TEMA 6

LENGUAJE DE DEFINICIÓN DE DATOS (DDL)

DDL
- CREATE
- ALTER
- DROP

CURSO 23-24

ÍNDICE

ESQUEMA EJEMPLO TEMA (CICLISMO)

EQUIPO (nomeq:varchar2(25), director:varchar2(100))
CP= {nomeq}

CICLISTA (dorsal: number(3), nombre: varchar2 (30), edad: number(2), nomeq: varchar2(25))
CP= {dorsal}
CA= {nomeq} → EQUIPO
VNN= {nomeq}
VNN= {nombre}

ETAPA (netapa: number(2), km: number(3), salida: varchar2(35), llegada varchar2(35), dorsal: number(3))
CP= {netapa}
CA= {dorsal} → CICLISTA

PUERTO (nompuerto: varchar2(35), altura: number(4), categoria: char(1), pendiente: number(3,2), netapa: number(2), dorsal: number(3))
CP= {nompuerto}
CA= {netapa} → ETAPA
BORRADO EN CASCADA
CA= {dorsal} → CICLISTA
VNN= {netapa}

MAILLOT (codigo: char(3), tipo: varchar2(30), color: varchar2(20), premio: number(7))
CP= {codigo}

LLEVAR (netapa: number(2), codigo: char(3), dorsal: number(3))
CP= {netapa,codigo}
CA= {netapa}→ ETAPA
CA= {codigo}→ MAILLOT
BORRADO EN CASCADA
CA= {dorsal}→ CICLISTA
VNN= {dorsal}

ÍNDICE

0.- INTRODUCCIÓN

Recordando…

- SQL= DML + DDL + DCL+ DTL
  - DML= Lenguaje de manipulación de datos:
    - Consulta
    - Inserción
    - Modificación
    - Borrado

    DE DATOS
  - DDL= Lenguaje de definición de datos
    - Creacion
    - Modificación
    - Borrado

    DE LOS OBJETOS DE LA BD
  - DCL= Lenguaje de control de acceso
  - DTL= Lenguaje de Control de transacciones

0.- INTRODUCCIÓN

DDL (DEFINICIÓN OBJETOS)	
SENTENCIA	**DESCRIPCION**
CREATE	CREACIÓN de objetos BD
DROP	BORRADO de objetos BD
ALTER	MODIFICACIÓN de objetos BD

DML (MANIPULACIÓN DATOS)	
SENTENCIA	**DESCRIPCION**
SELECT	CONSULTA datos
INSERT	INSERCIÓN datos
UPDATE	MODIFICACIÓN datos
DELETE	BORRADO datos

DCL (CONTROL DE ACCESO)	
SENTENCIA	**DESCRIPCION**
GRANT/REVOKE	CONCEDER/REVOCAR privilegios

DTL (CONTROL TRANSACCIONES)	
SENTENCIA	**DESCRIPCION**
COMMIT/ROLLBACK	VALIDAR/DESHACER Transacción actual

0.- INTRODUCCIÓN

- El DDL se encarga de la definición de todos los objetos y estructuras (básicas, derivadas, internas) de la BD:
  - Tablas, vistas, usuarios, etc.

DDL (DEFINICIÓN OBJETOS)	
SENTENCIA	**DESCRIPCION**
CREATE	CREACIÓN de objetos BD
DROP	BORRADO de objetos BD
ALTER	MODIFICACIÓN de objetos BD

0.- INTRODUCCIÓN

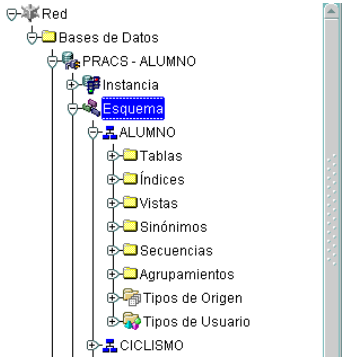
- Tal y como hemos visto a lo largo del curso el concepto teórico o estándar de esquema de BD difiere del concepto de esquema en Oracle:
  - En SQL estándar existe la sentencia CREATE SCHEMA para crear un esquema lógico de BD:

```
esquema::= CREATE SCHEMA [esquema]
           AUTHORIZATION usuario
           lista_elemento_esquema

elemento_esquema::= definicion_dominio|
                    definicion_tabla|
                    definicion_restriccion|
                    definicion_vista|
                    definicion_privilegio
```

  - En Oracle esta sentencia no existe.

0.- INTRODUCCIÓN

- En una BD Oracle asociado a cada usuario se crea un **esquema de usuario** en el que se almacenan todos los objetos creados por él (tablas, vistas, etc.)
  - Uno o varios esquemas lógicos relacionales



ÍNDICE

1.- USUARIOS

En Oracle (recordando…):

- **Usuario (User)**: cuenta con nombre definida en la BD que goza de distintos privilegios o permisos.
  - Define un esquema de usuario con su mismo nombre
  - Contiene todos aquellos objetos lógicos de la BD de los que es propietario el usuario
    - Creados por el usuario.
  - En principio un usuario **sólo puede acceder a los objetos de su esquema**
    - Podría acceder a los objetos del esquema de otro usuario si se le asignan permisos para ello.

