B3

Lenguajes de Bases de Datos Relacionales

Tema 7. SQL: Definición de Datos (DDL)

Tema 7. SQL: Definición de Datos

Objetivos

- Conocer los **orígenes** e historia del lenguaje ANSI **SQL**
- Diferenciar entre Modelo Relacional formal y el ANSI SQL
- Aprender la **sintaxis** del **SQL** con el fin de escribir sentencias de *definición*, *modificación* o *eliminación* de estructuras de datos

  Data Definition Language (DDL) del SQL

  - Distinguir entre las sentencias de definición de datos (**DDL**) y las de manipulación de datos (**DML**) del lenguaje SQL
  - Diferenciar entre el lenguaje ANSI SQL y los **dialectos SQL** implementados por sistemas de bases de datos comerciales

7. SQL: Definición de Datos

Bibliografía

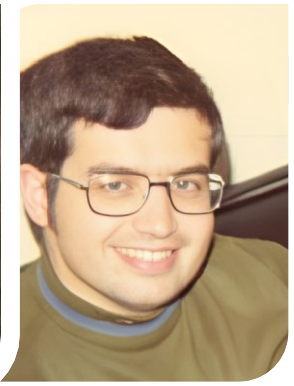
- [CB 2015] Connolly, T.M.; Begg C.E.: ***Database Systems: A Practical Approach to Design, Implementation, and Management***, 6th Edition. Pearson. (Capítulos 6 y 7)
- [EN 2016] Elmasri, R.; Navathe, S.B.: ***Fundamentals of Database Systems***, 7th Edition. Pearson. (Caps. 6 y 7)

7. SQL: Definición de datos

Contenidos

- 7.1 Introducción, orígenes e historia del SQL
- 7.2 Definición de Datos
  - Creación , alteración y destrucción de tablas

7.1 Introducción, orígenes e historia del SQL



- ***S**tructured **Q**uery **L**anguage*
- Primer lenguaje de BD de alto nivel
  - Diseñado e implementado en el *IBM's Research Laboratory* (San José, California), para el SGBD Relacional experimental "*System R*"
  - A principios de la década de 1970
  - Por Donald D. **Chamberlin** y Raymond F. **Boyce** después de conocer el *modelo relacional* de Edgar F. Codd
- Primeras implementaciones: en 1979
  - **Oracle** y poco después INGRES
- Definición de un **lenguaje estándar** para SGBDR

ORACLE®

INGRES

ANSI (*American National Standards Institute*)
+ **ISO** (*International Standardization Organization*)
+ **IEC** (International Electrotechnical Commission)

ANSI

7.1 Introducción, orígenes e historia del lenguaje SQL

- Evolución del estándar SQL (**ANSI SQL**)
  ISO/IEC 9075 (ANSI X3.135)

Nombre	Comentarios
SQL-86	Primera formalización de ANSI
SQL-89	Restricciones de integridad
SQL-92	Primera versión estable que incluye el 'core' del estándar SQL, el cual **no ha cambiado** en las revisiones posteriores
SQL:1999	Añade expresiones regulares, disparadores, algunas características de orientación a objetos…
SQL:2003	Incluye características XML, funciones analíticas, secuencias, …
SQL:2006	Uso conjunto de SQL y XML, integración de XQuery en SQL, …
SQL:2008	Incluye disparadores INSTEAD OF, sentencia TRUNCATE, …
SQL:2011	Añade datos temporales, mejoras en cláusula FETCH, …
SQL:2016	Añade búsquedas de patrones, funciones de tabla polimórficas, compatibilidad con JSON. Es la **versión actual** del estándar
SQL-2019	Aún está en borrador (draft) Añade arrays multidimensionales

7.1 Introducción, orígenes e historia del lenguaje SQL

- Lenguaje de BD completo (no sólo de consulta), cuyos comandos se organizan en...

Lenguaje de Definición de Datos (LDD)

- Órdenes para **crear, modificar** y **eliminar** estructuras de datos (**tablas, vistas,...**)
- CREATE, ALTER, DROP, ...

Lenguaje de Manipulación de Datos (LMD)

- Comandos para **introducir datos, actualizarlos** y **eliminarlos**
- Extraer (recuperar o **seleccionar**) datos almacenados
- INSERT, UPDATE, DELETE y SELECT

Otros...

- Incorporación de **SQL dentro de código** escrito con un Lenguaje de Programación de propósito general (Pascal, C, etc.), etc.

7.1 Introducción, orígenes e historia del lenguaje SQL

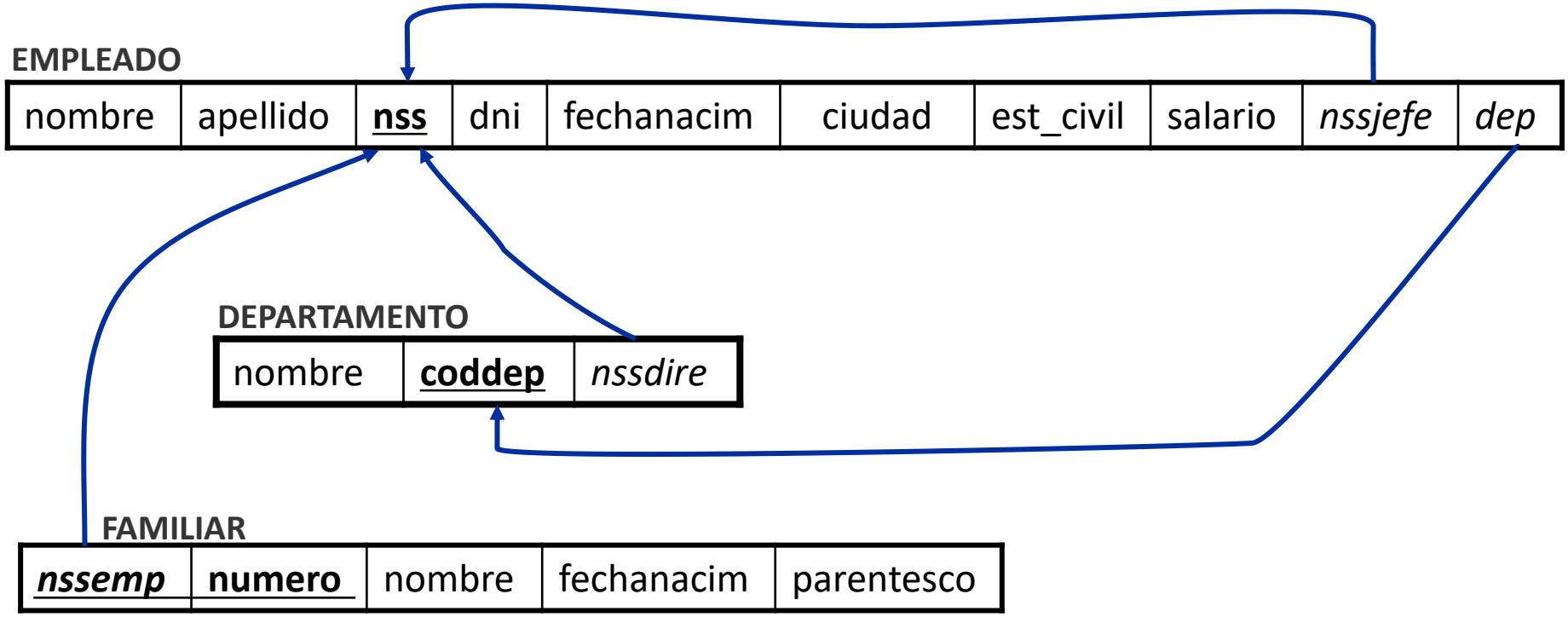
ANSI SQL

- Utiliza los términos **tabla, columna, fila, …**
- Una tabla **puede** tener **filas idénticas**
  - Permite definir *tablas sin claves*: las filas se identifican internamente, a nivel físico
  - En general *tabla = 'bag' de filas*
  - Para que una tabla no admita filas repetidas, definir restricciones de clave primaria y alternativas
- **Columnas** de tabla **ordenadas** en orden de creación
- Es posible indicar un **orden de visualización** de las filas
- Una **clave ajena** puede referenciar a una **clave primaria o** bien a una clave **alternativa** (UNIQUE)

Modelo Relacional formal

- Utiliza los términos formales **relación, atributo, tupla, …**
- Una relación **jamás** contiene **tuplas repetidas**
  - Relación = conjunto (*set*) de tuplas
- **Atributos** de relación **no ordenados**
- Una **clave ajena** sólo puede referenciar a una **clave primaria**

SQL. *Running Example*



- Un esquema (simplificado) de base de datos, "Empresa"
  - Las referencias (clave ajena/clave primaria) están representadas mediante flechas
  - Las claves primarias están subrayadas

