Ud 2.
Base de Datos Relacionales

SQL : Oracle

IES Maestre de Cva.
Isabel Guerrero

Objetivos

- Definir las estructuras físicas de almacenamiento
- Crear tablas
- Seleccionar tipos de datos adecuados
- Definir campos claves en las tablas
- Implantar las restricciones establecidas en el diseño lógico
- Verificación mediante conjuntos de pruebas
- Uso de asistentes y herramientas gráficas
- Uso del lenguaje de definición de datos (DDL)
- Definir y documentar el diccionario de datos

Pag. :2

Contenidos

- Herramientas gráficas y de texto proporcionadas por los SGBD
- El lenguaje de definición de datos(LDD)
- Creación, modificación y eliminación de BBDD
- Creación, modificación y eliminación de tablas
- Implementación de restricciones

Pag. :3

Introducción a SQL

- **SQL**: es un lenguaje de consulta estructurado utilizado por los sistemas gestores de bases de datos para ORGANIZAR, GESTIONAR y RECUPERAR datos.
- Es un lenguaje para todo tipo de usuario: administradores, desarrolladores y usuarios normales.
- Se puede utilizar de forma interactiva (el usuario escribe órdenes desde el teclado y al instante obtiene el resultado) y de forma embebida (mezclando instrucciones propias del lenguaje de programación (C, PL/QL,PHP, ..) con sentencias SQL para acceder a SGBD) o en entornos visuales.

Pag. :4

Sublenguajes de SQL

- **LENGUAJE DE DEFINICIÓN DE DATOS (DDL o LDD)**

Sentencias que permiten definir la estructura y organización de los datos.

- **LENGUAJE DE MANIPULACIÓN DE DATOS (DML o LMD)**

Sentencias que permiten añadir, modificar y suprimir datos de la BD.

- **LENGUAJE DE CONTROL DE DATOS (LCD o DCL)**

Sentencias sobre el acceso y utilización de los datos.

Pag. :5

Tipos de sentencias SQL: Lenguaje Manipulación de datos (DML)

- **SELECT:** Recupera datos de la BD
- **INSERT:** Añade nuevas filas a una tabla de la BD
- **DELETE:** Borra filas a una tabla de la BD.
- **UPDATE**: Modifica filas de una tabla de la BD

Pag. :6

Tipos de sentencias SQL: Lenguaje Definición de Datos (DDL):

- **CREATE TABLE:** Crea una nueva tabla
- **DROP TABLE:** Suprime una tabla de la BD
- **ALTER TABLE**: Modifica una tabla de la BD
- **CREATE VIEW:** Crea una nueva vista en la BD
- **DROP VIEW**: Suprime una vista de la BD
- **CREATE INDEX:** Crea un índice para una serie de columnas
- **DROP INDEX:** Borra un índice
- **CREATE SYNONYM:** Crea un alias para una tabla
- **DROP SYNONYM:** Borra un sinónimo

Pag. :7

Tipos de sentencias SQL: Lenguaje de Control de Datos(LCC)

❖ **Control de Acceso:**

- **GRANT:** Concede privilegios de acceso a usuarios
- **REVOKE**: Suprime privilegios de acceso a usuarios

❖ **Control de transacciones:**

- **COMMIT:** Finaliza la transacción actual
- **ROLLBACK:** Suprime la transacción actual

Pag. :8

Notación para la sintaxis del lenguaje

- **Palabras en mayúsculas**. Estas son las palabras reservadas del lenguaje. Por ejemplo SELECT, DROP, CREATE.
- **Palabras** en **minúscula**: especifica sintaxis mas en detalle o elementos que cambian
- **Corchetes**: Opcional
- **Llaves**: opciones alternativas
- **Puntos suspensivos:** repetición

  SELECT columna [,columna] ... FROM tabla

  CREATE {DATABASE |SCHEMA} nombre

Pag. :9

El lenguaje de definición de datos (LDD)

- Las funciones de este sublenguaje son:
  - Crear tablas, índices y otros objetos de la base de datos (como vistas, sinónimos, etc.)
  - Definir las estructuras físicas donde se almacenarán los objetos de las bases de datos (espacios de tablas (tablespaces), ficheros de datos (datafiles), etc.)

Pag. :10

LDD: Instrucciones Básicas

- **CREATE tipo_objeto Nombre Definición.**
  - Crea un objeto de un determinado tipo (DATABASE, TABLE, INDEX, etc.) con un nombre (por ejemplo Actores o Personajes) y una definición (CodigoPersonaje, Nombre, etc.).
- **DROP tipo_objeto Nombre.**
  - Elimina un tipo de objeto especificado mediante un nombre. Por ejemplo, la sentencia DROP TABLE Actores, borraría de la base de datos la tabla Actores junto con todos sus datos.
- **ALTER tipo_objeto Nombre Modificación.**
  - Modifica la definición de un objeto. Por ejemplo, la sentencia ALTER TABLE Actores DROP COLUMN Fecha, eliminaría la columna Fecha de la tabla Actores.

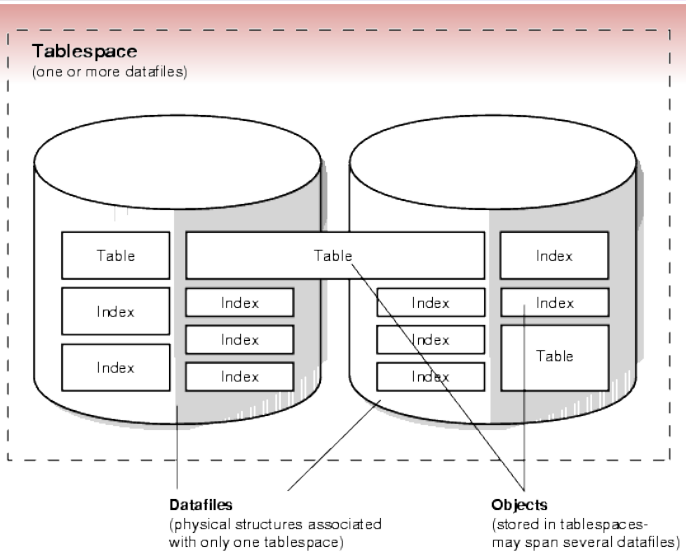
Pag. :11

Oracle: Introducción a la organización física

- Una Base de Datos está formada por un conjunto de archivos de datos, donde se encuentran almacenadas las tablas.
- Estos archivos están agrupados formando **TABLESPACES**.
- Un TABLESPACE es la unidad lógica de almacenamiento de datos representada físicamente por uno o más archivos de datos.
- Se recomienda crear distintos tablespaces para las tablas de distintas aplicaciones.
- El proceso de creación de una base de datos en Oracle es bastante elaborado.
- Oracle dispone de la utilidad gráfica (**Asistente de configuración de Base de Datos**, dentro de: Todos los programas/Oracle-Db11g-Home/ Herramientas de configuración y migración/) que facilita este proceso

Pag. :12

Oracle: Introducción a la organización física



Pag. :13

Oracle: Introducción a la organización física

- **BASE DE DATOS <-> TABLESPACE**
  - Una BD está formada por uno o varios tablespaces que almacenan los datos de sus tablas.
- **TABLESPACE<-> FICHERO**
  - Cada tablespace de una BD consiste en uno o varios ficheros de datos, que son estructuras físicas controladas por el Sistema Operativo en el que Oracle está instalado.
- En la Versión 11g están en el directorio:

  C:\oracle\oradata\bdisabel
- En Oracle 10g EX en C:\oraclexe\oradata
- **BASE DE DATOS <-> FICHERO**
  - Los datos de la BD son almacenados en los ficheros de datos de los tablespaces.

Pag. :14

Oracle: Tablespaces del sistema

- **SYSTEM**: Información de la propia gestión de Oracle.
  - Siempre está activo (online) cuando la BD está abierta.
  - Contiene las tablas del DD, concretamente en el 1º fichero.
  - Todos los elementos PL/SQL (procedimientos, funciones, paquetes y triggers) residen en este tablespace.
- Una pequeña BD podría tener solamente este tablespace.
- Contiene los segmentos de ROLLBACK (deshacer) con la información de deshacer.
- **USERS**: tablas personales de los usuarios.
- **TEMP**: tablas temporales.
- ...