1.- USUARIOS

- Al crear una BD Oracle se crean automáticamente 2 usuarios para realizar las tareas de administración:

SYSTEM	SYS
• Tareas GENERALES administración • Conveniente crear otro usuario • Nosotros hemos creado *alumno*	• Propietario tablas diccionario de datos • Tareas ESPECIALES administración • Backup/Recovery • Etc.

- En principio, estos usuarios son los únicos que disponen del permiso para crear usuarios:
  - Deben comenzar a crear los usuarios que podrán acceder a la BD y asignarles los permisos adecuados.
  - **Por ejemplo**: es conveniente que SYSTEM cree un nuevo usuario administrador de la BD (como *alumno*)

ÍNDICE

1.1- CREACIÓN DE USUARIOS

```
creacion_usuario::= CREATE USER usuario
                    IDENTIFIED BY password
                    [DEFAULT TABLESPACE tablespace]
                    [TEMPORARY TABLESPACE tablespace]
                    [comalista_cuota]
                    [PROFILE perfil]

cuota::= QUOTA {entero {K|M}|UNLIMITED} ON tablespace
```

- usuario es el nombre de usuario con clave de acceso password
- DEFAULT TABLESPACE asigna al usuario un tablespace por defecto para almacenar sus objetos:
  - Si no se asigna ninguno será SYSTEM (no recomendable).

The default setting for the default tablespaces of all users is the `SYSTEM` tablespace. If a user does not create objects, and has no privileges to do so, then this default setting is fine. However, if a user is likely to create any type of object, then you should specifically assign the user a default tablespace, such as the `USERS` tablespace. Using a tablespace other than `SYSTEM` reduces contention between data dictionary objects and user objects for the same data files. In general, do not store user data in the `SYSTEM` tablespace.

1.1- CREACIÓN DE USUARIOS

Continúa...

- TEMPORARY TABLESPACE asigna al usuario un tablespace temporal por defecto.
  - En las versiones anteriores a la 10 si no se asigna ninguno será SYSTEM.
  - A PARTIR DE LA VERSIÓN 10 si no se asigna ninguno será el tablespace temporal por defecto definido en la BD
    - En nuestra BD de prácticas será TEMP.
    - Si no hay ningún tablespace temporal por defecto asignado en la BD será SYSTEM (no recomendable).

If you do not explicitly assign the user a temporary tablespace, then Oracle Database assigns the user the default temporary tablespace that was specified at database creation, or by an `ALTER DATABASE` statement at a later time. If there is no default temporary tablespace explicitly assigned, then the default is the `SYSTEM` tablespace or another permanent default established by the system administrator. Do not store user data in the `SYSTEM` tablespace. Assigning a tablespace to be used specifically as a temporary tablespace eliminates file contention among temporary segments and other types of segments.

1.1.- CREACIÓN DE USUARIOS

Continúa...

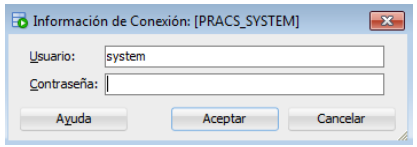
- comalista_*cuota* permite asignar un espacio determinado al usuario en ciertos tablespaces.
  - Este tamaño puede especificarse:
    - con un número entero en Kilobytes (K) o Megabytes (M).
    - indicando UNLIMITED, así el usuario dispone de cuota ilimitada en el tablespace.
  - Si no se asigna cuota en un tablespace **no se dispone de espacio en este**.
  - A partir de la versión 10 de Oracle no es posible especificar cuota para los tablespaces temporales.
- PROFILE asigna un perfil al usuario (limite recursos por sesión)
  - Si se omite se asigna el perfil por defecto (DEFAULT).
  - Los vimos en el boletín de instalación de Oracle.
    - Consultad la documentación de Oracle.

1.1.- CREACIÓN DE USUARIOS

- **Ejemplos**:
  - Supongamos que el usuario system se conecta a la BD de prácticas con cualquier herramienta de las que hemos visto en clase (SQL* Plus o SQL Developer)
    - A lo largo del tema, para ejemplificar de manera visual la conexión a un esquema de usuario indicaremos la cadena de conexión

```
CONNECT system/ADMIN123@pracs;
```

    - Aunque sabemos que en SQL Developer esta se lanza en background desde la ventana de una conexión…



1.1.- CREACIÓN DE USUARIOS

- **Ejemplos…**

```
CONNECT system/ADMIN123@pracs;

CREATE USER pruebas IDENTIFIED BY pruebas
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON USERS;
```

```
CREATE USER jward
 IDENTIFIED BY password
 DEFAULT TABLESPACE data_ts
 QUOTA 100M ON test_ts
 QUOTA 500K ON data_ts
 TEMPORARY TABLESPACE temp_ts
 PROFILE clerk;
```