7.2 Definición de Datos en SQL

- El *Diseño Lógico Específico* consiste en *escribir el Esquema Lógico con la sintaxis propia del modelo de datos particular del SGBD comercial elegido*
  - Hemos elegido ***Oracle***, así que debemos conocer el dialecto SQL que ofrece para poder definir tablas
  - Pero no tiene sentido que estudiemos "sólo un SQL específico": hemos de estudiar el **SQL estándar ANSI**, pues todos los SQL comerciales se basan en él. De este modo, seremos capaces de aprender rápidamente cualquier dialecto SQL de cualquier SGBD comercial con el que tengamos que trabajar
- Sentencias o instrucciones que permiten definir (crear) nuevas tablas, alterar su estructura y eliminarlas:
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE

Creación de tablas

- **Sentencia CREATE TABLE**
  - Define (crea) una tabla: nombre, columnas y restricciones
  - **Nombre** único dentro del esquema
  - Para cada **Columna** hay que especificar…
    - **nombre,**
    - **tipo de datos**
    - **restricciones** de columna
  - **Restricciones** de tabla…
    - de **clave** candidata,
    - de **integridad de entidad,**
    - de **integridad referencial,** o
    - restricciones de **otro tipo**

```
CREATE TABLE empleado (
 nombre …,
 apellido …,
 nss …,
 dni …,
 fechanacim …,
 ciudad …,
 est_civil …,
 salario …,
 dep …,
    nssjefe …,
…
);
```

Creación de tablas

- **Ordenamiento** de columnas y filas
  - Una vez creada la tabla, sus **columnas** quedan **ordenadas** tal como aparecen en la instrucción CREATE TABLE
  - Las **filas** (contenido) **no** están **ordenadas** de ningún modo concreto
    - Quedan en el orden de **inserción**

```
CREATE TABLE empleado (
  nombre ...,
  apellido ...,
  nss ...,
  dni ...,
  fechanacim ...,
  ciudad ...,
  est_civil ...,
  salario ...,
  dep ...,
  nssjefe ...,
  ...
);
```

orden

Tabla EMPLEADO

nombre	apellido	nss	dni	fechanacim	ciudad	est_civil	salario	dep	nssjefe	...

- Las tablas creadas con CREATE TABLE son denominadas ***tablas base***
  - Esto significa que el SGBD las almacena físicamente en algún fichero de la base de datos en disco

Definir **columnas de tabla**

- En la sentencia CREATE TABLE cada columna se define en una línea
  - La definición termina con una coma: ,
- Para cada columna hay que especificar...
  - El **nombre** de la columna
  - El **tipo de datos**, que se elige entre los tipos de datos ofrecidos por SQL
  - Y **restricciones** de columna

```
CREATE TABLE empleado (
 nombre   VARCHAR(15)  NOT NULL,
 ...
 salario  NUMERIC(6,2) NOT NULL,
 dep      CHAR(3)      NULL,
 ... );
```

Este código es *SQL estándar*: los **tipos de datos** son del ANSI SQL (lenguaje teórico). En la práctica hay que usar los que ofrece el SGBD que se utilice (***Oracle***, *MySQL*, etc.)

Tipos de Datos de columna ANSI SQL

Estos **tipos de datos** son del **ANSI SQL** (teórico)

- **Numéricos**
  - Enteros y Reales
    - **INTEGER** (también INT), SMALLINT,
    - REAL (simple precisión), DOUBLE PRECISION, FLOAT(p)
  - Con formato
    - **NUMERIC(p,e)** o DECIMAL(p,e) (también DEC(p,e))

      p: precisión (número total de dígitos del número)
      e: escala (cuantos dígitos, de los **p** totales, son decimales); el valor por omisión de e=0
      Ejemplo: NUMERIC(7,2) corresponde a números con 7 dígitos en total, de los cuales 2 son decimales: 99.999,99

Tipos de Datos de columna ANSI SQL

- **Cadena de caracteres**
  - Longitud fija **CHAR(n)** n: n° de caracteres; por omisión n=1
  - Longitud variable **VARCHAR(n)** n: máximo n° de caracteres

Estos **tipos de datos** son del **ANSI SQL** (teórico)

- **Cadena de Bits**
  - Longitud fija **BIT(n)** n: n° de bits; por omisión n=1
  - Longitud variable **BIT VARYING(n)** n: máximo n° de bits

Tipos de Datos de columna ANSI SQL

- **Temporales**
  - **DATE** (10 posiciones) = YEAR, MONTH, DAY (yyyy-mm-dd)
  - **TIME** (8 posiciones) = HOUR, MINUTE, SECOND (hh:mi:ss)
    - Sólo permitidas fechas y horas válidas
  - **TIMESTAMP** (marca de tiempo)
    - DATE, TIME, fracciones de segundo y (si se incluye WITH TIME ZONE) desplazamiento respecto al UTC (huso horario estándar)
  - **INTERVAL**
    - Período de tiempo, para incrementar/decrementar el valor actual de una fecha, hora o marca de tiempo
    - Se califica con YEAR/MONTH o DAY/TIME para indicar su naturaleza

Estos **tipos de datos** son del **ANSI SQL** (teórico)

Oracle. Tipos de Datos nativos

- De ***cadena de caracteres***
  - **CHAR**[(size)] -- por omisión *size=1*
  - NCHAR[(size)]
  - **VARCHAR2**(size)
  - NVARCHAR2(size)
  - LONG --anticuado, usar CLOB
- ***Numéricos***
  - **NUMBER**[(**p**[,**s**])] -- por omisión p=38 y s=0
    - INTEGER -- Oracle lo convierte a NUMBER(38,0)
- De ***punto flotante***
  - FLOAT[(p)]
  - BINARY_FLOAT --32bits
  - BINARY_DOUBLE --64bits

Estos **tipos de datos** del SQL de Oracle **SÍ** son los que podremos usar en las prácticas

Oracle. Tipos de Datos nativos

- ***Fechas y Tiempo***
  - **DATE** -- incluye DATE y TIME de ANSI SQL
  - TIMESTAMP[(fractional_seconds_precision)]
  - TIMESTAMP[(fractional_seconds_precision)] WITH TIME ZONE
  - TIMESTAMP[(fractional_seconds_precision)] WITH LOCAL TIME ZONE
- ***Intervalos*** de tiempo
  - INTERVAL YEAR [(year_precision)] TO MONTH
  - INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)]

Estos **tipos de datos** del SQL de Oracle **SÍ** son los que podremos usar en las prácticas

Oracle. Tipos de Datos nativos

- ***Datos crudos***
  - Datos binarios o cadenas de bytes
  - Gráficos, documentos...
  - RAW(size)
  - LONG RAW -- mejor BLOB
- ***Objetos grandes***

  (LOB: **L**arge **OB**jects)
  - BLOB
  - CLOB
  - NCLOB
  - BFILE -- objeto almacenado fuera de la BD
- De ***identificación de filas***
  - ROWID
  - URWOID[(size)]

A partir de ahora, en los ejemplos de las diapositivas de la asignatura usaremos y las sentencias y tipos de datos de ***Oracle SQL***, en vez de ANSI SQL, para evitar confusiones en las prácticas

Definir Restricciones de Columna

- Además del nombre y del tipo de datos, la definición de una columna puede incluir **restricciones** de integridad que afectan a los **valores de esa columna:**
- **Cláusula NULL o NOT NULL**
  - Indica si una columna puede contener NULL o no

```
CREATE TABLE empleado (
...
nombre VARCHAR2(25) NOT NULL,
...
dep   CHAR(3)     NULL,
... );
```

  - Por **omisión** (si no se indica nada), se asume **NULL**
  - Excepto para las columnas componentes de una **clave primaria** que, como deben ser NOT NULL, para ellas se asume **NOT NULL** por omisión

¡Restricción de Integridad de Entidad!

Definir Restricciones de Columna

- **Cláusula DEFAULT *valor***
  - Justo detrás del tipo de datos y antes de cualquier otra restricción de columna

    ⓘ En *Oracle SQL* si se pone en otro lugar surge error de sintaxis

```
CREATE TABLE empleado (
...
salario NUMBER(6,2) DEFAULT 1000 NOT NULL,
... );
```

  - Si una columna no tiene DEFAULT, su valor por defecto es
    - NULL, si está definida como que admite NULL
    - Ninguno, si está definida como NOT NULL
      - No admitirá valor por defecto: al insertar una fila, siempre habrá que darle un valor a la columna

Definir **Restricciones de Tabla**

- Además de las columnas, la sentencia CREATE TABLE incluye la definición de **restricciones** de integridad **que afectan** al contenido de la tabla, es decir **a todas las filas:**
  - Cuál es la clave primaria
  - Cuáles son las claves alternativas (si las hay)
  - Qué columnas son claves ajenas (si las hay)
  - Qué otras restricciones deben cumplir (si las hay) los valores de ciertas columnas
- Cada restricción se define en una línea
  - Y finaliza con una coma, salvo la última

Definir Restricciones de Tabla

- **Cláusula PRIMARY KEY (*lista_columnas*)**
  - Indica qué columnas componen la **clave primaria** (PK)
    - Sus **valores** (concatenados, si es compuesta) siempre serán **únicos** dentro de la tabla
  - Y también asegura que **ningún componente** de la PK puede contener **NULL**
    - Se asume **NOT NULL** por omisión para cada columna componente de la PK
    - ☑ Restricción de Integridad de Entidad

```
CREATE TABLE estudiante (
 DNI  CHAR(9) NOT NULL,
 ...
 PRIMARY KEY(DNI),
...);
```