Pag. :15

Reglas para los nombres de Oracle

- **Reglas para los nombres (nombre de columnas, nombre de tablas, …):**
  - Deben de comenzar con una letra.
  - Pueden tener una longitud de 1 a 30 caracteres de largo.
  - Deben contener solamente A-Z, a-z, 0-9, _, $ y #.
  - No deben duplicar el nombre de otro objeto que sea propiedad del mismo usuario o schema.
  - No debe ser una palabra reservada.

Pag. :16

Creación de tablas : CREATE TABLE

- Para crear tablas se usa el comando **CREATE TABLE**:

```
CREATE TABLE [esquema.]Nombre_tabla
(    Definición_Columna [,Definición_Columna] ...
     [,Constraint_tabla] [,Constraint_tabla]…
 )
[Opciones_ tabla];
```

- **Definición_Columna:**

  nombre_columna tipo_datos [Constraint_Columna]…

Pag. :17

Tipos de Datos

TIPO DATO	Significado
VARCHAR2(tamaño)	Dato carácter de longitud variable. Máx. 4000.
CHAR(tamaño)	Dato carácter de longitud fija. Máx. 255
NUMBER(p,s)	Dato numérico de longitud variable. **p:** entre 1..38
DATE	Valores de fecha y hora.
LONG	Dato carácter de longitud variable hasta 2 Gb.
CLOB	Dato carácter single-byte de hasta 4 Gb
RAW(tamaño)	Datos binarios según tamaño especificado
LONG RAW	Datos binarios de longitud variable hasta 2 Gb
BLOB	Datos binarios hasta 4 Gb.

8

Tipos de datos: CHAR y VARCHAR2

- Para **cadenas de caracteres** : que contienen datos alfanuméricos.
  - **CHAR(tamaño):** Cadenas de longitud fija (de 1 a 2000 bytes)
  - Ejemplos:

    SEXO CHAR;

    CODIGO CHAR(4);
  - **VARCHAR2(tamaño):**Cadenas de longitud variable (de 1 de 4000 bytes)
  - Ejemplos:

    NOMBRE VARCHAR2(30);

    DESCR_ARTIC VARCHAR2(40);

Pag. :19

Tipos de datos : NUMBER

- **NUMBER([precisión], [escala])**
  - Almacena datos **numéricos** :Enteros, decimales, con signo o sin él.
  - Precisión: número total de dígitos del número (de 1 a 38)
  - Escala : número de dígitos decimales
  - Ejemplos:

    N NUMBER(4,2); -- 2 enteros y 2 decimales

    CANTIDAD NUMBER(3); -- 3 enteros

Pag. :20

Tipos de datos : LONG

- **LONG**
  - Almacena cadenas de caracteres de longitud variable hasta 2 gigabytes. Se utiliza para textos muy grandes
  - Máximo 1 campo tipo LONG por tabla
  - No admite restricciones de integridad
  - No indexa
  - No se puede utilizar en cláusulas WHERE, GROUP BY, ORDER BY, CONNECT BY, DISTINCT, UNION, INTERSECT y MINUS.

Pag. :21

Tipos de datos: DATE

- **DATE**
  - Almacena información de fechas y horas

    Siglo/año/mes/día/hora/minutos/segundos
  - Longitud fija de 7 bytes.
  - Intervalo de fechas válidas del 1 de enero de 4712 antes de Cristo a el 31 de diciembre de 9999.
  - Ejemplo: **FECHA_VENTA DATE;**

Pag. :22

Tipos de datos: RAW y LONG RAW

- **RAW**
  - Almacena pequeños datos binarios.
  - Máximo 2000 bytes.
  - Permite indexar por ellos.
- **LONG RAW**
  - Almacena datos binarios. Utilizado para almacenar gráficos, sonidos, ...
  - Máximo 2 gigabytes.
  - No permite indexación.

Pag. :23

EJEMPLOS DE CREATE TABLE

- Ejemplos:

```
CREATE TABLE EMPLEADOS(
        DNI             VARCHAR2(9),
        NOMBRE          VARCHAR2(40) ,
        DIRECCION       VARCHAR2(50)
);
CREATE ARTICULOS(
        ID_ARTIC        NUMBER(6),
        DESCR_ARTIC     VARCHAR2(50),
        PRECIO_UNIT     NUMBER(6,2)
);
CREATE TABLE VENTAS(
        ID_ARTIC        NUMBER(6),
        FECHA           DATE ,
        UNIDADES        NUMBER(5)
);
```

Pag. :24

Creación de tablas : CREATE TABLE
Definición de Restricciones (Constraint)

- Una **constraint** es una restricción de una columna. Es un conjunto de reglas que han de cumplir los datos en sus valores antes de almacenarse en las tablas.
- Las **constraint o restricciones** pueden ser de **columna** o de **tabla**.
- **Constraint de columna** van justo después de cada columna:

```
CREATE TABLE EMPLEADOS(
        DNI             VARCHAR2(9)  CONSTRAINT PK_EMP PRIMARY KEY,
        NOMBRE          VARCHAR2(40)  NOT NULL,
        DIRECCION       VARCHAR2(50)
);
```

- **Constraint de tabla** se ponen al final de las definiciones de todas las columnas, pueden afectar a mas de una columna (ES MEJOR UTILIZAR ESTE TIPO). Es obligatorio utilizarlas cuando una restricción afecte a más de una columna, por ejemplo una clave primaria que tenga dos columnas.

```
CREATE TABLE VENTAS(
        ID_ARTIC        NUMBER(6),
        FECHA           DATE ,
        UNIDADES        NUMBER(5),
        CONSTRAINT PK_VEN PRIMARY KEY(ID_ARTIC,FECHA)
);
```

Pag. :25

Creación de tablas : CREATE TABLE
Restricciones(CONSTRAINT)

- **Definición_columna con restricción de columna:**

```
nombre_columna     tipo_datos
    [[CONSTRAINT [nombre_restriccion]]
            {[NOT NULL]/
            [UNIQUE]/
            [PRIMARY KEY]/
            [DEFAULT valor]/
            [REFERENCES nombre_tabla(columna)]
                    [ON DELETE CASCADE/SET NULL]]/
            [CHECK condición] },
    ]...
```

Pag. :26

Creación de tablas : CREATE TABLE Restricciones(CONSTRAINT)

- **Constraint_de_tabla:**

```
[CONSTRAINT [nombre_constraint]]
{     [NOT NULL]/
       [UNIQUE(columna1[,columna2])]/
       [PRIMARY KEY (columna1[,columna2])]/
       [FOREIGN KEY(columna1[,columna2])
              REFERENCES   nombre_tabla[(columna[,columna]…)]
                     [ON DELETE CASCADE/SET NULL/] ]
      / [CHECK condición],
      …..
}
```

Pag. :27

CREATE TABLE: NOT NULL/DEFAULT

- **NOT NULL** y **NULL** especifican que la columna admite o no admite valores nulos, es decir si un campo puede quedar sin valor, o con valor desconocido.
- **DEFAULT** indica el valor por defecto que toma el campo si no es especificado de forma explícita. Por ejemplo, si se declara la siguiente columna:

  `CODIGO_POSTAL NUMBER(5) NOT NULL DEFAULT 13001`

- Se indica que el campo Código Postal es un número entero de 5 caracteres, que no admite valores nulos, y que en caso de no especificarlo, por ejemplo, en una inserción, el valor que tomará es 13001.