1.1.- CREACIÓN DE USUARIOS

- Cuando se crea un usuario este no dispone de ningún tipo de permiso:
  - No puede realizar nada en la BD, ni siquiera conectarse.
  - Se le deben conceder **PERMISOS** o **PRIVILEGIOS**.
- Los permisos se conceden/revocan mediante las ordenes de SQL **GRANT/REVOKE** respectivamente.
  - Ver sintaxis SQL
- A un usuario se le pueden conceder 2 tipos de permisos o privilegios:
  - **PERMISOS SOBRE OBJETOS** ya existentes en la BD
  - **PERMISOS DE SISTEMA**

1.1.- CREACIÓN DE USUARIOS

- **PERMISOS SOBRE OBJETOS:** Se conceden a un usuario sobre objetos ya existentes en la BD.
  - Estos objetos pertenecen a algún usuario.
  - Permiten controlar el acceso a tales objetos.
  - **Ejemplo**: asignación de permiso de selección sobre la tabla *alumno.emple* a un usuario *pepito*.
- **PERMISOS DE SISTEMA**: Se conceden a un usuario para permitirle realizar ciertas operaciones sobre la BD
  - Afectan a la BD en general no a objetos particulares.
  - **Ejemplo**: concesión de permiso para CREAR TABLAS (CREATE TABLE) al usuario *pepito*.

1.1.- CREACIÓN DE USUARIOS

- Ver documentos
  - **Resumen PERMISOS DE ORACLE.**
    - Permisos de objeto
      - Dependen del tipo de objeto sobre el que se concede el permiso.
    - Permisos de sistema
      - Se suelen agrupar por tipos de objeto.
      - Su dinámica es un poco particular pero entenderla es sencillo.
  - **Sintaxis SQL**
    - PRIVILEGIOS O PERMISOS (SINTAXIS ORACLE)
      - concesion_privilegio_obj vs concesion_privilegio_sist

1.1.- CREACIÓN DE USUARIOS

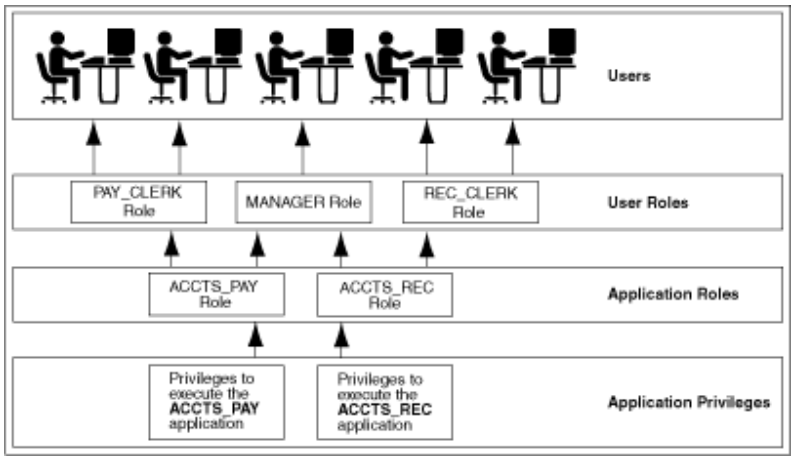
- **Ejemplo:**

```
CONNECT system/ADMIN123@pracs;

CREATE USER pruebas IDENTIFIED BY pruebas
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON USERS;

GRANT CREATE SESSION, CREATE USER TO pruebas;

GRANT SELECT ON ALUMNO.EMPLE TO pruebas;

CONNECT pruebas/pruebas@pracs;

CREATE USER otro IDENTIFIED BY otro
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA 500M ON USERS;

CONNECT otro/otro@pracs;
```


1.1.- CREACIÓN DE USUARIOS

- **ROL**: conjunto de privilegios agrupado **identificado con un nombre** que puede concederse a un usuario (o a otro rol)
  - La concesión de un rol implica la concesión "en bloque" de todos los privilegios que agrupa.
  - Los privilegios de un rol pueden ser de sistema y de objeto.
  - Permiten agrupar las tareas que pueden realizar ciertos tipos o grupos de usuarios.
  - La utilización de roles reduce enormemente el trabajo de administración de la seguridad en una BD.

1.1.- CREACIÓN DE USUARIOS



1.1.- CREACIÓN DE USUARIOS

ROLES PREDEFINIDOS

- Existen una serie de ROLES PREDEFINIDOS en Oracle listos para su utilización.
  - Estos roles y los permisos que contienen pueden variar de una versión a otra de Oracle
    - Se debe consultar siempre en la documentación de Oracle correspondiente a cada versión.
  - Los **roles predefinidos** más utilizados son los siguientes:
    - CONNECT
    - RESOURCE
    - DBA

1.1.- CREACIÓN DE USUARIOS

- Para la versión 11g de Oracle…
  - **CONNECT:**

CONNECT	Provides the CREATE SESSION system privilege. This role is provided for compatibility with previous releases of Oracle Database. You can determine the privileges encompassed by this role by querying the DBA_SYS_PRIVS data dictionary view.

  - **RESOURCE:**

RESOURCE	Provides the following system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE. This role is provided for compatibility with previous releases of Oracle Database. You can determine the privileges encompassed by this role by querying the DBA_SYS_PRIVS data dictionary view.

  - **DBA:**

DBA	Provides all system privileges WITH ADMIN OPTION. This role is provided for compatibility with previous releases of Oracle Database. You can determine the privileges encompassed by this role by querying the DBA_SYS_PRIVS data dictionary view.