★ Un CREATE TABLE puede incluir sólo una cláusula PRIMARY KEY

Definir Restricciones de Tabla

- **Cláusula UNIQUE (*lista_columnas*)**
  - Indica qué columnas forman una **clave alternativa**
    - Sus **valores** (concatenados, si es compuesta) siempre serán **únicos** dentro de la tabla
  - Hay que incluir una cláusula UNIQUE distinta para cada una de las claves alternativas que tenga la tabla

```
CREATE TABLE estudiante (
 DNI       CHAR(9)     NOT NULL,
 expediente NUMBER(6)    NOT NULL,
 nombre    VARCHAR2(60) NOT NULL,
 email     VARCHAR2(35) NOT NULL,
 PRIMARY KEY(DNI),
 UNIQUE (expediente),
 UNIQUE (email),
...);
```

★ Un CREATE TABLE puede incluir ninguna, una o muchas cláusulas UNIQUE

Definir Restricciones de Tabla

- Se permite que una clave UNIQUE **contenga** el **NULL**
  - La(s) columna(s) se define(n) como NULL en el CREATE TABLE
- Una clave UNIQUE puede ser **compuesta**

```
CREATE TABLE paciente(
   num_historial CHAR(15)          NOT NULL,
   DNI                    CHAR(9)          NULL,
   nombre                 VARCHAR2(60)  NOT NULL,
   f_nacimiento  DATE                NOT NULL,
   NSS                    NUMBER(12)    NOT NULL,
   PRIMARY KEY(num_historial),
   UNIQUE (DNI),
   UNIQUE (NSS, f_nacimiento)
   );
```

Dos pacientes pueden tener el mismo NSS (por ejemplo, varios niños hermanos que como pacientes usan el NSS de uno de sus progenitores). Por eso NSS en solitario no es clave

  - DNI es clave alternativa y admite NULL, p.ej. para pacientes de corta edad, para quienes no se ha expedido el DNI

Definir Restricciones de Tabla

- **Cláusula CHECK (*expresión*)**
  - Para especificar **restricciones adicionales**
  - Expresa una condición sobre los valores de una o varias columnas que debe cumplir toda fila de la tabla

**El salario de un empleado está entre 600 y 4000€*

```
CREATE TABLE empleado (
...,
 CHECK ( salario >= 600 AND salario <= 4000 ),
... );
```

  - Puede definir restricciones que implican a varias columnas de la misma tabla

**Toda película se estrena tras finalizar su rodaje*

```
CREATE TABLE pelicula (
...,
 CHECK ( fecha_fin_rodaje < fecha_estreno ),
... );
```

Oracle. Restricciones de Tabla

- La condición de la **cláusula CHECK** tiene *fuertes* ***limitaciones***:
  - Debe ser una **expresión booleana** evaluable que use los valores en la fila que está siendo insertada o actualizada.
  - **Sólo puede incluir comparaciones entre columnas de la tabla, o entre una columna y un valor constante**
  - **No** puede contener **subconsultas**.
  - **Tampoco** puede incluir **funciones no deterministas** (SYSDATE, SYSTIMESTAMP, CURRENT_DATE, CURRENT_TIMESTAMP, DBTIMEZONE, LOCALTIMESTAMP, SESSIONTIMEZONE, UID, USER y USERENV).
  - **No** puede hacer referencia a **columnas de otras tablas**.
  - **No** puede contener **constantes de tipo fecha** que **no** estén **completamente especificadas**.
  - **No** puede incluir llamadas a **funciones definidas por el usuario**.
  - **No** puede contener las **pseudocolumnas** LEVEL, ROWNUM, NEXTVAL, CURRVAL.
  - …

Definir Restricciones de Tabla

- **Cláusula FOREIGN KEY (*lista_columnas*) REFERENCES *tabla(columnas_clave)***
  - Define qué columnas forman una **clave ajena** o externa
  - Y a qué **tabla** y **clave** referencia

```
CREATE TABLE empleado (
...
dep    CHAR(3) NULL,
...,
FOREIGN KEY (dep) REFERENCES departamento(coddep),
...
);
```

Tabla referenciada

Clave Ajena (columna de EMPLEADO)

Clave (columna de DEPARTAMENTO)

  - El ANSI SQL y *Oracle SQL* permiten que una clave ajena haga referencia a una clave …
    - Primaria (PRIMARY KEY), o bien a
    - Alternativa (UNIQUE)

Pero esto es MUY poco habitual

Ejemplo de definición de una tabla


```
CREATE TABLE empleado (
   nombre          VARCHAR2(25)      NOT NULL,
   apellido        VARCHAR2(15)      NOT NULL,
   nss             NUMBER(12)        NOT NULL,
   dni             CHAR(9)           NOT NULL,
   fechanacim      DATE              NULL,
   ciudad          VARCHAR2(30)      ,
   est_civil       CHAR(1)           ,
   salario         NUMBER(6,2)       DEFAULT 1000 NOT NULL,
   dep             CHAR(3)           NULL,
   nssjefe         NUMBER(12)        ,
   cuantos_familiares NUMBER(2) DEFAULT 0 NOT NULL,
   PRIMARY KEY (nss),
   UNIQUE (dni),
   FOREIGN KEY (dep) REFERENCES departamento(coddep),
   FOREIGN KEY (nssjefe) REFERENCES empleado(nss),
   CHECK (nssjefe <> nss),
   CHECK (est_civil IN ('S','C','V','D','P')),
   CHECK (salario > 0)
);
```

Se puede omitir NOT NULL.
Al ser clave primaria, se asume

Se asume NULL por omisión

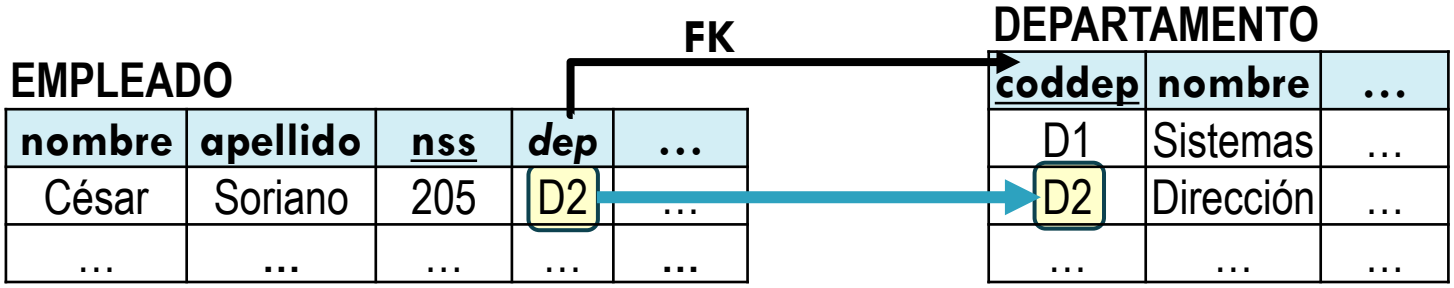
El tipo de datos de una clave ajena debe ser el mismo que el de la clave primaria referenciada

Aquí ya no se pone la coma por ser la última.
A continuación (línea siguiente) va el paréntesis de cierre y el ;

Definir Claves Ajenas

- Cada clave ajena (FK) debe estar formada por tantas **columnas** como tenga la clave a la que referencia
- Cada *columna de la FK* debe tener el mismo **tipo de datos** que la *columna correspondiente* de la clave *a la que referencia*

- Al definir una clave ajena mediante **FOREIGN KEY el SGBD garantiza la *Integridad Referencial***
  - Los valores (no nulos) de la clave ajena siempre deberán existir en clave de la tabla a la que referencia



Definir Claves Ajenas

- Pero **hay operaciones que pueden romper la Integridad Referencial** y dejarían la BD en un estado incorrecto
  - Ejemplo: en CUENTA, la columna titular es una clave ajena que referencia a CLIENTE(codigo)

CLIENTE

codigo	nombre	direccion	ciudad
1210	García, A.	Gran Vía, 6	Murcia
0300	Zapata, D.	Ronda Norte, 3	Murcia
1003	Arjona, C.	Paseo Rosales, 9	Molina
2689	Sancho, B.	Plaza Mayor, 2	Patiño
3679	Burgos, C.	Camino Viejo, 20	Yecla

CUENTA

numero	saldo	*titular*
200	85.005	2689
505	40.000	1003
821	50.000	1210
426	35.620	1003
005	29.872	2689
315	3.500	0300

- ¿Qué pasaría si en CLIENTE
  - se **elimina** la **fila** del **cliente** 1003 ?
  - se **cambia el código del cliente** 2689 al valor 3030 ?