Pag. :28

CREATE TABLE: Clave Primaria:PRIMARY KEY

- **PRIMARY KEY:** indica que la columna definida es la clave primaria. Una tabla tan solo puede tener una clave primaria.
- Como nombre de la constraint podemos poner: **PK_Nombre_Tabla**

```
CREATE TABLE  ALUMNOS(
        DNI          VARCHAR2(9)
           CONSTRAINT PK_ALU PRIMARY KEY,  -- Constraint de columna
        NOMBRE  VARCHAR2(50),
        DIRECCION VARCHAR2(60)
);
CREATE TABLE  MATRICULAS(
        DNI        VARCHAR2(9) ,
        CURSO  NUMBER(2),
        NIVEL    NUMBER(3),
        CONTRAINT PK_MATR PRIMARY KEY(DNI,CURSO)
         -- Constraint de tabla
);
```

Pag. :29

CREATE TABLE: UNIQUE

- **UNIQUE KEY** controla que no haya ningún valor repetido en el campo o campos. Crea un índice para esta columna. Puede tener valores nulos.
- Como nombre de la constraint podemos poner: **UK_Tabla_Columna:**

```
 CREATE TABLE  ALUMNOS(
        DNI          VARCHAR2(9)
                  CONSTRAINT PK_ALUM PRIMARY KEY,
                        -- Constraint de columna
        NOMBRE   VARCHAR2(50)
                  CONSTRAINT UK_ALUM_NOM UNIQUE,
        DIRECCION VARCHAR2(60)
      );
      CREATE TABLE  MATRICULAS(
        DNI        VARCHAR2(9) ,
        CURSO    NUMBER(2),
        NIVEL     NUMBER(3),
        CONSTRAINT UK_MATR_DNI_CUR UNIQUE(DNI,CURSO)
                  -- Constraint de tabla
      );
```

Pag. :30

CREATE TABLE: Clave ajena: FOREIGN/REFERENCES

- **FOREIGN KEY ..REFERENCES:** indica que la columna o columnas es una clave ajena de otra tabla.
- **Nombre_columa1**: son las columnas de la tabla que forman la clave ajena.
- **Nombre_columa2:** son las columnas de la tabla referenciada.
- Las claves ajenas tiene que ser clave primaria en la tabla referenciada.

Pag. :31

CREATE TABLE: Clave ajena: FOREIGN/REFERENCES

- **Ejemplos:**

```
CREATE TABLE  ALUMNOS(
        DNI         VARCHAR2(9),
        NOMBRE  VARCHAR2(50),
        DIRECCION VARCHAR2(60),
        CONSTRAINT PK_ALUMNOS PRIMARY KEY(DNI)
);
 CREATE TABLE  MATRICULAS(
     DNIA         VARCHAR2(9),
     CURSO      NUMBER(2),
     NIVEL        NUMBER(3),
     CONSTRAINT PK_MATRICULA
       PRIMARY KEY(DNI,CURSO),
     CONSTRAINT FK_MAT_ALU_DNI FOREIGN KEY(DNIA)
                 REFERENCES ALUMNOS(DNI)
 );
```

Pag. :32

CREATE TABLE: Clave ajena: FOREIGN/REFERENCES

- Ejemplo de clave ajena definiendo la restricción de tipo tabla :

```
CREATE TABLE ARTICULOS(
        IDARTIC             NUMBER(5)
                    CONSTRAINT PK_ARTICULO PRIMARY KEY,
        NOMBRE              VARCHAR2(60) ,
        UNIDALMACEN         NUMBER(4),
        PRECIOUNIT          NUMBER(8,2)
) ;
CREATE TABLE VENTAS(
        IDARTICULO          NUMBER(5) ,
        FECHA               DATE,
        UNIDADES            NUMBER(4),
        CONSTRAINT PK_VENTAS PRIMARY KEY(IDARTICULO ,FECHA),
        CONSTRAINT FK_VEN_ART_IDARTCULO
                    FOREIGN KEY(IDARTICULO) REFERENCES ARTICULOS(IDARTIC)
                    ON DELETE CASADE
);
```

- **ON DELETE CASCADE**: si se borran las columnas de la tabla ARTICULO se borran todos los registros que coincidan en la tabla VENTAS.
- En **Oracle**, si las columnas se llaman igual se puede poner:

```
FOREIGN KEY (IDARTICULO) REFERENCES ARTICULOS
```

Pag. :33

CREATE TABLE: Clave ajena: FOREIGN/REFERENCES

- Ejemplo de clave ajena :

```
CREATE TABLE ARTICULOS(
        IDARTIC             NUMBER(5) CONSTRAINT PK_ARTICULO PRIMARY KEY,
        NOMBRE              VARCHAR2(60) ,
        UNIDALMACEN         NUMBER(4),
        PRECIOUNIT          NUMBER(8,2)
) ;
CREATE TABLE VENTAS(
        IDARTICULO          NUMBER(5)
            CONSTRAINT FK_VEN_ART_IDART  REFERENCES ARTICULOS(IDARTIC)
                    ON DELETE CASADE,
        FECHA               DATE,
        UNIDADES            NUMBER(4),
         CONSTRAINT PK_VENTAS PRIMARY KEY(IDARTICULO ,FECHA)
);
```

- **ON DELETE CASCADE**: si se borran las columnas de la tabla ARTICULO se borran todos los registros que coincidan en la tabla VENTAS.
- En **Oracle**, si las columnas se llaman igual se puede poner:

```
FOREIGN KEY (IDARTICULO) REFERENCES ARTICULOS
```

Pag. :34

CREATE TABLE: Clave ajena: FOREIGN/REFERENCES

- Ejemplo de clave ajena , el orden de las columnas en la clave ajena debe ser el mismo, es decir, si en VENTAS la clave primaria está declarada como (IDARTICULO,FECHA) en la FK debe tener el mismo orden. El tipo de las columnas en ambas tablas debe ser el mismo :

```
CREATE TABLE  VENTAS_HIS(
        IDVENTA          NUMBER(5),
        DESCRIPCION      VARCHAR2(200),
        IDARTICULO_H     NUMBER(4),
        FECHA_H                  DATE,
        FECHABAJA        DATE,
        FOREIGN KEY(IDARTICULO_H,FECHA_H)
                REFERENCES  VENTAS(IDARTICULO,FECHA)
);
```

Pag. :35

CREATE TABLE: Clave ajena: FOREIGN/REFERENCES

- Las opciones **ON DELETE** establecen el comportamiento del gestor en caso de que las filas de la tabla padre (es decir, la tabla referenciada) se borren.
- Los comportamientos pueden ser **CASCADE, SET NULL .**
  - **CASCADE**: la operación se propaga en cascada a la tabla hija, es decir, si se actualiza la tabla padre, se actualizan los registros relacionados de la tabla hija, y si se borra un registro de la tabla padre, se borran aquellos registros de la tabla hija que estén referenciando al registro borrado.
  - **SET NULL:** se establece a NULL la clave foránea afectada por un borrado o modificación de la tabla padre.
- Si no se especifica ON DELETE por defecto se actúa como NO ACTION.
- **Oracle no implementa** la opción **ON UPDATE** por lo que hay que recurrir a otros métodos para realizar las acciones de actualización, como por ejemplo, mediante TRIGGERS (disparadores).

Pag. :36

CREATE TABLE: CHECK

- **CHECK:** nos indica una condición debe cumplir la columna antes de ser insertada.
- En **Oracle**, **si chequea** antes de insertar.
- **No** admite **subconsultas**.
- En **Oracle no** permite utilizar la fecha del **sistema(SYSDATE o CURRENT_DATE**) como parte de la condición:

```
CREATE TABLE VENTAS(
        IDARTICULO      NUMBER(5),
        FECHA           DATE CHECK (FECHA <SYSDATE),
        UNIDADES        NUMBER(4),
        PRIMARY KEY(IDARTICULO,FECHA),
        FOREIGN KEY(IDARTICULO) REFERENCES ARTICULOS(IDARTICULO)
);
```

Pag. :37

CREATE TABLE: CHECK Operadores

OPERADORES ARITMÉTICOS:

Operador	Significado	Ejemplo
+	Suma	A+B
-	Resta	A-B
/	División	A/B
*	Multiplicación	A*B
%	Resto de la división	A%B

Pag. :38

CREATE TABLE: CHECK
Operadores

OPERADORES DE COMPARACIÓN O RELACIONALES

Operador	Significado	Ejemplo
!=, <>	Distinto	A !=0
> , >=	Mayor , Mayor o igual que	A>0 , A>=0
< , <=	Menor, Menor o igual	A<0 , A<=0
LIKE	Se utiliza para unir cadenas de caracteres. **%** : Representa cualquier cadena de caracteres de 0 o mas caracteres. **_**: Representa un único carácter cualquiera.	NOMBRE LIKE 'GOMEZ%' NOMBRE LIKE 'A_B'
IN	Igual a cualquiera de los miembros entre paréntesis	IDDEPARTAMENTO IN (10, 30)
NOT IN	Distinto a cualquiera de los miembros entre paréntesis	IDDEPARTAMENTO NOT IN (20,30,40);
BETWEEN	Contenido en el rango	IDDEPARTAMENTO BETWEEN 10 AND 40;
NOT BETWEEN	Fuera del rango	IDDEPARTAMENTO NOT BETWEEN 10 AND 40;