1.1.- CREACIÓN DE USUARIOS

- Para la versión 9i de Oracle…

Table 10–1 Predefined Roles

Role Name	Created By (Script)	Description
CONNECT	SQL.BSQ	Includes the following system privileges: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE	SQL.BSQ	Includes the following system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
DBA	SQL.BSQ	All system privileges WITH ADMIN OPTION

Note: The previous three roles are provided to maintain compatibility with previous versions of Oracle and may not be created automatically in future versions of Oracle. Oracle Corporation recommends that you design your own roles for database security, rather than relying on these roles.

1.1.- CREACIÓN DE USUARIOS

ROLES PREDEFINIDOS:

- SYS y SYSTEM tienen asignado el rol DBA.
  - Pueden realizar todas las acciones sobre la BD
- Los roles predefinidos anteriores son los más usados y conocidos, **sin embargo**:
  - Se mantienen por compatibilidad.
  - A partir de Oracle 9 se recomienda, por seguridad, no utilizarlos y crear roles personalizados.
  - Existen múltiples referencias a esta recomendación en los distintos manuales y guías de la documentación de Oracle.

1.1.- CREACIÓN DE USUARIOS

- **Por ejemplo:** documentación de Oracle 11g…

> **Note:**
> Each installation should create its own roles and assign only those privileges that are needed, thus retaining detailed control of the privileges in use. This process also removes any need to adjust existing roles, privileges, or procedures whenever Oracle Database changes or removes roles that Oracle Database defines. For example, the `CONNECT` role now has only one privilege: `CREATE SESSION`.

> **Note:**
> As a security administrator, you should create your own roles and assign only those privileges that are needed. For example, many users formerly granted the `CONNECT` privilege did not need the additional privileges `CONNECT` used to provide. Instead, only `CREATE SESSION` was actually needed, and in fact, that is the only privilege `CONNECT` presently retains.
> Creating organization-specific roles gives an organization detailed control of the privileges it assigns, and protects it in case Oracle Database changes the roles that it defines in future releases.. Chapter 4, "Configuring Privilege and Role Authorization" discusses how to create and manage roles.

- Es posible consultar todos los roles predefinidos en Oracle 11 g en la documentación:
  - Roles predefinidos en Oracle 11g

1.1.- CREACIÓN DE USUARIOS

- **Ejemplo 1:**
  - Creación del usuario alumno en la BD PRACS
  - Concesión del rol DBA al usuario alumno para permitirle la administración de la BD

```
CONNECT system/ADMIN123@PRACS;

CREATE USER alumno IDENTIFIED BY alumno
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP;

GRANT dba TO alumno;
```

  - Ver scripts creación del usuario alumno

1.1.- CREACIÓN DE USUARIOS

- **Ejemplo 2:**
  - ¿Qué estamos haciendo?

```
CONNECT system/ADMIN123@PRACS;
CREATE ROLE new_rol;
GRANT CREATE SESSION,CREATE TABLE,CREATE USER,CREATE ROLE TO new_rol;
CREATE USER nuevo IDENTIFIED BY nuevo
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA 100M ON USERS;
GRANT new_rol TO nuevo;
```

  - ¿Qué puede hacer el usuario "***nuevo***"?

ÍNDICE

1.2.- MODIFICACIÓN DE USUARIOS

```
alteracion_usuario::= ALTER USER usuario
                      {[IDENTIFIED BY password]
                      [DEFAULT TABLESPACE tablespace]
                      [TEMPORARY TABLESPACE tablespace]
                      [comalista_cuota]
                      [PROFILE perfil]}
```

- Las cláusulas son las mismas que en CREATE USER
- Sólo los usuarios con el privilegio de sistema ALTER USER pueden modificar usuarios.
  - **Excepción**: cualquier usuario puede modificar su password sin gozar de este permiso.
- **Ejemplo**:

```
CONNECT system/ADMIN123@PRACS
ALTER USER alumno
QUOTA 1000 M ON USERS
```


ÍNDICE

1.3.- ELIMINACIÓN DE USUARIOS

```
eliminacion_usuario::= DROP USER usuario [CASCADE]
```

- Un usuario no podrá eliminarse si es propietario de algún objeto de la BD:
  - La opción CASCADE permite eliminar a un usuario junto con todos sus objetos.
  - Sólo los usuarios con el privilegio de sistema DROP USER pueden eliminar usuarios.

ÍNDICE

2.- TABLAS

2.1.- CREACIÓN DE TABLAS

definicion_tabla::= CREATE TABLE tabla (
comalista_*definicion_atributo*
[comalista_*restriccion_tabla*])

- tabla es el nombre de la tabla a crear.
- comalista_*definicion_atributo* define los atributos de la tabla.

definicion_atributo::= atributo {tipo_dato[1]| dominio}
[DEFAULT expresion]
[lista_*restriccion_atributo*]

- atributo es un nombre de atributo al que se le asigna un tipo de datos o un dominio predefinido (sólo en SQL estandar)
  - No pueden definirse dominios en Oracle.
- La cláusula DEFAULT permite especificar un valor por defecto para el atributo:
  - Este valor se asignará automáticamente al atributo en el caso de que no se le dé valor **al insertar una fila**.