Definir Claves Ajenas

- 1) En CLIENTE se **elimina la fila** del cliente 1003

CLIENTE

codigo	nombre	direccion	ciudad
1210	García, A.	Gran Vía, 6	Murcia
0300	Zapata, D.	Ronda Norte, 3	Murcia
~~1003~~	~~Arjona, C.~~	~~Paseo Rosales, 9~~	~~Molina~~
2689	Sancho, B.	Plaza Mayor, 2	Patiño
3679	Burgos, C.	Camino Viejo, 20	Yecla

CUENTA

numero	saldo	*titular*
200	85.005	2689
505	40.000	1003
821	50.000	1210
426	35.620	1003
005	29.872	2689
315	3.500	0300

- ¡Hay filas en CUENTA que le hacían referencia! ***¿Qué pasa con ellas?***
- No se pueden quedar así, porque la *Restricción de Integridad Referencial* no permite que en CUENTA.titular haya valores que no existan en CLIENTE.codigo

Definir Claves Ajenas

- 2) En CLIENTE se **cambia el valor de la clave primaria** (codigo) de 2689 a 3030

CLIENTE

codigo	nombre	direccion	ciudad
1210	García, A.	Gran Vía, 6	Murcia
0300	Zapata, D.	Ronda Norte, 3	Murcia
1003	Arjona, C.	Paseo Rosales, 9	Molina
3030	Sancho, B.	Plaza Mayor, 2	Patiño
3679	Burgos, C.	Camino Viejo, 20	Yecla

CUENTA

numero	saldo	*titular*
200	85.005	**2689**
505	40.000	1003
821	50.000	1210
426	35.620	1003
005	29.872	**2689**
315	3.500	0300

- Hay filas en CUENTA que le hacían referencia mediante ese valor ***¿Qué se hace con ellas?***
- Tampoco se pueden quedar así: la *Restricción de Integridad Referencial* no permite valores en CUENTA.titular **no existentes** en CLIENTE.codigo

Definir Claves Ajenas

- ¿Cómo puede evitar el SGBD esos estados incorrectos?
  ¿Cómo **mantener la integridad referencial**?

- Al definir cada clave ajena hay que indicar las **acciones de mantenimiento de la integridad referencial**, o **acciones referenciales**, teniendo en cuenta su semántica
- En la cláusula **FOREIGN KEY** hay que añadir las cláusulas **ON DELETE** *acción*
  y **ON UPDATE** *acción*

acción ∈ { NO ACTION, CASCADE, SET NULL, SET DEFAULT }

Veamos esto del **mantenimiento de la integridad referencial** con más detenimiento

Mantenimiento de la Integridad Referencial

R2 ⟶ R1

Operación: **Eliminar** una **fila** t de R1 que está **referenciada por otras** de R2

Acciones posibles (AL BORRAR, **ON DELETE**)

1. **Rechazar** la operación (acción por defecto: la no-acción)
   - Sólo permite borrar t si ninguna otra fila hace referencia a t
2. **Cascada**. Propagar la eliminación
   1° Borrar todas las filas de R2 que referencian a t
   2° Eliminar la fila t de R1
3. **Establecer nulo** 👁 Sólo si se definió que la FK en R2 permite NULL
   1° Toda fila de R2 que referencia a t pasa a contener NULL en la FK
   2° Eliminar la fila t de R1
4. **Establecer valor por defecto** 👁 Sólo si existe DEFAULT para la FK
   1° Toda fila de R2 que referencia a t toma su valor por defecto en las columnas FK
   2° Eliminar la fila t de R1

Mantenimiento de la Integridad Referencial

CUENTA ⟶ CLIENTE

- **Ejemplo 1 ON DELETE**
  - En CLIENTE se **elimina la fila** del cliente 1003

CLIENTE

codigo	nombre	direccion	ciudad
1210	García, A.	Gran Vía, 6	Murcia
0300	Zapata, D.	Ronda Norte, 3	Murcia
1003	Arjona, C.	Paseo Rosales, 9	Molina
2689	Sancho, B.	Plaza Mayor, 2	Patiño
3679	Burgos, C.	Camino Viejo, 20	Yecla

CUENTA

numero	saldo	*titular*
200	85.005	2689
505	40.000	1003
821	50.000	1210
426	35.620	1003
005	29.872	2689
315	3.500	0300

- No tiene sentido que haya cuentas con cliente NULL
- Ni borrar automáticamente todas las cuentas (¡perder esos datos!) de un cliente si es eliminado (¡quizá por error!)
- Lo correcto: no permitir borrar un cliente si tiene cuentas
- La acción debe ser **rechazar: ON DELETE NO ACTION**

Mantenimiento de la Integridad Referencial

SALON_HOTEL ⟶ HOTEL

- **Ejemplo 2 ON DELETE**
  - En HOTEL se **elimina la fila** del hotel H01

HOTEL

codigo	nombre	direccion	ciudad
~~H01~~	~~Gran Hotel~~	~~Gran Calle, 4~~	~~Murcia~~
H02	Las Colinas	Ronda Sureste, 3	Murcia
H03	Triton Hotel	Paseo Salado, 9	Jumilla
H04	Las Salinas	Plaza Sola, 2	Yecla

SALON_HOTEL

hotel	id_salon	...
~~H01~~	~~01~~	
H03	01	
~~H01~~	~~02~~	
H02	01	
~~H01~~	~~03~~	
H03	02	
...		

- Imposible que un salón tenga NULL en hotel (∈ PK!)
- Que un hotel tenga salones no debe impedir que sea eliminado
- Lo correcto: si se borra un hotel, se deben eliminar todos sus salones
- La acción debe ser **cascada: ON DELETE CASCADE**

Mantenimiento de la Integridad Referencial

EMPLEADO ⟶ DEPARTAMENTO

- **Ejemplo 3 ON DELETE**
  - En DEPARTAMENTO se **elimina la fila** del D03

EMPLEADO

NSS	nombre	*dep*	...
123456789012	García, A.	**NULL**	
444444444444	Zapata, D.	D02	
333333333333	Arjona, C.	NULL	
222222222222	Sancho, B.	**NULL**	
556644332255	Gómez, H.	NULL	
998877665544	Bolado, F.	D01	

DEPARTAMENTO

codigo	nombre	...
D01	Personal	
D02	I+D	
~~D03~~	~~Administración~~	

- Si se elimina un departamento, sus empleados pueden quedar sin departamento asignado
- La acción debe ser **establecer nulos: ON DELETE SET NULL**

Mantenimiento de la Integridad Referencial

R2 ⟶ R1

Operación: **Modificar** el valor de **la clave CK de una fila t** de R1 **referenciada por otras** filas de R2

Acciones posibles (AL MODIFICAR, **ON UPDATE**)

1. **Rechazar** la operación (acción por defecto: la no-acción)
   - Sólo se permite modificar la CK de t si ninguna fila referencia a t
2. **Cascada**. Propagar la modificación
   - Toda fila de R2 que referencia a t seguirá haciéndolo

   1° Cambiar su valor de FK por el nuevo valor de la CK de t
   2° Modificar el valor de la CK de t
3. **Establecer nulos** – 👁 Sólo si se ha definido que la FK de R2 permite NULL

   1° Toda fila de R2 que referencia a t pasa a contener NULL en la FK
   2° Modificar el valor de la CK de t
4. **Establecer valor por defecto** 👁 Sólo si existe DEFAULT para la FK

   1° Toda fila de R2 que referencia a t toma su valor por defecto en las columnas FK
   2° Modificar el valor de la CK de t

Mantenimiento de la Integridad Referencial

CUENTA ⟶ CLIENTE

- **Ejemplo 1 ON UPDATE**
  - En CLIENTE se **cambia el código del cliente** 2689 a 3030

CLIENTE

codigo	nombre	direccion	ciudad
1210	García, A.	Gran Vía, 6	Murcia
0300	Zapata, D.	Ronda Norte, 3	Murcia
1003	Arjona, C.	Paseo Rosales, 9	Molina
3030	Sancho, B.	Plaza Mayor, 2	Patiño
3679	Burgos, C.	Camino Viejo, 20	Yecla

CUENTA

numero	saldo	*titular*
200	85.005	2689
505	40.000	**3030**
821	50.000	1210
426	35.620	1003
005	29.872	**3030**
315	3.500	0300

- Se puede modificar de igual modo los valores de la clave ajena para que las filas de CUENTA sigan referenciando a la misma fila de CLIENTE
- La acción debe ser **cascada: ON UPDATE CASCADE**

Esta es la acción más común para ON UPDATE

Mantenimiento de la Integridad Referencial

CUENTA ⟶ CLIENTE

- **Ejemplo 2 ON UPDATE**
  - En CLIENTE se **cambia el código del cliente** 2689 a 3030

CLIENTE

codigo	nombre	direccion	ciudad
1210	García, A.	Gran Vía, 6	Murcia
0300	Zapata, D.	Ronda Norte, 3	Murcia
1003	Arjona, C.	Paseo Rosales, 9	Molina
2689	Sancho, B.	Plaza Mayor, 2	Patiño
3679	Burgos, C.	Camino Viejo, 20	Yecla

~~3030~~ → **2689**

CUENTA

numero	saldo	*titular*
200	85.005	**2689**
505	40.000	1003
821	50.000	1210
426	35.620	1003
005	29.872	**2689**
315	3.500	0300