Pag. :39

CREATE TABLE: CHECK
Operadores

OPERADORES LÓGICOS:

Operador	Significado	Ejemplo
AND	**A AND B** Cierto si son ciertas a y b.	A=1 AND B>10 (SEXO='H' OR SEXO='M') AND EDAD>10
OR	**A OR B** Cierto si a o b son ciertas	A=1 OR B>10 SEXO='H' OR SEXO='M'
NOT	Negación	NOT A=0

Pag. :40

CREATE TABLE: CHECK

- Ejemplos de **CHECK** en **Oracle**:

```
CREATE TABLE ALUMNOS(
        DNI       VARCHAR2(9),
        CURSO VARCHAR2(4),
        EDAD    NUMBER(2),
        NIVEL    NUMBER(1),
        FECHA_NAC      DATE,
        CONSTRAINT PK_ALUMNOS PRIMARY KEY(DNI),
        CONSTRAINT CK_EDAD CHECK (EDAD>=12 AND EDAD<=16),
        CONSTRAINT CK_NIVEL CHECK (NIVEL >0 AND NIVEL <7),
         CONSTRAINT CK_CURSO_EDAD
            CHECK (CURSO IN (1,2) AND (EDAD >=12 AND EDAD<=14) )
        CONTRAINT CK_FECHA_NAC
            CHECK(FECHA_NAC>=TO_DATE('01/10/2000','DD/MM/YYYY')
                  AND FECHA_NAC <=TO_DATE('01/01/2100','DD/MM/YYYY'))
);
```

Pag. :41

CREATE TABLE: CHECK

- Para que la condición **CHECK** de las fecha en **Oracle** no de error, hay que convertir una cadena '01/12/2000' a fecha mediante la función TO_DATE. Tiene el siguiente formato:

  TO_DATE(cadena_fecha, cadena_formato)

- **cadena_fecha** : una fecha entre comillas simples
  '01/12/2012'
- **cadena_formato**: es el formato que tendrá nuestra fecha, puede ser 'DD/MM/YYYY', donde DD es el día, MM es el mes y YYYY es el año con 4 dígitos.

Pag. :42

EJEMPLOS CREATE TABLE

```
CREATE TABLE DEPARTAMENTO(
        IDDEPT          NUMBER(4),
        NOMBRE          VARCHAR2(30) NOT NULL,
        LOC             VARCHAR2(40),
        TIPO            CHAR,
        CONSTRAINT PK_DEP PRIMARY KEY(IDDEPT),
        CONSTRAINT CK_DEP_TIPO CHECK(TIPO IN ('H','U','I'))
);
CREATE TABLE EMPLEADO(
        IDEMPLE         NUMBER(6),
        NOMBRE          VARCHAR2(10),
        DNI             VARCHAR2(9) UNIQUE,
        PROVINCIA       NUMBER(2) DEFAULT 13,
        SEXO            CHAR,
        FECHA_ALTA      DATE,
        IDDEPT          NUMBER(4) REFERENCES DEPARTAMENTO,
        PRIMARY KEY(IDEMPLE),
        CHECK(UPPER(SEXO) IN ('H','M')),
                -- UPPER- CONVIERTE A MAYÚSCULAS
);
```

Pag. :43

Diferencias entre SGBD

- Cada gestor de base de datos efectúa sus propias modificaciones al formato de la sintaxis **CREATE TABLE**.
- La cláusula **opciones_tabla** permite especificar las peculiaridades de cada gestor con respecto al almacenamiento en soporte físico de sus tablas.
- Además, cada gestor incorpora diversas características, por ejemplo, Oracle y DB2 implementan tipos de datos distintos a MySQL o a SQL Server y Access.

Pag. :44

Características de la creación de tablas para Orade

- **Opción_tabla** en **Oracle**:

```
TABLESPACE nombre_tablespace
STORAGE (INITIAL  valor_inicial NEXT valor_siguiente
        MINEXTENTS minimo
        MAXEXTENTS {maximo/UNLIMITED }
        PCTINCREASE  incremento)
```

Pag. :45

Características de la creación de tablas para Orade

- En Oracle, la mayoría de las opciones tienen que ver con su almacenamiento físico.
- Por ejemplo, las tablas deben ser almacenadas en un contenedor llamado **tablespace** (Espacio de tablas).
- Por defecto, si no se indican opciones de almacenamiento, la tabla se ubica en el tablespace del usuario, pero si se quiere ubicar en otro tablespace, se puede incluir la opción **tablespace nombre** para designar otro tablespace.
- Además, se puede definir cómo se reserva el espacio en disco para controlar el crecimiento desmedido del tamaño de una tabla mediante la cláusula **storage**.

Pag. :46

Características de la creación de tablas para Oracle

- Ejemplo de creación de tablas con opciones propias de **Oracle**

```
CREATE TABLE PEDIDO(
   CODIGO          NUMBER(5) PRIMARY KEY,
   FECHA   DATE,
   ESTADO          CHAR(1) ,
   CONSTRAINT CK_ESTADO
   CHECK (ESTADO IN ('P' ,'E' ,'R')
)
TABLESPACE ADMINISTRACION
STORAGE (INITIAL 100K NEXT 100K
            MINEXTENTS 1 MAXEXTENTS UNLIMITED
            PCTINCREASE 10);
```

Pag. :47

Características de la creación de tablas para Orade

- **DEFAULT STORAGE:** Define el almacenamiento de la tabla.
  - **INITIAL:** Especifica el tamaño inicial de la 1ª extensión
  - **NEXT:** Tamaño de la siguiente extensión
  - **MINEXTENTS**: Nº Mínimo de extensiones
  - **MAXEXTENTS:** Nº Total de extensiones
  - **PCTINCREASE:** Es un factor de crecimiento de las siguientes extensiones. (Por defecto 50)

    NEXT=NEXT + (PCTINCREASE*NEXT)/100

Pag. :48

Consulta de las tablas de una base de datos en Oracle

- En **Orade**, se pueden consultar las vistas **USER_TABLES, DBA_TABLES** Y **ALL_TABLES**.
- Las vistas que comienzan por **USER_** , aportan información sobre los objetos que posee el usuario conectado.
- Las vistas que comienzan por **DBA_** son solo accesibles por los administradores de bases de datos y muestran información sobre todos los objetos.
- Finalmente, las vistas que comienzan por **ALL_** muestran la información sobre los objetos a los que el usuario tiene acceso, sean suyos o no.

Pag. :49

Modificación de tablas: ALTER TABLE

ALTER TABLE nombre_tabla Especificación_Alter [, Especificación_Alter] ...;

- ***Especificación_Alter:***

```
{ADD definición_columna
|ADD (definición_columna, definición_columna, ... )
|ADD [CONSTRAINT [símbolo]] PRIMARY KEY (nombre_columna, ... )
|ADD [CONSTRAINT [símbolo]] UNIQUE (nombre_columna, ... )
|ADD [CONSTRAINT [símbolo]] FOREIGN KEY (nombre_columna, ... )
      REFERENCES nombre_tabla(nombre_columna,…)
|ADD [CONSTRAINT [símbolo]] CHECK (condicion)
|MODIFY definición_columna
|MODIFY CONSTRAINT  nombre_constraint {DISABLE/ENABLE}
|RENAME COLUMN anterior_nombre_columna TO nuevo_nombre_columna
|DROP COLUMN nombre_columna
|DROP PRIMARY KEY
|DROP CONSTRAINT nombre_constraint
}
```

Pag. :50

Modificación de tablas: ALTER TABLE: Añadir Columnas

ADD definición_columna

ADD (definición_columna, definición_columna, ...)