1: Los tipos de datos permitidos dependerán del SGBD.

2.1.- CREACIÓN DE TABLAS

- Cláusula DEFAULT (continúa):
  - expresion se construye a partir de constantes, operadores y funciones del sistema utilizando la sintaxis apropiada:
    - Es decir, expresion puede ser:
      - una constante,
      - una expresión aritmética,
      - una función Oracle,
      - o bien, cualquier combinación válida de estas.
    - No puede hacer referencia a otros atributos de la tabla.
    - El tipo de datos resultado de la expresion debe coincidir con el tipo de datos del atributo.

2.1.- CREACIÓN DE TABLAS

- **EJEMPLOS**: (esquema CICLISMO)

```
CREATE TABLE EQUIPO (
NOMEQ    VARCHAR2 (25),
DIRECTOR VARCHAR2 (100)
)
```

```
CREATE TABLE CICLISTA (
DORSAL    NUMBER (3),
NOMBRE    VARCHAR2 (30)
          DEFAULT 'SIN NOMBRE',
EDAD      NUMBER (2)
          DEFAULT 18,
NOMEQ     VARCHAR2 (25)
)
```

2.- TABLAS

2.1- CREACIÓN DE TABLAS
2.1.1.- Restricciones

- Como ya sabemos, es posible definir una serie de restricciones sobre las tablas de una BD relacional:
  - Valor no nulo, clave primaria, clave ajena, etc.
  - Limitan los posibles estados de la BD.
- Al crear una tabla en SQL es posible definir 2 tipos de restricciones (**CONSTRAINTS**):
  - **De ATRIBUTO**: afectan a un solo atributo o columna.
    - Se definen cuando se define el atributo.
  - **De TABLA**: pueden afectar a varios atributos.
    - Se definen **al final de la definición de la tabla**.
    - Toda restricción de atributo o columna puede definirse siempre como restricción de tabla.

2.1- CREACIÓN DE TABLAS
2.1.1.- Restricciones

- En Oracle **las restricciones siempre tienen un nombre**.
  - Es recomendable poner nuestro propio nombre, ya que si no se indica Oracle lo elegirá por nosotros (sysxxxx).
  - Este nombre es utilizado para dar el mensaje de error cuando la restricción es violada.
    - Da muchas pistas sobre lo que está pasando y donde.
- Conviene seguir cierta homogeneidad al darles nombre. Algunas **recomendaciones** son:

TIPO RESTRICCIÓN	NOMBRE	EJEMPLO
CLAVES PRIMARIAS	tabla_PK	ciclista_PK
CLAVES AJENAS	tabla_tablareferenciada_FK	ciclista_equipo_FK
VNN	tabla_atributo_VNN	ciclista_nombre_VNN
CLAVES ALTERNATIVAS	tabla_atributo_UNI (si afecta a un solo atributo) tabla_nombre_UNI (si afecta a varios atributos)	ciclista_nomeq_UNI ciclista_equipounico_UNI
CHECK	tabla_atributo_CHK (si afecta a un solo atributo) tabla_nombre_CHK (si afecta a varios atributos)	ciclista_edad_CHK ciclista_rangoedad_CHK

2.- TABLAS

2.1.1.- RESTRICCIONES
2.1.1.1.- Restricciones de ATRIBUTO

```
definicion_atributo::= atributo {tipo_dato| dominio}
                       [DEFAULT expresion]
                       [lista_restriccion_atributo]
```

- lista_*restriccion_atributo* define una serie de restricciones sobre el atributo o columna (restricciones de atributo).

```
restriccion_atributo::=[CONSTRAINT restriccion]
                       {[NOT] NULL|
                       UNIQUE|
                       PRIMARY KEY|
                       REFERENCES tabla [(atributo)]
                              [directriz_borrado]
                              [directriz_actualizacion]|
                       CHECK (condicion)}
```

- Con la cláusula CONSTRAINT es posible darle un nombre a cada restricción (aunque es opcional es **MUY recomendable**[2])
  - Si no se le asigna un nombre generado automáticamente por Oracle.

2: El SGBD podrá utilizarlo en sus mensajes de error para indicar que restricción no se cumple.

2.1.1.1.- RESTRICCIONES DE ATRIBUTO

- La cláusula NOT NULL impide la presencia del valor nulo en el atributo o columna.
- La cláusula PRIMARY KEY especifica que el atributo es la clave primaria:
  - Si la clave primaria está formada por más de un atributo **NO podrá definirse como una restricción de atributo**.
    - Se crea como una restricción de tabla (ya lo veremos).
  - En Oracle al crear una CP se crea automáticamente:
    - Una restricción de valor no nulo (NOT NULL) en cada uno de sus atributos.
    - Un índice formado por todos los atributos de CP.