- Si es un requisito impedir la modificación de la clave primaria de CLIENTE si ya existen filas de CUENTA que lo referencian...
- ... la acción debe ser **rechazar: ON UPDATE NO ACTION**

Mantenimiento de la Integridad Referencial

EMPLEADO ⟶ DEPARTAMENTO

- **Ejemplo 3 ON UPDATE**
  - En DEPARTAMENTO se cambia el codigo del D03 al D3

EMPLEADO

NSS	nombre	*dep*	...
123456789012	García, A.	**NULL**	
444444444444	Zapata, D.	D02	
333333333333	Arjona, C.	NULL	
222222222222	Sancho, B.	**NULL**	
556644332255	Gómez, H.	NULL	
998877665544	Bolado, F.	D01	

DEPARTAMENTO

codigo	nombre	...
D01	Personal	
D02	I+D	
D3	Administración	

Quizá junto con el "codigo" también se modifique "nombre"

- Si se modifica el valor de la PK de un departamento y se entiende que ahora esa fila representa *otro departamento*, entonces tiene sentido que sus antiguos empleados queden *sin departamento* asignado...
- ... la acción referencial debe ser **establecer nulos: ON UPDATE SET NULL**

Mantenimiento de la Integridad Referencial

- Así que hay que completar cada FOREIGN KEY con las cláusulas **ON DELETE** y **ON UPDATE**
- En ANSI SQL, hay 4 posibles acciones referenciales :

***acción*∈{NO ACTION,CASCADE,SET NULL,SET DEFAULT}**

- En nuestro ejemplo…

ANSI SQL

```
CREATE TABLE empleado (
…,
FOREIGN KEY (dep) REFERENCES departamento(coddep)
            ON DELETE NO ACTION
            ON UPDATE CASCADE,
FOREIGN KEY (nssjefe) REFERENCES empleado(nss)
              ON DELETE SET NULL
              ON UPDATE CASCADE,
… );
```

Oracle. Acciones Referenciales

- **ON DELETE**
  - En Oracle SQL, hay 3 posibles acciones en caso de borrado: NO ACTION, CASCADE y SET NULL
  - En la cláusula **ON DELETE *acción*** solo se puede especificar una de estas dos: **CASCADE** o **SET NULL**
  - Y por omisión (si no se indica nada): se asume **NO ACTION**
- **ON UPDATE**
  - Oracle SQL **NO implementa** la cláusula ON UPDATE del ANSI SQL
  - Siempre se asume ON UPDATE **NO ACTION**

Oracle asume NO ACTION para ON DELETE y ON UPDATE

```
CREATE TABLE empleado (
   ...,
   FOREIGN KEY (dep) REFERENCES departamento(coddep),
   FOREIGN KEY (nssjefe) REFERENCES empleado(nss)
                              ON DELETE SET NULL,
   ...);
```

Oracle asume NO ACTION para ON UPDATE

Ejemplo de definición de una tabla


```
CREATE TABLE empleado (
   nombre          VARCHAR2(25)      NOT NULL,
   apellido        VARCHAR2(15)      NOT NULL,
   nss             NUMBER(12)        NOT NULL,
   dni             CHAR(9)           NOT NULL,
   fechanacim      DATE              NULL,
   ciudad          VARCHAR2(30)      ,
   est_civil       CHAR(1)           ,
   salario         NUMBER(6,2)       DEFAULT 1000 NOT NULL,
   dep             CHAR(3)           NULL,
   nssjefe         NUMBER(12)        ,
   cuantos_familiares NUMBER(2) DEFAULT 0 NOT NULL,
   PRIMARY KEY (nss),
   UNIQUE (dni),
   FOREIGN KEY (dep)    REFERENCES departamento(coddep),
            -- ON DELETE NO ACTION ON UPDATE CASCADE
   FOREIGN KEY (nssjefe) REFERENCES empleado(nss),
            -- ON DELETE SET NULL ON UPDATE CASCADE
 CHECK (nssjefe <> nss),
 CHECK (est_civil IN ('S','C','V','D','P')),
 CHECK (salario > 0)
);
```

El código queda más claro si incluimos **comentarios** con las **acciones referenciales que quisiéramos haber indicado** para ON DELETE y ON UPDATE

En las prácticas, ON DELETE y ON UPDATE **deben** estar dentro de comentarios

Nombres de restricción

- Dar nombre a una restricción **es opcional**, pero es MUY conveniente y se recomienda hacerlo
- **Anteponer** a una cláusula PRIMARY KEY, FOREIGN KEY, UNIQUE o CHECK esto:

  CONSTRAINT *nombre_RI*

  - El *nombre_RI* debe ser **único dentro del mismo esquema**
    - Dos tablas distintas del mismo esquema (mismo usuario en Oracle, por ejemplo) no pueden contener restricciones con igual nombre
- **Identifica una restricción**, por si después debe ser eliminada o sustituida por otra

- Y el SGBD lo usa en los **mensajes de error** generados ante intentos de incumplir la restricción

Ejemplo de definición de una tabla


```
CREATE TABLE empleado (
 nombre                    VARCHAR2(25)    NOT NULL,
 apellido                  VARCHAR2(15)    NOT NULL,
 nss                       NUMBER(12)      NOT NULL,
 dni                       CHAR(9)         NOT NULL,
 fechanacim                DATE            NULL,
 ciudad                    VARCHAR2(30)    ,
 est_civil                 CHAR(1)         ,
 salario                   NUMBER(6,2)     DEFAULT 1000 NOT NULL,
 dep                       CHAR(3)         NULL,
 nssjefe                   NUMBER(12)      ,
 cuantos_familiares NUMBER(2) DEFAULT 0 NOT NULL,
 CONSTRAINT emp_pk    PRIMARY KEY (nss),
 CONSTRAINT emp_ak    UNIQUE (dni),
 CONSTRAINT emp_fk_dep FOREIGN KEY (dep)
                  REFERENCES departamento(coddep),
               --ON DELETE NO ACTION ON UPDATE CASCADE
 CONSTRAINT emp_fk_emp FOREIGN KEY (nssjefe) REFERENCES empleado(nss),
               --ON DELETE NO ACTION ON UPDATE CASCADE
 CONSTRAINT emp_jefe_ok CHECK (nssjefe <> nss),
 CONSTRAINT emp_ec_ok   CHECK (est_civil IN ('S','C','V','D','P')),
 CONSTRAINT emp_sal_ok  CHECK (salario > 0)
);
```

Especificación simple de restricciones

- **Especificación "corta"**: Si la restricción afecta sólo a una columna, puede especificarse en la definición de dicha columna
- Por ejemplo, si una clave ajena no es compuesta, no se necesita la cláusula FOREIGN KEY completa, basta con la cláusula REFERENCES…

```
CREATE TABLE empleado (
  nombre      VARCHAR2(15)   NOT NULL,
  nss         NUMBER(12)     PRIMARY KEY,
  dni         CHAR(9)        NOT NULL UNIQUE,
  est_civil   CHAR(1)        CHECK (est_civil IN
                               ('S','C','V','D','P')),
  salario     NUMBER(6,2)    DEFAULT 1000 NOT NULL CHECK(salario>0),
  dep         CHAR(3)        NULL REFERENCES departamento(coddep),
  nssjefe     NUMBER(12)     REFERENCES empleado(nss),
  …, -- resto de columnas
  CHECK (nssjefe <> nss)
  );
```

ⓘ Se recomienda **NO usar la especificación corta**. El CREATE TABLE es más legible si se separa la definición de las columnas de las restricciones de integridad. Además, es conveniente dar un nombre a cada restricción

Esta restricción afecta a 2 columnas, por lo que no se puede especificar en la definición de una sola de esas columnas. Se deja así

Ejemplos de clave ajena compuesta

- Una clave ajena es compuesta cuando lo es la clave (primaria o alternativa) a la que referencia

```
CREATE TABLE hotel(
 codigo CHAR(4)     NOT NULL,
 nombre VARCHAR2(30) NOT NULL,
 ... -- otras columnas
 CONSTRAINT hotel_pk PRIMARY KEY(codigo)
);
CREATE TABLE salon_hotel(
 id_salon  CHAR(2)   NOT NULL,
 hotel     CHAR(4)   NOT NULL,
 capacidad NUMBER(3) NOT NULL,
 ... -- otras columnas
 CONSTRAINT salon_pk PRIMARY KEY (hotel, id_salon),
 CONSTRAINT salon_fk_hotel FOREIGN KEY(hotel)
                  REFERENCES hotel(codigo)
                   ON DELETE CASCADE
 ...);
```

La clave primaria está compuesta por dos columnas

- Cada salón referencia al hotel en el cual está ubicado (clave ajena)

Ejemplos de clave ajena compuesta

- La tabla RESERVA_SALON almacena las distintas reservas de los salones de cada hotel a lo largo del tiempo

```
CREATE TABLE reserva_salon(
 numero   NUMBER(3)          NOT NULL,
 hotel    CHAR(4) NOT NULL,
 salon    CHAR(2) NOT NULL,
 fecha    DATE    NOT NULL,
 ... --otras columnas
 CONSTRAINT reserva_pk   PRIMARY KEY(numero),
 CONSTRAINT res_fk_salon FOREIGN KEY(hotel,salon)
                 REFERENCES salon_hotel(hotel,id_salon)
                   -- ON DELETE NO ACTION
);
```