- Esta opción **ADD** permite añadir columnas a una tabla
- En Oracle las columnas se insertan al final.
- Con la segunda opción nos permitirá insertar varias columnas a la vez.
- Inserta la columna POBLACION a la tabla EMPLEADOS:

```
ALTER TABLE EMPLEADOS ADD POBLACION VARCHAR2(20);
```

- Inserta una columna llamada OBSERVACIONES a la tabla CLIENTES:

```
ALTER TABLE CLIENTES ADD OBSERVACIONES NUMBER(9) ;
```

- No se puede insertar una columna con la constraint de NOT NULL si la tabla tiene registros.
- Para añadir varias columnas:

```
ALTER TABLE CLIENTES ADD ( DIRECCION2 VARCHAR2(50),
    MOVIL NUMBER(9));
```

Pag. :51

Modificación de tablas: ALTER TABLE: Añadir Constraints

```
ADD [CONSTRAINT [símbolo]] PRIMARY KEY (nombre_columna, ... )
ADD [CONSTRAINT [símbolo]] UNIQUE (nombre_columna, ... )
ADD [CONSTRAINT [símbolo]] FOREIGN KEY (nombre_columna, ... )
    REFERENCES nombre_tabla(nombre_columna,…)
ADD [CONSTRAINT [símbolo]] CHECK (condicion)
```

- Añade una clave primaria a la tabla EMPLEADOS

```
ALTER TABLE EMPLEADOS
    ADD CONSTRAINT PK_EMPLEADOS PRIMARY KEY(IDEMPLEADO);
```

- Añade una constraint de valor único a la tabla EMPLEADOS de la columna DNI:

```
ALTER TABLE EMPLEADOS ADD
    CONSTRAINT UK_DEPART UNIQUE(DNI);
```

- Inserta una constraint en la tabla:

```
ALTER TABLE EMPLEADOS ADD CONSTRAINT CK_SALARIO
    CHECK  (SALARIO>100);
```

Pag. :52

Modificación de tablas: ALTER TABLE: Añadir Constraints

- Añade una clave ajena a la tabla EMPLEADOS de la columna DEPT_NO de la tabla DEPART:

```
ALTER TABLE EMPLEADOS
    ADD CONSTRAINT FK_EMPLEADOS_DEPART
    FOREIGN KEY(DEPT_NO) REFERENCES DEPART(DEPT_NO);
```

- Añade una restricción check a la tabla CLIENTES para que la columna SEXO tenga los valores H o M:

```
ALTER TABLE CLIENTES
    ADD CONSTRAINT CK_CLIENTES_SEXO
    CHECK( SEXO IN ('H','M'));
```

- En **Oracle**, para añadir una constraint de tipo **CHECK**, todas las columnas de la tabla deben cumplir la condición, si no es así no dejará insertar la restricción.

Pag. :53

Modificación de tablas: ALTER TABLE: Modificar Columnas

MODIFY definición_columna

- No se puede modificar una columna poniéndole la opción NOT NULL si la columna tiene valores nulos.
- Insertar una condición de **NOT NULL** en una tabla en **ORACLE**:

```
ALTER TABLE EMPLEADOS MODIFY NOMBRE  NOT NULL;
ALTER TABLE DEPART ADD
  CONSTRAINT CK_DNOMBRE CHECK (DNOMBRE IS NOT NULL);
```

- Si la tabla tiene datos, **no** se puede **disminuir** la **longitud** de una columna.
- Modifica la columna APELLIDOS de la tabla EMPLEADOS

```
ALTER TABLE EMPLEADOS MODIFY APELLIDO VARCHAR2(100);
```

Pag. :54

Modificación de tablas: ALTER TABLE:Activar/Desact Constraint

MODIFY CONSTRAINT nombre_constraint {DISABLE/ ENABLE}

❖ Las constraints se pueden activar (ENABLE) o desactivar (DISABLE), para que no actúen.

```
ALTER TABLE EMPLEADOS MODIFY SYS_C0001233 DISABLE;
ALTER TABLE EMPLEADOS MODIFY SYS_C0001233 ENABLE;
```

Pag. :55

Modificación de tablas: ALTER TABLE: Borrar Columnas

DROP COLUMN nombre_columna

❖ Para borrar una columna:

```
ALTER TABLE EMPLEADOS DROP COLUMN DIR;
```

❖ Para poder borrar una columna ésta no puede estar como clave ajena en otra tabla.

Pag. :56

Modificación de tablas: ALTER TABLE: Borrado Restricciones

```
DROP PRIMARY KEY [CASCADE]
DROP CONSTRAINT nombre_constraint
```

- Con la opción **DROP** se pueden eliminar las restricciones de claves foráneas y primarias, dejando el tipo de dato y su contenido intacto.
- Para **borrar** la **clave primaria** :

```
ALTER TABLE EMPLEADOS DROP PRIMARY KEY;
```

- Si hay tablas que **referencian** como **clave ajena** a esta columna habrá que indicar la opción **CASCADE**, para eliminarlas todas**:**

```
ALTER TABLE EMPLEADOS DROP PRIMARY KEY CASCADE;
```

- Para eliminar una clave ajena llamada FK_EMPLEADOS_DEPART:

```
ALTER TABLE EMPLEADOS
    DROP CONSTRAINT FK_EMPLEADOS_DEPART;
```

Pag. :57

Modificación de tablas: ALTER TABLE: Renombrar Columnas

```
RENAME COLUMN anterior_nombre_columna
        TO nuevo_nombre_columna
```

- Para cambiar el nombre de una columna:

```
ALTER TABLE EMPLEADOS
    RENAME COLUMN APELLIDO TO NOMBRE;
```

Pag. :58

Modificación de tablas: ALTER TABLE

- Para borrar la clave primaria de la tabla Clientes, eliminar la columna CodigoCliente, añadir la columna NIF y establecerlo como clave primaria, hay que ejecutar las siguientes sentencias:

```
ALTER TABLE CLIENTES DROP PRIMARY KEY;
ALTER TABLE CLIENTES DROP COLUMN CODIGOCLIENTE;
ALTER TABLE CLIENTES ADD COLUMN NIF VARCHAR(10) PRIMARY KEY;
```

- En **Oracle** se puede borrar cualquier constraint, siempre que sepamos el nombre.

Pag. :59

DROP TABLE: Borrado de Tablas

- En **Oracle**, para eliminar una tabla:

```
DROP [TEMPORARY] TABLE tbl_name
    [CASCADE CONSTRAINT];
```

- No se permite el borrado de varias tablas en la misma sentencia, y con la opción **CASCADE CONSTRAINT** elimina las claves foráneas de las tablas relacionadas (en cascada).

```
--Elimina la tabla Partidos
DROP TABLE Partidos;
--Elimina la tabla Jugadores y
--borra las claves foráneas de otras tablas
DROP TABLE Jugadores CASCADE CONSTRAINT;
```

Pag. :60

Renombrado de tablas

- Para renombrar la tabla JUGADORES y llamarla FUTBOLISTAS:

```
RENAME JUGADORES TO FUTBOLISTAS;
```

Pag. :61

Índices: CREATE INDEX

- Un **índice** de una base de datos es una estructura de datos que mejora la velocidad de las operaciones, permitiendo un rápido acceso a los registros de una tabla en una base de datos. Al aumentar drásticamente la velocidad de acceso, se suelen usar sobre aquellos campos sobre los cuales se hacen frecuentes búsquedas.
- El índice tiene un funcionamiento similar al índice de un libro, guardando parejas de elementos: el elemento que se desea indexar y su posición en la base de datos.
- Cuando se crea una tabla y se le define una clave primaria se crea un índice por esa o esas columnas.