2.1.1.1.- RESTRICCIONES DE ATRIBUTO

- La cláusula UNIQUE permite definir al atributo como una clave alternativa o única:
  - No podrán existir dos filas en la tabla con el mismo valor en el atributo.
- La cláusula CHECK especifica una *condicion* que tendrá que satisfacer el valor que se asigne al atributo.
  - *condicion* es de la misma forma que en la cláusula WHERE pero con limitaciones:
    - Sólo puede hacer referencia al atributo de la restricción.
    - No puede incluir subconsultas ni funciones agregadas.
    - No puede utilizar las funciones SYSDATE, UID y USER.
  - Se satisface *condicion* si se evalúa a cierto o indefinido.
    - ¿Por qué creéis que es necesario esto?

2.1.1.1.- RESTRICCIONES DE ATRIBUTO

- **EJEMPLO:**

```
CREATE TABLE  EJEMPLO (
ID          NUMBER (3)
            CONSTRAINT EJEMPLO_PK PRIMARY KEY,
NOMBRE      VARCHAR2 (30)
            DEFAULT 'SIN NOMBRE'
            CONSTRAINT EJEMPLO_NOMBRE_VNN NOT NULL,
UNICO       NUMBER (3)
            DEFAULT 150
            CONSTRAINT EJEMPLO_UNICO_CHK
               CHECK (UNICO>=100 AND UNICO<200)
            CONSTRAINT EJEMPLO_UNICO_UNI UNIQUE,
FECHA       DATE
)
```

2.1.1.1.- RESTRICCIONES DE ATRIBUTO

- Mediante REFERENCES se indica que la columna es una clave ajena que referencia a tabla
  - atributo es el atributo que constituye la CP en tabla.
    - Si no se especifica se busca en tabla un atributo con el mismo nombre.
  - Las directrices de actualización y borrado permiten indicar al SGBD que debe hacer para restaurar la integridad cuando se viola por, una actualización o borrado respectivamente:

```
directriz_actualizacion::=  ON UPDATE
                                {CASCADE |
                                SET NULL |
                                SET DEFAULT |
                                NO ACTION}
```

```
directriz_borrado::= ON DELETE
                         {CASCADE |
                         SET NULL |
                         SET DEFAULT |
                         NO ACTION}
```


2.1.1.1.- RESTRICCIONES DE ATRIBUTO

Directrices de ACTUALIZACIÓN Y BORRADO

- En ambos casos hay 4 posibilidades:
  - **CASCADE**: borrado o modificación en CASCADA.
  - **SET NULL**: borrado o modificación A NULOS.
  - **SET DEFAULT**: Borrado o modificación fijando al valor por defecto en lugar de a nulos.
  - **NO ACTION**: no realiza ninguna acción y prohíbe la ejecución de operaciones que violen la integridad referencial.
    - valor asumido por defecto.
- Conviene indicar que en Oracle SÓLO es posible indicar las directrices de BORRADO CASCADE y SET NULL.
  - El resto de directrices no están soportadas (ni siquiera actualización en cascada).

2.1.1.1.- RESTRICCIONES DE ATRIBUTO

- **EJEMPLO**: (esquema CICLISMO)

```
CREATE TABLE CICLISTA (
DORSAL        NUMBER (3)
              CONSTRAINT CICLISTA_PK PRIMARY KEY ←
NOMBRE        VARCHAR2 (30)
              DEFAULT 'SIN NOMBRE'
              CONSTRAINT CICLISTA_NOMBRE_VNN NOT NULL ←
EDAD          NUMBER (2)
              DEFAULT 18
              CONSTRAINT CICLISTA_EDAD_CHK CHECK (EDAD>=18 AND EDAD<45) ←
NOMEQ         VARCHAR (25)
              CONSTRAINT CICLISTA_NOMEQ_VNN NOT NULL
              CONSTRAINT CICLISTA_EQUIPO_FK REFERENCES EQUIPO (NOMEQ)
                  ON DELETE CASCADE
)
```


2.- TABLAS

2.1.1.- RESTRICCIONES
2.1.1.2.- RESTRICCIONES DE TABLA

```
definicion_tabla::= CREATE TABLE tabla (
                        comalista_definicion_atributo
                        [comalista_restriccion_tabla] )

restriccion_tabla::=    [CONSTRAINT restriccion]
                        {UNIQUE (comalista_atributo)|
                        PRIMARY KEY (comalista_atributo)|
                        FOREIGN KEY (comalista_atributo)
                              REFERENCES tabla [(comalista_atributo)]
                              [directriz_borrado]
                              [directriz_actualizacion]|
                        CHECK (condicion)}
```


2.1.1.2.- RESTRICCIONES DE TABLA

- Las restricciones que pueden definirse a nivel de tabla son las mismas que a nivel de atributo
  - Excepto VNN que solo se puede definir a nivel de atributo
- **La diferencia de que pueden afectar a más de un atributo de la tabla**.
  - En todos los casos debe indicarse explícitamente la lista de atributos afectados por la restricción.
  - La *condicion* de la cláusula CHECK puede hacer referencia a varios atributos de la tabla (con las mismas limitaciones vistas a nivel de atributo).
  - En el caso de restricción de clave ajena debe indicarse la cláusula FOREIGN KEY (a nivel de atributo se omitía)
- **Toda restricción de atributo o columna** puede definirse a nivel de tabla.