Una clave ajena debe tener tantas columnas como tiene la clave primaria a la que referencia

- La referencia es hacia la tabla SALON_HOTEL, puesto que cada reserva se refiere a cierto salón en concreto
  - No tiene sentido hacer referencia en solitario a la tabla HOTEL, por ejemplo
  - Como la clave primaria de SALON_HOTEL tiene dos columnas, entonces la clave ajena también debe tener dos

Ejemplos de clave ajena compuesta

HOTEL

codigo	nombre	...
H001	ZENIT	
H002	NH AMISTAD	
H003	HESPERIA	
H004	NELVA	
H005	CAMPANILE	
H006	IBIS	

SALON_HOTEL

hotel	id_salon	capacidad	...
H001	S1	60	
H001	S2	50	
H001	S3	100	
H002	S1	85	
H002	S2	50	
H003	S1	25	
H004	S1	95	
H004	S2	50	
H004	S3	85	

RESERVA_SALON

numero	hotel	salon	fecha	...
2	H001	S2	14/09/2019	
3	H001	S1	14/09/2019	
5	H002	S1	15/10/2019	
1	H002	S2	09/09/2019	
6	H003	S1	19/10/2019	
7	H004	S1	20/10/2019	
4	H001	S1	09/10/2019	
8	H004	S1	18/11/2019	

Un mismo salón puede ser reservado varias veces

Ejemplos de clave ajena compuesta

- Esquema de la biblioteca

```
CREATE TABLE ejemplar (
  ISBN   CHAR(13)   NOT NULL,
  numero NUMBER(2)  NOT NULL,
  estante CHAR(4)   NOT NULL,
  PRIMARY KEY (ISBN, numero),
  FOREIGN KEY (ISBN) REFERENCES libro(ISBN)
);
CREATE TABLE prestamo (
  libro     CHAR(13)  NOT NULL,
  ejemplar  NUMBER(2) NOT NULL,
  socio     CHAR(4)   NOT NULL,
  fecha     DATE      NOT NULL,
  devolucion DATE     NOT NULL,
  PRIMARY KEY (libro, ejemplar, socio, fecha),
  FOREIGN KEY (libro, ejemplar)
    REFERENCES EJEMPLAR(ISBN, numero),
  FOREIGN KEY (socio) REFERENCES socio(codigo)
);
```

Clave primaria compuesta por dos columnas

El tipo de datos de cada columna de una clave ajena debe ser el mismo que el de la correspondiente columna dentro de la clave primaria referenciada

Clave ajena compuesta por 2 columnas, **porque la clave primaria** de EJEMPLAR **está compuesta** por 2 columnas

Alterar la estructura de una tabla

- **Sentencia ALTER TABLE**
- Permite modificar la estructura de una tabla existente, independientemente de que contenga datos (filas)
  - **Añadir** una nueva **columna**
  - **Modificar** una **columna** ya existente
  - **Eliminar** una **columna**
  - **Añadir** una **restricción**
    - Añadir una **CHECK**, una **clave primaria**, una clave **alternativa**
    - Añadir una **clave ajena**
  - **Modificar** una **restricción**
    - **Activar y desactivar** restricciones
  - **Eliminar** una **restricción**

Veremos la sentencia **ALTER TABLE de Oracle SQL:** incluye las cláusulas del ANSI SQL, aunque cambia un poco la sintaxis e incorpora muchas más opciones

Añadir una columna a una tabla

Oracle SQL

- **ALTER TABLE** *tabla*
  ADD (*definición_columna);*

```
ALTER TABLE empleado
 ADD (fecha_contrato DATE);
```

```
ALTER TABLE empleado
 ADD (puesto VARCHAR2(12));
```

- Todas las filas de la tabla tendrán NULL en la nueva columna
- No está permitido incluir NOT NULL en la definición de la nueva columna...
  - Si es necesaria esta restricción, podrá añadirse después
- ... salvo que también se indique un valor por defecto:

```
ALTER TABLE empleado
  ADD (puesto VARCHAR2(12) DEFAULT 'aprendiz' NOT NULL);
```

Renombrar una columna

Oracle SQL

- Renombrar una **columna** de una **tabla**

```
ALTER TABLE empleado
 RENAME COLUMN fecha_contrato TO inicio_contrato;
```

- Renombrar una **tabla**

```
ALTER TABLE empleado RENAME TO emp;
```

- Renombrar un **elemento** (**tabla**, vista, secuencia, sinónimo privado) de la base de datos

```
RENAME empleado TO emp;
```

Modificar definición de una columna

Oracle SQL

- **ALTER TABLE** *tabla* **MODIFY** *columna acción;*
  - Permite modificar…
    - Tipo de datos
    - DEFAULT
    - NULL o NOT NULL
    - UNIQUE

```
ALTER TABLE empleado
 MODIFY apellido VARCHAR2(30);
        --antes tamaño 15

ALTER TABLE departamento
 MODIFY nssdire DEFAULT '123';

ALTER TABLE empleado
 MODIFY salario NOT NULL;

ALTER TABLE departamento
 MODIFY nombre UNIQUE;
```

Eliminar una columna de una tabla

Oracle SQL

- **ALTER TABLE** *tabla* **DROP COLUMN** *columna*;
  - Permite **una sola columna**

```
ALTER TABLE empleado
 DROP COLUMN fechanacim;
```

- **ALTER TABLE** *tabla* **DROP(***lista_columnas***);**
  - Permite eliminar **varias columnas a la vez**

```
ALTER TABLE empleado
 DROP (fechanacim, est_civil);
```

Eliminar una columna de una tabla

Oracle SQL

- **Eliminar una columna referenciada**
  - Existen claves ajenas que le hacen referencia
  - O aparece en la definición de una vista (*se verá*)
  - Sólo se elimina una columna si **no está referenciada** desde ninguna otra tabla o vista

    ALTER TABLE empleado **DROP COLUMN** ciudad;
  - Oracle **no permite eliminar** una **columna referenciada**

    ALTER TABLE departamento **DROP COLUMN coddep**;

    Esta sentencia falla
  - Se puede **forzar** su **eliminación con** la cláusula **CASCADE CONSTRAINTS**
    - Se elimina la columna, así como toda *restricción de integridad referencial* (y toda vista que le haga referencia)

Eliminar una columna de una tabla

Oracle SQL

- Ejemplo de eliminación de una columna referenciada

EMPLEADO

nombre	apellido	nss	...	*dep*
JONÁS	SOLANO	123		D1
RIGOBERTA	CALAVERA	321		D3
EUSEBIO	MULETAS	222		D2
MACARENO	SOSO	111		D1
CASIANA	FABERGÉ	333		D3
FILOMENA	RASCAS	234	34E	D1
GUMERSINDA	MIMOS	543	45F	NULL

DEPARTAMENTO

coddep	nombre	*nssdire*
D2	INVESTIGACION	222
D1	ADMINISTRACION	111
D3	PERSONAL	333
D4	TRAINING	NULL

ALTER TABLE departamento
DROP COLUMN coddep **CASCADE CONSTRAINTS**;

- **Elimina** la columna **coddep** junto con la *restricción de integridad referencial* que vincula EMPLEADO.dep con DEPARTAMENTO
- EMPLEADO.**dep** no es eliminada, pero **deja de ser clave ajena**

Eliminar una columna de una tabla

Oracle SQL

- Ejemplo de eliminación de una columna referenciada

ALTER TABLE departamento
DROP COLUMN coddep **CASCADE CONSTRAINTS**;

La columna "dep" deja de ser clave ajena. Queda como un **atributo normal**, cuyos valores ya no tienen por qué existir en ningún otro lugar

EMPLEADO

nombre	apellido	nss	...	dep
JONÁS	SOLANO	123		D1
RIGOBERTA	CALAVERA	321		D3
EUSEBIO	MULETAS	222		D2
MACARENO	SOSO	111		D1
CASIANA	FABERGÉ	333		D3
FILOMENA	RASCAS	234	34E	D1
GUMERSINDA	MIMOS	543	45F	NULL

Desaparecen las referencias

DEPARTAMENTO

nombre	*nssdire*
INVESTIGACION	222
ADMINISTRACION	111
PERSONAL	333
TRAINING	NULL

Obviamente, sólo hay que ejecutar sentencias de este tipo si tiene sentido hacerlo...