Pag. :62

Índices: CREATE INDEX

- Sintaxis de las sentencias de creación y borrado:

```
CREATE [UNIQUE] INDEX nombre_indice
    ON nombre_tabla(nombre_col_1[ASC|DESC],…);

DROP INDEX nombre_indice ON nombre_tabla ;
```

- Ejemplo:

```
CREATE INDEX I_EMPLE_APE ON EMPLE(APELLIDO);
CREATE INDEX I_EMPLE_APE_OF
    ON EMPLE(APELLIDO,OFICIO);
```

Pag. :63

Creación de usuarios: CREATE USER

- Usuarios que se crean en la BD cuando se instala, tiene rol de administrador:
  - **SYS**. Propietario de las tablas del diccionario de datos.
  - **SYSTEM**. Realiza la administración de la BD (usuarios,...)
- Sentencia para crear un usuario:

```
CREATE USER nombre_usuario
    IDENTIFIED BY contraseña
    [DEFAULT TABLESPACE nombre_tablespace]
    [TEMPORARY TABLESPACE nombre_tablespace]
    [QUOTA {entero {K|M} | UNLIMITED} ON no_tablespace
    [PROFILE perfil]
    [ACCOUNT LOCK/UNLOCK];
```

Pag. :64

Creación de usuarios: CREATE USER

- **DEFAULT TABLESPACE**: Tablespace donde por defecto se crearán sus objetos
- **TEMPORARY TABLESPACE**: tablespace temporal (ordenaciones,..)
- **QUOTA**: espacio máximo asignado al usuario en un tablespace. Por defecto al crear un usuario tiene establecido el permiso UNLIMITED TABLESPACE. Para poder asignar cuotas, primero hay que quitarle este permiso.
- **PROFILE**: perfil asignado al usuario
- **ACCOUNT LOCK/UNLOCK**: cuenta bloqueada o desbloqueada.

Pag. :65

Ejemplos de creación de usuarios

```
CREATE USER JUAN IDENTIFIED BY JUAN
  DEFAULT TABLESPACE USERS
  TEMPORARY TABLESPACE TEMP
  QUOTA 200M ON USERS
  QUOTA 0 ON SYSTEM;
```

- Si el usuario tiene asignado el permiso de UNLIMITED TABLESPACE, para poder asignarles cuotas hay que quitar primero este permiso:

```
REVOKE UNLIMITED TABLESPACE FROM USUARIO;
```

Pag. :66

Modificación de usuarios: ALTER USER

```
ALTER USER nombre_usuario
  IDENTIFIED BY contraseña
  [DEFAULT TABLESPACE nombre_tablespace]
  [TEMPORARY TABLESPACE nombre_tablespace]
  [QUOTA {entero {K|M} | UNLIMITED} ON n_tablespace
  [PROFILE perfil]
  [ACCOUNT UNLOCK/LOCK]
  [DEFAULT ROLE nombre_rol1[,nombre_rol2]..|ALL |NONE];
```

Cada usuario puede cambiar únicamente su clave de acceso, el resto teniendo el privilegio **ALTER USER**.

Pag. :67

Modificación de usuarios: ALTER USER

- **PROFILE perfil :** Asigna un perfil por defecto a un usuario
- **ACCOUNT UNLOCK/LOCK:** Desbloquear/Bloquear cuentas
- **DEFAULT ROLE nombre_rol1[,nombre_rol2]..|ALL| NONE:** Activa uno o varios roles para un usuario(debe tenerlos asignado previamente con un GRANT).
- Si se especifica ALL: actúan todos los roles
- Si se especifica NONE: no actúa ninguno
- Cuando se le asigna uno o varios roles por defecto los demás que no se asignan, dejan de actuar.

Pag. :68

Ejemplos de ALTER USER

- Modifica el usuario JUAN asignándole la clave JUAN y una cuota de 100M en el tablespace SYSTEM.

```
ALTER USER JUAN IDENTIFIED BY JUAN
    QUOTA 100 M ON SYSTEM;
```

- Modifica el usuario JUAN asignándole por defecto el perfil PERFIL_UNO:

```
ALTER USER JUAN PROFILE PERFIL_UNO;
```

- Desbloquea el usuario JUAN

```
ALTER USER JUAN ACCOUNT UNLOCK;
```

- Asigna al usuario JUAN los roles CONNECT y RESOURCE

```
GRANT CONNECT,RESOURCE TO JUAN;
```

- Asigna al usuario JUAN el rol RESOURCE como rol por defecto.

```
ALTER USER JUAN DEFAULT ROLE RESOURCE;
```

Pag. :69

Borrado de usuarios: DROP USER

- Sentencia para eliminar un usuario:

```
DROP USER nombre_usuario [CASCADE];
```

- **CASCADE**: Se borran también los objetos que contiene el esquema del usuario
- Ejemplo:

```
DROP USER JUAN CASCADE;
```

- Para poder eliminar un usuario se ha de tener el privilegio DROP USER.

Pag. :70

Diccionario datos: Información sobre usuarios

- La tabla del diccionario de datos para saber toda la información de los usuarios es:

USER_USERS

DBA_USERS (todos los usuarios, debe de tener permisos dba para ver esta tabla)

Pag. :71

Privilegios: GRANT

- Es la capacidad de un usuario de realizar determinadas operaciones o a acceder a determinados objetos de otros usuarios. Cuando se crea un usuario no tiene ningún permiso, ni siquiera el de conexión a la BD (CREATE SESSION)

2 tipos:

- **Privilegios sobre los OBJETOS** (los objetos son: tablas, vistas, índices,…)
- **Privilegios del SISTEMA**

Pag. :72

Crear un Rol

- **Rol**: conjunto de privilegios a los cuales se les asigna un nombre. Se le pueden asignar privilegios sobre objetos y del sistema.
- Instrucción para crear un rol:

```
CREATE ROLE nombre_rol ;
```

- Después se le asignan los privilegios con GRANT:

```
CREATE ROLE ROLALUMNO;

GRANT  CREATE SESSION, CREATE USER, DROP USER,
      ALTER USER          TO ROLALUMNO;

GRANT SELECT, UPDATE, DELETE ON  EMPLE TO
ROLALUMNO;
```

- Borrar una rol:

```
DROP ROLE nombre_rol;

DROP ROLE ROLALUMNO;
```

Pag. :73

Privilegios : Roles del sistema (Algunos)

Rol	Privilegios
CONNECT	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE	CREATE CLUSTER, CREATE PROCEDURE, CREATE TABLE, CREATE SEQUENCE, CREATE TRIGGER
DBA	TODOS LOS PRIVILEGIOS DEL SISTEMA
EXP_FULL_DATABASE	SELECT ANY TABLE, BACKUP ANY TABLE, INSERT, UPDATE, DELETE sobre las tablas de SYS.INCVID
IMP_FULL_DATABASE	BECOME USER

Pag. :74

Privilegios sobre los OBJETOS

Nos van a permitir acceder y realizar cambios en los objetos de otros usuarios.

	Tabla	Vista	Secuencia	Procedimiento
ALTER	X		X	
SELECT	X	X		
INSERT	X	X		
DELETE	X	X		
UPDATE	X	X		
EXECUTE				X
INDEX	X			
REFERENCES	X(FK)			
ALL				

Pag. :75

Privilegios sobre OBJETOS: GRANT..ON..TO

GRANT {privilegio1 [, privilegio2]... | ALL [PRIVILEGES]}

[(columna1 [, columna2]...)]

ON [usuario.]objeto

TO {usuario1 | rol1 | PUBLIC} [, {usuario2 | rol2 | PUBLIC} ...]

[WITH GRANT OPTION];

- **privilegio1,…:** son los privilegios otorgados.
- **ON [usuario.]objeto:** Objeto sobre el que se dan los privilegios.
- **TO usuario/rol/PUBLIC:** usuario/s a los que se le concede el permiso. Se puede dar el permiso a un ROL creado anteriormente o a todos los usuarios(PUBLIC)
- **WITH GRANT OPTION:** Puede dar el privilegio/s otorgado a otros usuarios.
- **PUBLIC:** concede el permiso a todos los usuarios
- El usuario que concede el privilegio debe ser el propietario de dicho objeto

Pag. :76

Ejemplos de privilegios sobre objetos

- Se conecta el usuario MARIA a la base de datos BDORACLE:

  SQL> CONNECT MARIA/MARIA@BDORACLE;
- María le concede el privilegio de insertar en su tabla EMPLE al usuario JUAN:

  SQL> GRANT INSERT ON EMPLE TO JUAN;
- Se conecta el usuario JUAN:

  SQL>CONNECT JUAN/JUAN@BDORACLE
- Si intenta visualizar los datos de la tabla EMPLE no le deja porque no tiene permisos:

  SQL> SELECT * FROM MARIA.EMPLE;

  Privilegios insuficientes
- Insertar sí le deja porque se le ha dado dicho permiso anteriormente:

  SQL> INSERT INTO MARIA.TABLA_EMPLE

  VALUES('1','JOSE SANCHEZ','C/TOLEDO,1');

