos adecuados) que las que hay entre paréntesis al lado de SET. Los formatos son:



UPDATE NombreTabla

SET { nombre_atr = valor [, nombre_atr = valor, ...] WHERE nombre_atr = (SELECT ...)







UPDATE NombreTabla

SET (columna1, columna2, ...) = (SELECT col1, col2, ...)

WHERE condición;





UPDATE NombreTabla

SET columna1 = (SELECT col1 ...), columna2 = (SELECT col2 ...)

WHERE condición;



Si quisiéramos actualizar el precio del producto de id 1 al valor máximo de todos los productos incrementados en un 20%:



UPDATE PRODUCTOS

SET precio = (SELECT MAX(precio) * 1.2

FROM PRODUCTOS)

WHERE product_id = 1;









En la tabla CENTROS, se desea igualar la dirección y el número de plazas del código de centro 10 a los valores de las columnas correspondientes que están almacenadas para el código de centro 50.





3. DICCIONARIO DE DATOS

- El diccionario de datos es una *guía en la que se describe la BD* y los *objetos* que la forman.
- Toda la información de las tablas creadas con el comando CREATE TABLE queda registrada en el diccionario de datos de Oracle.
- En una BD relacional, el diccionario de datos está formado por tablas y vistas que pueden ser consultadas por los usuarios proporcionando información acerca de:
 - La estructura lógica y física de la BD.
 - Las definiciones de todos los objetos de la BD: tablas, vistas, índices, disparadores, procedimientos, funciones, etcétera.
 - Los valores por defecto de las columnas de las tablas.
 - Información acerca de las restricciones de integridad.
 - Los privilegios y roles otorgados a los usuarios.
 - ₽ etc...



- Existen varias categorías de vistas del diccionario:
 - Ustas con el prefijo DBA: muestran una vista global de toda la base de datos sólo accesible para los administradores.
 - Vistas con el prefijo ALL: pueden ser consultadas por todos los usuarios y ofrecen información sobre todos los objetos del sistema.
 - ➡ Vistas con el prefijo USER: ofrecen información de los objetos propios de un usuario.
- Los objetos del diccionario de datos a los que un usuario puede acceder se encuentran en la *vista* DICTIONARY, que es propiedad del usuario SYS, creado por Oracle para realizar las tareas de administración de la base de datos. Con la orden :

SQL> SELECT TABLE_NAME FROM DICTIONARY;



se visualizan los objetos del diccionario de datos a los que se puede acceder.





El prefijo de una vista del diccionario indica el nivel de acceso a él: Vistas USER y ALL accesibles para todos los usuarios y vistas DBA sólo accesibles para administradores.

VISTAS PEL PICCIONARIO PE PATOS:

PATOS	VISTAS	
Información de tablas y otros objetos	USER_TABLES, USER_OBJECTS, USER_CATALOG	
Información de restricciones	USER_CONSTRAINTS, ALL_CONSTRAINTS, DBA_CONSTRAINTS USER_CONS_COLUMNS, ALL_CONS_COLUMNS, DBA_CONS_COLUMNS	
Información sobre vistas	USER_VIEWS, ALL_VIEWS	
Información sobre sinónimos	USER_SYNONYMS, ALL_SYNONYMS	





VISTAS DEL DICCIONARIO DE DATOS

DESCRIBE. El comando describe nombre_tabla permite ver la definición de una tabla concreta de la base de datos. En esta descripción aparecerán los nombres de los atributos, el tipo de datos de cada atributo y si tiene o no permitidos valores nulos.

Sintaxis:

{ DESC | DESCRIBE } nombre tabla





JULY DESC DEFAINT		
Nombre	¿Nulo?	Tipo

DEPT NO **DNOMBRE**

SOI > DESC DEDART

LOC

NOT NULL NUMBER(2)

VARCHAR2(14)

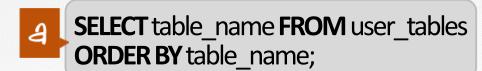
VARCHAR2(14)



- **USER_TABLES**. Los usuarios pueden consultar las tablas creadas por medio de la vista USER_TABLES.
 - Esta vista contiene información acerca de las tablas: nombre de la tabla, nombre del tablespace, número de filas, información de almacenamiento, etcétera.
 - **⇔**Sintaxis:

SELECT TABLE_NAME
FROM { USER_TABLES | ALL_TABLES | DBA_TABLES }
[ORDER BY table_name];







SELECT table_name **FROM** all_tables **ORDER BY** table name;



SELECT table_name **FROM** dba tables;



SELECT * FROM all_tables
WHERE OWNER = 'OT'
ORDER BY table name;



Si el usuario no es *administrador* (DBA), se producirá el siguiente error:

ORA-00942: table or view does not exist



- □ USER_CONSTRAINTS. Existen una serie de vistas creadas por Oracle que contienen información referente a las restricciones definidas por los usuarios en las tablas. Contienen la siguiente información general :
 - USER_CONSTRAINTS: definiciones de *restricciones de tablas propiedad del usuario*.
 - ALL_CONSTRAINTS: definiciones de restricciones sobre tablas a las que <u>puede acceder</u> el usuario.
 - DBA_CONSTRAINTS: todas las definiciones de *restricciones* sobre todas las tablas.