2.1.1.2.- RESTRICCIONES DE TABLA

- **EJEMPLOS**: esquema ciclismo

```
CREATE TABLE LLEVAR (
NETAPA     NUMBER(2)
           CONSTRAINT LLEVAR_ETAPA_FK REFERENCES ETAPA (NETAPA),
CODIGO     CHAR(3)
           CONSTRAINT LLEVAR_MAILLOT_FK REFERENCES MAILLOT (CODIGO),
DORSAL     NUMBER(3)
           CONSTRAINT DORSAL_VNN NOT NULL
           CONSTRAINT LLEVAR_CICLISTA_FK REFERENCES CICLISTA (DORSAL),
CONSTRAINT LLEVAR_PK PRIMARY KEY (NETAPA, CODIGO)
)
```

- La siguiente definición es equivalente:

```
CREATE TABLE LLEVAR (
NETAPA   NUMBER(2),
CODIGO   CHAR(3),
DORSAL   NUMBER(3)
         CONSTRAINT DORSAL_VNN NOT NULL,
CONSTRAINT LLEVAR_PK PRIMARY KEY (NETAPA, CODIGO),
CONSTRAINT LLEVAR_ETAPA_FK FOREIGN KEY (NETAPA) REFERENCES ETAPA (NETAPA),
CONSTRAINT LLEVAR_MAILLOT_FK FOREIGN KEY (CODIGO) REFERENCES MAILLOT (CODIGO),
CONSTRAINT LLEVAR_CICLISTA_FK FOREIGN KEY (DORSAL) REFERENCES CICLISTA (DORSAL)
)
```


2.1.1.2.- RESTRICCIONES DE TABLA

- Las siguientes definiciones serían INCORRECTAS:

```
CREATE TABLE LLEVAR (
NETAPA     NUMBER(2)
           CONSTRAINT LLEVAR_PK PRIMARY KEY (NETAPA)
           CONSTRAINT LLEVAR_ETAPA_FK REFERENCES ETAPA (NETAPA),
CODIGO   CHAR(3)
           CONSTRAINT LLEVAR_PK PRIMARY KEY (CODIGO)
           CONSTRAINT LLEVAR_MAILLOT_FK REFERENCES MAILLOT (CODIGO),
DORSAL   NUMBER(3)
           CONSTRAINT LLEVAR_DORSAL_VNN NOT NULL
           CONSTRAINT LLEVAR_CICLISTA_FK REFERENCES CICLISTA (DORSAL)
)
```

```
CREATE TABLE LLEVAR (
NETAPA   NUMBER(2),
CODIGO   CHAR(3),
DORSAL   NUMBER(3)
         CONSTRAINT LLEVAR_DORSAL_VNN NOT NULL,
CONSTRAINT LLEVAR_PK PRIMARY KEY (NETAPA, CODIGO),
CONSTRAINT LLEVAR_FK FOREIGN KEY (NETAPA, CODIGO, DORSAL)
           REFERENCES ETAPA (NETAPA), MAILLOT (CODIGO), CICLISTA (DORSAL)
)
```

2.- TABLAS

2.1.- CREACIÓN DE TABLAS
2.1.2.- Cláusula TABLESPACE

- En Oracle las tablas se almacenan en tablespaces:
  - Al crear una tabla es posible indicar en que tablespace debe almacenarse mediante la cláusula TABLESPACE

```
definicion_tabla::= CREATE TABLE tabla (
                    comalista_definicion_atributo
                    [comalista_restriccion_tabla] )
                    [TABLESPACE nom_tablespace]
```

  - Esta cláusula no es estándar de SQL. Sólo de Oracle.
  - Es opcional. Si no se incluye la tabla se almacenará en:
    - El tablespace asignado por defecto para el usuario activo.
    - El tablespace SYSTEM si no se ha indicado ningún tablespace por defecto para el usuario activo.
      - SITUACIÓN NO RECOMENDABLE!

2.- TABLAS

2.1.- CREACIÓN DE TABLAS
2.1.3.- A partir de consultas. Clonado

- Oracle permite crear una tabla a partir de una consulta:

```
definicion_tabla::= CREATE TABLE tabla
                    [(comalista_atributo)]
                    [TABLESPACE nom_tablespace]
                    AS sentencia_select
```

  - Esta posibilidad no es estándar de SQL.
  - La nueva tabla hereda las características de las columnas recuperadas en la consulta
    - Incluso los nombres de los atributos si estos se omiten en comalista_atributo
    - Las restricciones definidas en las tablas originales también son creadas en la nueva tabla
      - Sólo si el propietario **NO** le ha asignado un nombre al crearla
      - Para volverse loco!
  - Esta opción es muy útil para realizar clonado de tablas.