  1 registro creado

Pag. :77

Ejemplos de privilegios sobre objetos

- Se puede dar privilegios a columnas concretas, la siguiente sentencia concede el permiso a JUAN de actualizar sólo la columna SUELDO de la tabla EMPLE:

  GRANT UPDATE(SALARIO) ON EMPLE TO JUAN;
- Conceder todos los privilegios sobre EMPLE a todos los usuarios:

  GRANT ALL ON EMPLE TO PUBLIC;
- Para JUAN pueda conceder el privilegio a otro usuario:

  CONNECT MARIA/MARIA@BDORACLE

  GRANT INSERT ON EMPLE TO JUAN WITH GRANT OPTION;

  CONNECT JUAN/JUAN@BDORACLE

  GRANT INSERT ON MARIA.EMPLE TO PACO;

Pag. :78

Ejemplos de privilegios sobre objetos

- Para dar varios permisos a JUAN sobre la tabla EMPLE:

```
GRANT INSERT, SELECT, UPDATE(APELLIDO) ON EMPLE
    TO JUAN;
```

- Si queremos dar a un usuario el mismo permiso sobre diferentes objetos necesitamos poner varias sentencias GRANT:

```
GRANT INSERT, SELECT, UPDATE(APELLIDO,OFICIO)
    ON EMPLE TO JUAN,PACO, USUARIO01;
GRANT INSERT, SELECT, UPDATE(APELLIDO,OFICIO)
    ON DEPART TO JUAN,PACO, USUARIO01;
```

- Para conceder un permiso a un rol que hayamos creado anteriormente y posteriormente dar este rol al usuario JUAN:

```
GRANT INSERT, DELETE, SELECT ON EMPLE TO
ROL_EJEMPLO;
GRANT ROL_EJEMPLO TO JUAN;
```

Pag. :79

Creación de sinónimos

- Crea un **sinónimo** para algún objeto de la base de datos:

```
CREATE [OR REPLACE] [PUBLIC] SYNONYM
    [esquema.]sinonimo
    FOR [esquema.]objeto
```

- Con la opción **PUBLIC** se crea un sinónimo público accesible a todos los usuarios, siempre que tengan los privilegios adecuados para el mismo.
- Sirve para no tener que usar la notación **esquema.objeto** para referirse a un objeto que no es propiedad de usuario.

Pag. :80

Creación de sinónimos

- Ejemplo: crea un sinónimo llamado T_PEDIDOS para la tabla T_PEDIDOS del usuario PROGRAMADOR:

```
CREATE PUBLIC SYNONYM
    ST_PEDIDOS
    FOR PROGRAMADOR.T_PEDIDOS;
```

- De esta forma en vez de poner:

```
SELECT * FROM PROGRAMADOR.T_PEDIDOS;
```

- Podemos usar:

```
SELECT * FROM ST_PEDIDOS;
```

Pag. :81

Privilegios del SISTEMA: GRANT .. TO

- Los privilegios de sistema se pueden clasificar de la siguiente forma:
  - Los privilegios que permiten las operaciones en todo el sistema, por ejemplo **CREATESESSION**, **CREATE TABLESPACE**
  - Los privilegios que permiten la gestión de objetos en el propio esquema de un usuario, por ejemplo **CREATE TABLE.**
  - Los privilegios que permiten la gestión de objetos en cualquier esquema, por ejemplo **CREATE ANY TABLE**.

Pag. :82

Privilegios del SISTEMA: GRANT .. TO

GRANT {privilegio1| rol1} [, {privilegio2| rol2},...]

TO {usuario2|rol2|PUBLIC} [, {usuario2|rol2|PUBLIC} ...]

[WITH ADMIN OPTION];

- Concede un privilegio del sistema a un usuario/rol o a todos los usuarios.
- **WITH ADMIN OPTION:** Permite que el usuario receptor pueda otorgar dicho privilegio a otros

Pag. :83

Privilegios del SISTEMA: GRANT .. TO

Privilegio	Descripción
ANALYZE	
ANALIZE ANY	Analiza cualquier tabla o índice de la base de datos
AUDIT	
AUDIT ANY	Auditar cualquier objeto de la base de datos
AUDIT SYSTEM	Activar y desactivar la auditoría
DATABASE	
ALTER DATABASE	Modificar la base de datos. Añadir ficheros a los tablespaces, siempre que se tenga el privilegio del sistema operativo.
DATABASE LINK	
CREATE DATABASE LINK	Crear enlaces de base de datos privados en el esquema propio
INDEX	
CREATE ANY INDEX	Crear índices sobre cualquier tabla en cualquier esquema
ALTER ANY INDEX	Modificar índices sobre cualquier tabla en cualquier esquema
DROP ANY INDEX	Borrar índices sobre cualquier tabla en cualquier esquema
LIBRARY	
CREATE LIBRARY	Crear librerías externas en el esquema propio
CREATE ANY LIBRARY	Crear librerías externas en cualquier esquema
DROP LIBRARY	Eliminar librerías externas en el esquema propio
DROP ANY LIBRARY	Eliminar Crear librerías externas en cualquier esquema
PRIVILEGE	
GRANT ANY PRIVILEGE	Conceder cualquier privilegio del sistema, pero no sobre objetos
PROCEDURE	
CREATE PROCEDURE	Crear un procedimiento en el esquema propio
CREATE ANY PROCEDURE	Crear procedimientos bajo cualquier esquema
ALTER ANY PROCEDURE	Modificar cualquier procedimiento de la base de datos
DROP ANY PROCEDURE	Borrar cualquier procedimiento de la base de datos
EXECUTE ANY PROCEDURE	Ejecutar cualquier procedimiento de la base de datos

Pag. :84

Privilegios del SISTEMA: GRANT .. TO

PROFILE	
CREATE PROFILE	Crear perfiles
ALTER PROFILE	Modificar cualquier perfil de la base de datos
DROP PROFILE	Borrar cualquier perfil de la base de datos
ALTER RESOURCE COST	Calcular los costes de recursos utilizados en todas las sesiones de los usuarios
PUBLIC DATABASE LINK	
CREATE PUBLIC DATABASE LINK	Crear enlaces de base de datos públicos
DROP PUBLIC DATABASE LINK	Borrar enlaces de base de datos públicos
PUBLIC SYNONYM	
CREATE PUBLIC SYNONYM	Crear sinónimos públicos
DROP PUBLIC SYNONYM	Borrar sinónimos públicos

Pag. :85

Privilegios del SISTEMA: GRANT .. TO

ROLE	
CREATE ROLE	Crear roles
ALTER ANY ROLE	Modificar cualquier rol en la base de datos
DROP ANY ROLE	Borrar cualquier rol de la base de datos
GRANT ANY ROLE	Conceder cualquier rol de la base de datos
ROLLBACK SEGMENT	
CREATE ROLLBACK SEGMENT	Crear un segmento de rollback
ALTER ROLLBACK SEGMENT	Modificar un segmento de rollback
DROP ROLLBACK SEGMENT	Borrar un segmento de rollback
SESSION	
CREATE SESSION	Conectarse a la base de datos
ALTER SESSION	Ejecutar el comando alter session para cambiar parámetros de entorno
RESTRICTED SESSION	Conectarse a la base de datos que haya sido arrancada con el comando STARTUP RESTRICT
SEQUENCE	
CREATE SEQUENCE	Crear una secuencia en el propio esquema
CREATE ANY SEQUENCE	Crear una secuencia bajo cualquier secuencia
ALTER ANY SEQUENCE	Modificar cualquier secuencia de la base de datos
DROP ANY SEQUENCE	Borrar cualquier secuencia de la base de datos

Pag. :86

Privilegios del SISTEMA: GRANT .. TO

SYNONYM	
CREATE SYNONYM	Crear sinónimos en el esquema propio
CREATE ANY SYNONYM	Crear sinónimos bajo cualquier esquema
DROP ANY SYNONYM	Borrar cualquier sinónimo de la base de datos
SYSTEM	
ALTER SYSTEM	Ejecutar el comando ALTER SYSTEM
TABLE	
CREATE TABLE	Crear una tabla en el esquema propio
CREATE ANY TABLE	Crear tablas bajo cualquier esquema
ALTER ANY TABLE	Modificar la estructura de cualquier tabla de la base de datos
DROP ANY TABLE	Borrar cualquier tabla de la base de datos
EXPORT ANY TABLE	Realizar una exportación de cualquier tabla de la base de datos
LOCK ANY TABLE	Bloquear cualquier tabla o vista de la base de datos
COMMENT ANY TABLE	Comentar cualquier tabla de la base de datos
SELECT ANY TABLE	Realizar consultas de cualquier tabla de la base de datos
INSERT ANY TABLE	Realizar inserciones en cualquier tabla de la base de datos