Añadir una restricción

Oracle SQL

ALTER TABLE *tabla* **ADD**
CONSTRAINT *nombre_RI definición_RI*;

- **Añadir una restricción de integridad CHECK**

  * *Añadir una restricción que compruebe que los valores de "cuantos_familiares" son mayores o iguales a cero*

```
ALTER TABLE empleado ADD
 CONSTRAINT emp_fam_ok CHECK (cuantos_familiares >= 0);
```

  * *Añadir una restricción que compruebe que "fechanacim" tiene valores no nulos*

```
ALTER TABLE empleado ADD
 CONSTRAINT emp_fech_ok CHECK (fechanacim IS NOT NULL);
```

  * *Añadir una restricción **desactivada***

```
ALTER TABLE empleado ADD
 CONSTRAINT emp_sal_max CHECK (salario<=5000) DISABLE;
```

Añadir una restricción

Oracle SQL

- **Añadir** una restricción de **clave primaria**

```
ALTER TABLE producto
    ADD CONSTRAINT producto_pk PRIMARY KEY(codigo);
```

  - La tabla **no** debe tener definida ya una clave primaria
  - La(s) columna(s) debe(n) existir en la tabla ("codigo")
  - Si la tabla ya contiene filas y en la columna (o combinación de columnas) hay valores repetidos, la sentencia **falla**

- **Añadir** una restricción de **clave alternativa** (única)

```
ALTER TABLE autor ADD
    CONSTRAINT autor_ak UNIQUE(nombre,apellido1,año_nacim);
```

  - La(s) columna(s) ya debe(n) existir en la tabla
  - Si la tabla ya contiene filas y en la columna (o combinación de columnas) hay valores repetidos, la sentencia **falla**

Añadir una restricción de clave ajena

Oracle SQL

- **Añadir** una restricción de **clave ajena**

```
ALTER TABLE componente
    ADD CONSTRAINT componente_fk_producto
    FOREIGN KEY(producto)
    REFERENCES producto(codigo);
```

  - La(s) columna(s) ya debe(n) existir en la tabla ("producto")
  - El tipo de datos de cada columna componente de la clave ajena debe ser igual al de la columna correspondiente en la clave a la que referencia

- A veces **no es posible crear una tabla con todas sus claves ajenas,** y hay que **añadir** alguna **después** de haber creado la tabla

Veamos un ejemplo...

Añadir una restricción de clave ajena

Oracle SQL

- Ejemplo: ***ciclo referencial*** entre tablas

  EMPLEADO.dep → DEPARTAMENTO(coddep)

  DEPARTAMENTO.nssdire → EMPLEADO(nss)

EMPLEADO

nombre	apellido	nss	...	*dep*
JONÁS	SOLANO	123		D1
RIGOBERTA	CALAVERA	321		D3
EUSEBIO	MULETAS	222		D2
MACARENO	SOSO	111		D1
CASIANA	FABERGÉ	333		D3
FILOMENA	RASCAS	234	34E	D1
GUMERSINDA	MIMOS	543	45F	NULL

DEPARTAMENTO

coddep	nombre	*nssdire*
D2	INVESTIGACION	222
D1	ADMINISTRACION	111
D3	PERSONAL	333
D4	TRAINING	NULL

- Veamos qué pasa al tratar de crear las dos tablas...

Añadir una restricción de clave ajena

Oracle SQL

Ejemplo: ***ciclo referencial*** *entre 2 tablas*

◀ Ejecución del script SQL

```
CREATE TABLE empleado(
 nss...,
 ...,
 dep...,
 ...
CONSTRAINT emp_fk_dep
     FOREIGN KEY dep REFERENCES departamento(coddep),
      -- ON DELETE NO ACTION ON UPDATE CASCADE
);

CREATE TABLE departamento(
 coddep...,
 ...
 nssdire...,
 ...
 CONSTRAINT dep_fk_emp
     FOREIGN KEY nssdire REFERENCES empleado(nss),
      -- ON DELETE NO ACTION ON UPDATE CASCADE
... );
```

ERROR

ERROR

EMPLEADO debe contener una clave ajena hacia DEPARTAMENTO (columna "dep"), **pero no puede incluirse en su CREATE TABLE, porque la tabla DEPARTAMENTO aún no ha sido creada**

La sentencia CREATE **falla**, y **NO se crea la tabla EMPLEADO**

DEPARTAMENTO debe contener una clave ajena hacia EMPLEADO (columna "nssdire")

Si la tabla **EMPLEADO no se ha podido crear**, **tampoco** se puede ejecutar esta sentencia: **no** se crea la tabla **DEPARTAMENTO**

Añadir una restricción de clave ajena

Oracle SQL

***Solución** al problema ejemplo:*

```
CREATE TABLE empleado(
 nss...,
 ...,
 dep...,
 ...);

CREATE TABLE departamento(
 coddep...,
 ...,
 nssdire...,
 ...,
 CONSTRAINT dep_fk_emp
      FOREIGN KEY nssdire REFERENCES empleado(nss),
       -- ON DELETE NO ACTION ON UPDATE CASCADE
 ...);
```

Crear la primera tabla (EMPLEADO) **sin la restricción de clave ajena** que provoca el error.
Incluye el atributo "dep", pero no la especificación de que es clave ajena (cláusula FOREIGN KEY).

Crear la otra tabla (DEPARTAMENTO) con su clave ajena, que ya referencia a una tabla creada anteriormente

Alterar la estructura de la primera tabla, para añadir la restricción de clave ajena

```
ALTER TABLE empleado ADD CONSTRAINT emp_fk_dep
      FOREIGN KEY dep REFERENCES departamento(coddep);
       -- ON DELETE NO ACTION ON UPDATE CASCADE
```

Desactivar/Activar una restricción

Oracle SQL

- **Inhabilitar** (desactivar) una restricción de tabla

  ALTER TABLE *tabla*
  DISABLE CONSTRAINT *nombre_RI*;

  * *Desactivar la RI de clave ajena en EMPLEADO que referencia a DEPARTAMENTO*

  ```
  ALTER TABLE empleado
    DISABLE CONSTRAINT emp_fk_dep;
  ```

  * *Inhabilitar la comprobación de que el jefe de un empleado no puede ser él mismo*

  ```
  ALTER TABLE empleado
    DISABLE CONSTRAINT emp_jefe_ok;
  ```

- **Habilitar** (activar) una restricción de tabla

  ALTER TABLE *tabla*
  ENABLE CONSTRAINT *nombre_RI*;

  * *Activar la RI de clave ajena en EMPLEADO que referencia a DEPARTAMENTO*

  ```
  ALTER TABLE empleado
    ENABLE CONSTRAINT emp_fk_dep;
  ```

Eliminar una restricción

Oracle SQL

- Eliminar la restricción de **clave primaria**

  ALTER TABLE *tabla*
  DROP PRIMARY KEY [CASCADE];

- Eliminar una restricción de **clave alternativa**

  ALTER TABLE *tabla*
  DROP UNIQUE (*lista_columnas***) [CASCADE];**

- Eliminar cualquier **restricción,** indicando su nombre

  ALTER TABLE *tabla*
  DROP CONSTRAINT *nombre_RI* **[CASCADE];**

- **Eliminar** una **PRIMARY KEY o UNIQUE falla si existe** alguna **clave ajena** que le haga referencia
  - Ahí entra en juego la opción CASCADE

Eliminar una restricción

Oracle SQL

- La **opción CASCADE** permite eliminar la PRIMARY KEY o UNIQUE junto con la restricción de clave ajena

  * *Eliminación de la restricción de clave primaria de una tabla*

    ALTER TABLE empleado
    DROP PRIMARY KEY CASCADE;

    Sin CASCADE fallaría, pues nss está referenciado desde DEPARTAMENTO.nssdire

  * *Eliminación de una restricción de clave única (alternativa)*

    ALTER TABLE empleado
    DROP UNIQUE(dni);

    Funciona bien; no necesita CASCADE, pues ninguna clave ajena hace referencia a dni

  - ¿Qué efectos tiene?
    - Se elimina la restricción de clave: las columnas que la componen pasan a ser "normales" y pueden repetir valores
    - Si se incluye CASCADE, también se elimina toda restricción de clave ajena que hacía referencia a la clave eliminada: son columnas "normales" que ya no referencian a nada

IMPORTANTE: No elimina columnas en ninguna tabla

Eliminar una restricción

Oracle SQL

- La cláusula **CONSTRAINT** permite eliminar cualquier restricción si se conoce su **nombre**

  * *Eliminación de una restricción de comprobación CHECK*

    ALTER TABLE empleado
    DROP CONSTRAINT emp_sal_ok;

  * *Eliminación de una restricción de clave primaria*

    ALTER TABLE empleado
    DROP CONSTRAINT emp_pk CASCADE;

  - Esta sentencia es **equivalente a la de DROP PRIMARY KEY**

  * *Eliminación de una restricción de clave única (alternativa)*

    ALTER TABLE empleado
    DROP CONSTRAINT emp_ak;

  - Esta sentencia es **equivalente a la de DROP UNIQUE(dni)**

Eliminar una tabla

Oracle SQL

- **Sentencia DROP TABLE**

  DROP TABLE table
  [CASCADE CONSTRAINTS] [PURGE];

  - Destrucción de una tabla (estructura y contenido)
    - Elimina sus filas (liberando el espacio reservado)
    - Y su definición en el Diccionario de Datos (metadatos)
  - Opciones:
    - Si omite CASCADE CONSTRAINTS, sólo destruye la tabla si no se le hace referencia desde ninguna otra tabla (clave ajena), ni es tabla base de ninguna vista
    - Con CASCADE CONSTRAINTS, además de eliminar la tabla, se destruye toda restricción de integridad referencial que haga referencia a claves primarias o únicas de la tabla eliminada
    - PURGE implica eliminar la tabla y liberar el espacio ocupado en un único paso: Oracle NO sitúa la tabla ni sus objetos dependientes en el *recycle bin*