COLUMNAS DE LA TABLA USER_CONSTRAINTS



SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, DELETE_RULE, INVALID **FROM** USER_CONSTRAINTS

WHERE TABLE_NAME='ALUMNO';

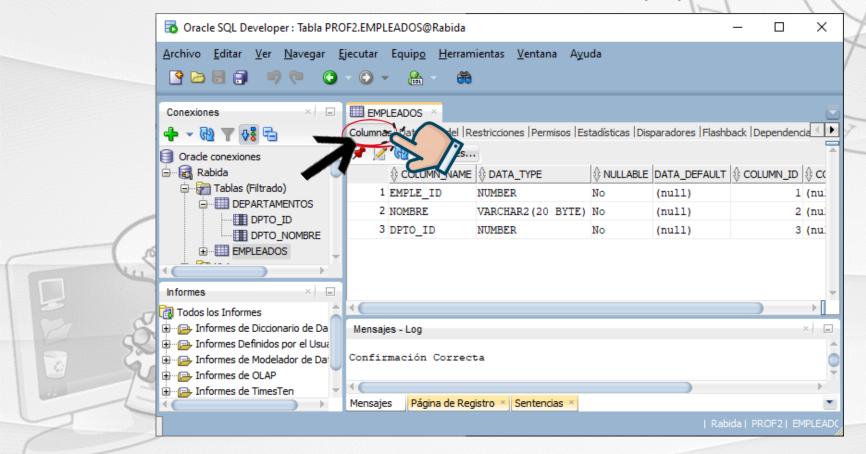






PESTAÑAS:

& Columnas: Información de los atributos y tipos



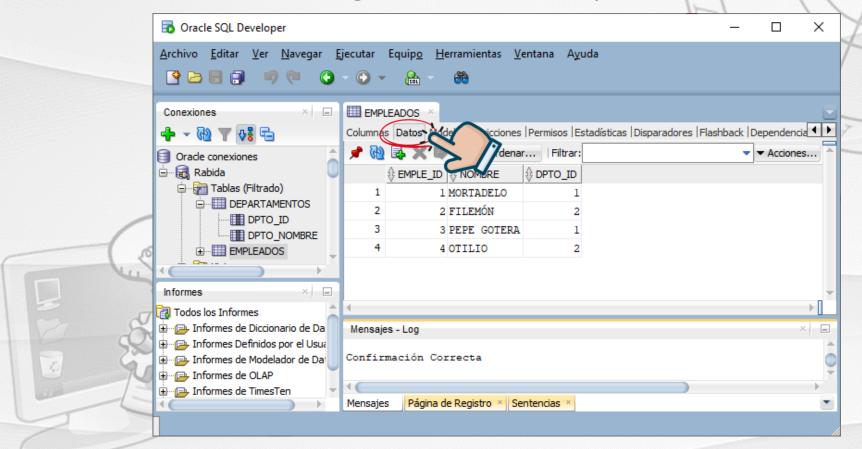






PESTAÑAS:

Batos: Datos cargados en la tabla (tuplas)



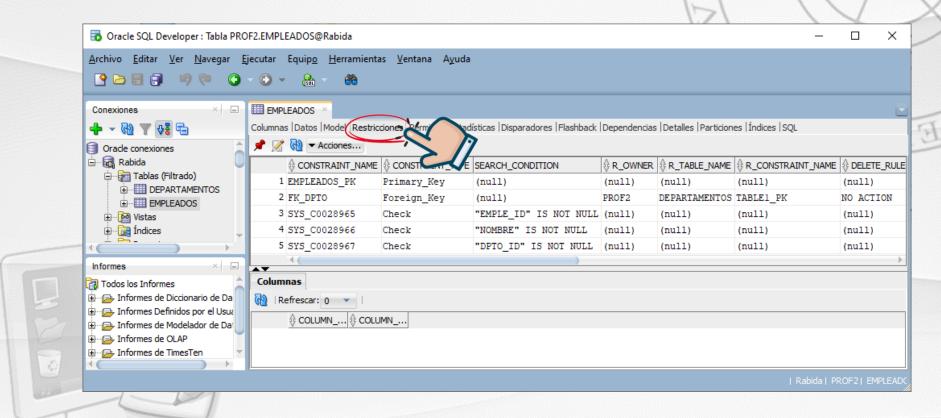






PESTAÑAS:

Restricciones: Constrainsts (restricciones) definidas sobre la tabla









PESTAÑAS:

A Modelo: Modelo Relacional de la Base de Datos