Pag. :87

Privilegios del SISTEMA: GRANT .. TO

UPDATE ANY TABLE	Realizar actualizaciones en cualquier tabla de la base de datos
DELETE ANY TABLE	Realizar borrados de filas de cualquier tabla de la base de datos
TABLESPACE	
CREATE TABLESPACE	Crear un tablespace en la base de datos
ALTER TABLESPACE	Modificar tablespaces
MANAGE TABLESPACE	Poder poner un tablespace OFF/ON LINE, y ejecutar el comando ALTER TABLESPACE BEGIN/END BACKUP
DROP TABLESPACE	Eliminar tablespaces
UNLIMITED TABLESPACE	Utilizar cantidades de espacio ilimitado en cualquier tablespace de la base de datos
TRIGGER	
CREATE TRIGGER	Crear un disparador en el esquema propio
CREATE ANY TRIGGER	Crear disparadores bajo cualquier esquema de la base de datos
ALTER ANY TRIGGER	Modificar cualquier disparador de la base de datos
DROP ANY TRIGGER	Borrar cualquier disparador de la base de datos
USER	
CREATE USER	Crear un usuario
BECOME USER	Convertirse de forma temporal en otro usuario
ALTER USER	Modificar cualquier usuario de la base de datos
DROP USER	Eliminar a un usuario
VIEW	
CREATE VIEW	Crear una vista en el esquema propio
CREATE ANY VIEW	Crear vistas bajo cualquier esquema
DROP ANY VIEW	Borrar cualquier vista de la base de datos

Pag. :88

Ejemplos de privilegios del sistema

- Al crear un usuario (éste no tiene ningún privilegio):

```
CREATE USER JUAN IDENTIFIED BY JUAN
    DEFAULT TABLESPACE USERS
    TEMPORARY TABLESPACE TEMP;
```

- Le damos a JUAN los privilegios necesarios: se le asignan los roles CONNECT y RESOURCE:

```
GRANT CONNECT, RESOURCE TO JUAN;
```

- Le asignamos a JUAN los privilegios: conexión, crear usuarios y borrar usuarios:

```
GRANT CREATE SESSION ,CREATE USER, DROP USER
    TO JUAN;
```

Pag. :89

Ejemplos de privilegios del sistema

- Se le van a dar permisos a JUAN para que pueda crear, modificar y borrar tablas en cualquier esquema.

- Hay que darle los privilegios necesarios

```
GRANT CREATE ANY TABLE, DROP ANY TABLE,
ALTER ANY TABLE TO JUAN;
```

Pag. :90

Ejemplos de privilegios del sistema

- Otorgamos a MARIA y PEDRO los permisos para que puedan: crear, eliminar y modificar un tablespace, crear un perfil, crear y borrar un rol y crear ficheros en la base de datos :

```
GRANT CREATE TABLESPACE, ALTER TABLESPACE ,
        DROP TABLESPACE,
        CREATE PROFILE, DROP PROFILE,
        CREATE ROLE, DROP ROLE,ALTER
DATABASE
TO MARIA, PEDRO;
```

Pag. :91

Ejemplos de privilegios del sistema

- Otorgamos al rol creado anteriormente ROLEJEMPLO los permisos para que puedan: crear tablas en cq. usuario, eliminar y modificar cualquier tabla, crear vistas, crear y borrar índices :

```
GRANT CREATE ANY TABLE, ALTER ANY TABLE,
    DROP ANY  TABLE,
    CREATE VIEW, CREATE INDEX, DROP INDEX
TO ROLEJEMPLO;
```

- Le asignamos ese rol a los usuarios ALUMNO1 y ALUMNO2:

```
GRANT ROLEJEMPLO TO ALUMNO1, ALUMNO2
```

Pag. :92

Quitar de privilegios: REVOKE

- Quitar permisos sobre **objetos**:

```
REVOKE {privilegio1 [, privilegio2]... | ALL [PRIVILEGES]
ON [usuario.]objeto
FROM {usuario | rol | PUBLIC} [, {usuario | rol | PUBLIC} ...];
```

- Quitar permisos del sistema o roles a usuarios:

```
REVOKE {privilegio | rol}  [, {privilegio | rol},...]
FROM {usuario | rol | PUBLIC} [, {usuario | rol | PUBLIC}..];
```

Pag. :93

Ejemplos de REVOKE

```
REVOKE CREATE SESSION, RESOURCE FROM JUAN;
REVOKE  SELECT ON TABLA FROM JUAN;
REVOKE CREATE TABLE FROM PUBLIC;
REVOKE SELECT ON TABLA FROM PUBLIC;
```

Pag. :94

Dic. Datos: Vistas sobre privilegios y roles

- **SESSION_PRIVS** : Privilegios del usuario activo.
- **USER_SYS_PRIVS**: Privilegios del sistema asignados al usuario.
- **USER_ROLE_PRIVS**: roles asignados al usuario activo.
- **SESSION_ROLES**: roles activos para el usuario.
- **USER_TAB_PRIVS**: Privilegios de tablas asignados al usuario activo .
- **USER_COL_PRIVS**: Privilegios que tiene el usuario sobre las columnas de las tablas
- **DBA_SYS_PRIVS**: Privilegios del sistema que tienen asignados todos los usuarios o los roles creados en Oracle.
- **DBA_TAB_PRIVS**: Privilegios de tablas asignados a los usuarios o a los roles.
- **DBA_COL_PRIVS**: Privilegios que tienen los usuarios sobre las columnas de las tablas
- **DBA_ROLES** : Todos los roles existentes.
- **DBA_ROLES_PRIVS**: Usuarios a los que han sido otorgados roles.
- **ROLE_SYS_PRIVS**: privilegios de tablas asignados a los roles.
- **ROLE_ROLE_PRIVS**: Roles que han sido otorgados a otros roles.
- **DBA_ROLE_PRIVS**: privilegios de los roles
- Sólo los usuarios con role DBA tiene acceso a las vistas que comiencen con DBA_

Pag. :95

Perfiles

- Define unos límites en la utilización de recursos de la BD.

```
CREATE PROFILE nombre_perfil LIMIT
 nombre_recurso {entero [K|M] | UNLIMITED |DEFAULT}
 [nombre_recurso {entero [K|M] | UNLIMITED |DEFAULT}]...
```

- Modifica un perfil:

```
ALTER PROFILE nombre_perfil LIMIT
 nombre_recurso {entero [K|M] | UNLIMITED |DEFAULT}
 [nombre_recurso {entero [K|M] | UNLIMITED |DEFAULT}]...
```

- Borra un perfil:

```
DROP PROFILE nombre_perfil [CASCADE]
```

- Toma el límite del perfil DEFAULT (recursos ilimitados)

Pag. :96

Perfiles

- **Nombre_recursos**: Define unos límites en la utilización de recursos de la BD:
- **SESSIONS_PER_USER**: Nº sesiones concurrentemente
- **CONNECT_TIME**: Minutos de conexión
- **CPU_PER_SESSION**: Limita el tiempo máximo de CPU por llamada
- **IDLE_TIME:Tiempo** de inactividad, tras el cual desconecta al usuario (minutos)
- **FAILED_LOGIN_ATTEMPTS**: Nº de intentos posibles antes de bloquear la cuenta
- **PASSWORD_LIFE_TIME**: Nº de días que deben pasar como max. para la autentificación

Pag. :97

Ejemplos de perfiles

```
SQL>CREATE PROFILE PERFIL_ALUM LIMIT
      SESSIONS_PER_USER 1
      CONNECT_TIME 200
      IDLE_TIME 20
      CPU_PER_CALL 20;
```

- Nº sesiones concurrentemente
- Minutos de conexión
- Tiempo de inactividad, tras el cual desconecta al usuario (minutos)
- Limita el tiempo máximo de CPU por llamada (cs de segundo)

Pag. :98