2.1.- CREACIÓN DE TABLAS
2.1.3.- A partir de consultas. Clonado

- **Ejemplo**:

```
CREATE TABLE copia_ciclista AS
SELECT *
FROM  CICLISTA
```


2.- TABLAS

2.2.- ELIMINACIÓN DE TABLAS

- **Sintaxis estándar vs sintaxis Oracle:**

eliminacion_tabla::= DROP TABLE tabla
{RESTRICT |CASCADE}

eliminacion_tabla::= DROP TABLE [usuario.]tabla
[CASCADE CONSTRAINTS]

- En principio, un usuario sólo puede eliminar las tablas de las que es propietario.
  - Los usuarios con el privilegio (de sistema) DROP ANY TABLE también pueden borrar las tablas de cualquier otro usuario.
  - El rol de administrador (dba) tiene asignado este privilegio.
- La opción CASCADE CONSTRAINTS elimina todas las restricciones de clave ajena que referencien a la tabla eliminada.
  - Si no se indica y existen claves ajenas a la tabla que pueden violarse la eliminación fallará.

2.- TABLAS

2.3.- MODIFICACIÓN DE TABLAS

- **Sintaxis de Oracle**: (sólo vemos esta)

```
alteracion_tabla::=   ALTER TABLE tabla
                      {ADD (comalista_definicion_atributo) |
ATRIBUTOS             MODIFY (comalista_definicion_atributo) |
                      DROP (comalista_atributo) [CASCADE CONSTRAINTS] |
                      RENAME COLUMN atributo_old TO atributo_new |
RESTRICCIONES         ADD (comalista_restriccion_tabla) |
                      DROP ref_restriccion}

ref_restriccion::=    PRIMARY KEY [CASCADE] |
                      UNIQUE (comalista_atributo) [CASCADE] |
                      CONSTRAINT restriccion [CASCADE]
```

- **ADD**: añade uno o más atributos al final de la tabla o una restricción de tabla.
- **MODIFY**: modifica uno o más atributos existentes.
- **DROP**: elimina uno o más atributos o una restricción.
- **RENAME COLUMN**: renombra un atributo.

2.3.- MODIFICACIÓN DE TABLAS

- **Clausulas asociadas a ATRIBUTOS:**
  - **ADD**: añade a la tabla uno o más atributos definidos por comalista_*definicion_atributo* al final de la tabla.
    - Todas las filas existentes se rellenan a NULL a no ser que se indique un valor por defecto.
  - **MODIFY**: modifica uno o más atributos existentes.
    - A través de esta clausula no es posible añadir/eliminar restricciones distintas de VNN.
    - Si permite cambiar el tipo o definir valores por defecto.
  - **DROP**: elimina uno o más atributos.
    - **CASCADE CONSTRAINTS** elimina las restricciones asociadas a los atributos eliminadas[3].
      - **Ejemplo**: tabla llevar (atributo netapa).

3: Realmente sólo hace falta para eliminar las restricciones de clave ajena o multicolumna que las referencian

2.3.- MODIFICACIÓN DE TABLAS

- **Clausulas asociadas a RESTRICCIONES:**
  - **ADD**: añade una o varias restricciones de tipo tabla a la tabla (aunque solo afecten a un atributo).
    - Según la sintaxis comalista_*restriccion_tabla*
      - Sólo permite UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
  - **DROP**: elimina una restricción

```
ref_restriccion::=    PRIMARY KEY [CASCADE] |
                      UNIQUE (comalista_atributo) [CASCADE] |
                      CONSTRAINT restriccion [CASCADE]
```

    - Primaria
    - Única
    - Por su nombre.
    - CASCADE elimina las claves ajenas que las referencian
      - Cuando proceda.

2.3.- MODIFICACIÓN DE TABLAS

- **Ejemplos**: ¿Qué hacen estas sentencias SQL?

1	`ALTER TABLE CICLISTA` `ADD (FECHA_ALTA DATE DEFAULT SYSDATE)`
2	`ALTER TABLE CICLISTA` `DROP (EDAD, FECHA_ALTA) CASCADE CONSTRAINTS`
3	`ALTER TABLE CICLISTA` `MODIFY (EDAD NUMBER(2) DEFAULT 20 NOT NULL)`
4	`ALTER TABLE CICLISTA` `RENAME COLUMN edad TO años`
5	`ALTER TABLE CICLISTA` `DROP PRIMARY KEY CASCADE`
6	`ALTER TABLE CICLISTA` `DROP CONSTRAINT CICLISTA_EQUIPO_FK`

2.3.- MODIFICACIÓN DE TABLAS

- **CONSIDERACIONES FINALES**:
  - Conviene tener en cuenta que un ALTER TABLE puede fallar ya que estamos modificando una tabla existente que puede tener datos.

CUESTIONES:

- Analiza distintos casos en los que ALTER TABLE puede fallar:
  - Al añadir un atributo.
  - Al modificar un atributo.
- ¿Cómo añadirías un valor por defecto a un atributo?
  - ¿Cómo afecta esto al atributo?
- ¿Cómo eliminarías un valor por defecto de un atributo?
- ¿Cómo eliminarías una restricción de valor no nulo de un atributo? (propón al menos 2 formas distintas)

ÍNDICE

3.- OTROS OBJETOS DE LA BD

- Las sentencias de CREATE, DROP y ALTER también se utilizan para la creación de otros objetos de la BD
  - Vistas
  - Índices
  - Sinónimos
  - Procedimientos almacenados
  - Triggers
  - Tablespaces
  - Etc.
- El DDL de SQL se encarga de todos los objetos y estructuras de la BD.
- La sintaxis de SQL que tenéis incluye estos objetos.