Esquema Lógico Específico

Ejemplo de la Biblioteca

Esquema Lógico Estándar de partida

EDITORIAL(nombre, calle, numero, cod_post, ciudad)
Admiten nulos: codpost
Clave primaria: nombre

LIBRO(ISBN, titulo, año, edicion, num_copias, *editorial*)
Admiten nulos: Ninguno
Clave primaria: ISBN
Clave ajena: editorial **Referencia_a** EDITORIAL(nombre)
Derivado: num_copias = contar tuplas e de EJEMPLAR
tales que e.ISBN = ISBN

EJEMPLAR(numero, estante, *ISBN*)
Admiten nulos: Ninguno
Clave primaria: (ISBN, numero)
Clave ajena: ISBN **Referencia_a** LIBRO(ISBN)

Esquema Lógico Estándar de partida

SOCIO(codigo, nombre, telefono, DNI, penalizado, fin_pena)
Admiten nulos: DNI, fin_pena
Clave primaria: codigo
Clave alternativa: DNI
Comprobar: penalizado IN ('SI', 'NO')

PRESTAMO(*libro, ejemplar*, *socio*, fecha, maxdevolucion, f_devolucion)
Admiten nulos: f_devolucion
Clave primaria: (libro, ejemplar, socio, fecha)
Clave ajena: (libro, ejemplar) **Referencia_a** EJEMPLAR(ISBN, numero)
Clave ajena: socio **Referencia_a** SOCIO(codigo)
Atributo derivado: maxdevolucion = fecha + 15

Esquema Lógico Estándar de partida

AUTOR(id, nombre, apellido1, apellido2, año_nacim, pais, num_premios)
Admiten nulos: apellido2
Clave primaria: id
Comprobar: (num_premios>0 AND num_premios<99)
/* También así: num_premios BETWEEN 0 AND 99 */

ESCRITO_POR(libro, autor)
Admiten nulos: Ninguno
Clave primaria: (libro, autor)
Clave ajena: libro **Referencia_a** LIBRO(ISBN)
Clave ajena: autor **Referencia_a** AUTOR(id)

Esquema Lógico Estándar de partida

Restricciones surgidas de la Traducción

ASERTO RI_editorial_publica_libros
COMPROBAR_QUE (
NO_EXISTE (una tupla en EDITORIAL
donde el valor de "nombre" NO_ESTÉ_ENTRE
(valores de "editorial" en LIBRO))**);**

ASERTO RI_libro_tiene_copias
COMPROBAR_QUE (
NO_EXISTE (una tupla en LIBRO
donde el valor de "ISBN" NO_ESTÉ_ENTRE
(valores de "ISBN" en EJEMPLAR))**);**

ASERTO RI_autor_escribe_libros
COMPROBAR_QUE (
NO_EXISTE (una tupla de AUTOR
donde el valor de "id" NO_ESTÉ_ENTRE
(valores de "autor" en ESCRITO_POR))**);**

Esquema Lógico Estándar de partida

Restricciones detectadas en el Diseño Conceptual (1/2)

ASERTO RI_socio_penalizado COMPROBAR_QUE
(NO EXISTE (una tupla en SOCIO
donde "penalizado"='SI'
y el valor de "codigo" ESTÉ_ENTRE
(valores de "socio" de tuplas en PRESTAMO
donde "f_devolucion" ES_NULL
y "maxdevolucion">=fecha_actual)))**;**

Un socio penalizado no puede tener préstamos activos

ASERTO RI_socio_maximo_prestamos COMPROBAR_QUE
(4 >= MÁXIMO(CUENTA PARA_TODA_TUPLA_DE SOCIO S
(tuplas en PRESTAMO
donde "socio" = S.codigo
y "f_devolucion" ES_NULL
y "maxdevolucion">=fecha_actual)))**;**

Un socio puede tener activos un máximo de 4 préstamos

Esquema Lógico Estándar de partida

Restricciones detectadas en el Diseño Conceptual (2/2)

ASERTO RI_escrito_por_maximo_autores COMPROBAR_QUE
(6 >= **MÁXIMO**(**CUENTA PARA_TODA_TUPLA_DE** LIBRO **L**
(tuplas en ESCRITO_POR
donde "libro" = **L.**ISBN))**)**;

Un libro puede estar escrito por un máximo de 6 autores

ASERTO RI_año_libro_año_autores COMPROBAR_QUE
(NO EXISTE (una tupla en LIBRO L
donde "año" < (valor de "año" de las tuplas en AUTOR
donde "id" ESTÉ_ENTRE (valores de "autor"
de tuplas en ESCRITO_POR
donde "libro" = L.ISBN))))**)**;

El año de publicación de un libro no puede ser anterior al de nacimiento de alguno de sus autores

Deducir las Acciones Referenciales

- Prestemos atención a...
  Cómo deducir las Acciones Referenciales

- Para **cada una** de las claves ajenas hemos de preguntarnos…
  - ¿Cómo debe comportarse el SGBD ante un intento de ***borrado de una fila referenciada*** por otras?
  - ¿Y ante un intento de ***cambio del valor de la clave*** (*primaria/unique*) de una *fila referenciada* por otras?

Integridad Referencial

LIBRO(ISBN, titulo, año, edicion, num_copias, *editorial*)
Clave primaria: ISBN
Clave ajena: editorial **Referencia_a** EDITORIAL(nombre)

- No se debe permitir eliminar una fila en EDITORIAL si existen filas en LIBRO que le hacen referencia
  - ON DELETE NO ACTION
- Si se cambia el nombre de una EDITORIAL, dicho valor también debería cambiar en las filas de LIBRO que lo contengan en el atributo 'editorial'
  - ON UPDATE CASCADE

LIBRO(ISBN, titulo, año, edicion, num_copias, *editorial*)
Clave primaria: ISBN
Clave ajena: editorial **Referencia_a** EDITORIAL(nombre)
ON DELETE NO ACTION **ON UPDATE** CASCADE

Integridad Referencial

EJEMPLAR(numero, estante, *ISBN*)
Clave primaria: (ISBN, numero)
Clave ajena: ISBN **Referencia_a** LIBRO(ISBN)

- Si se elimina una fila en LIBRO, con ella debe borrarse toda fila de EJEMPLAR que le referencia (proviene de un tipo de **Entidad Débil**)
  - ON DELETE CASCADE
- Si se cambia el ISBN de un LIBRO, dicho valor también debería cambiar en las filas de EJEMPLAR que lo contengan en el atributo "ISBN"
  - ON UPDATE CASCADE

EJEMPLAR(numero, estante, *ISBN*)
Clave primaria: (ISBN, numero)
Clave ajena: ISBN **Referencia_a** LIBRO(ISBN)
ON DELETE CASCADE **ON UPDATE** CASCADE

Integridad Referencial

PRESTAMO(*libro, ejemplar*, *socio*, fecha, maxdevolucion, f_devolucion)
Clave ajena: (libro, ejemplar) **Referencia_a** EJEMPLAR(ISBN, numero)

- Si se intenta eliminar un EJEMPLAR, pero existen préstamos en los que participa, no se debe permitir el borrado
  - ON DELETE NO ACTION
- Si en una fila de EJEMPLAR se cambia el valor de ISBN y/o de número (o en una fila de SOCIO se modifica el codigo), dicho cambio debe propagarse a los atributos libro y/o ejemplar de las filas de PRESTAMO que le hacen referencia
  - ON UPDATE CASCADE

PRESTAMO(*libro, ejemplar*, *socio*, fecha, maxdevolucion, f_devolucion)
Clave ajena: (libro, ejemplar) **Referencia_a** EJEMPLAR(ISBN, numero)
ON DELETE NO ACTION **ON UPDATE** CASCADE

Integridad Referencial

PRESTAMO(*libro, ejemplar*, *socio*, fecha, maxdevolucion, f_devolucion)
Clave ajena: socio **Referencia_a** SOCIO(codigo)

- Si se intenta eliminar un SOCIO, pero existen préstamos en los que participa, no se debe permitir el borrado
  - ON DELETE NO ACTION
- Si en una fila de SOCIO se modifica el codigo, dicho cambio debe propagarse al atributo socio de las filas de PRESTAMO que le hacen referencia
  - ON UPDATE CASCADE

PRESTAMO(*libro, ejemplar*, *socio*, fecha, maxdevolucion, f_devolucion)
Clave ajena: socio **Referencia_a** SOCIO(codigo)
ON DELETE NO ACTION **ON UPDATE** CASCADE

Integridad Referencial

ESCRITO_POR(*libro*, *autor*)
Clave ajena: libro **Referencia_a** LIBRO(ISBN)

- Si se elimina una fila de LIBRO, ya no se necesita mantener registrado quién escribió ese libro
  - ON DELETE CASCADE
  - Importante: se eliminarán filas en ESCRITO_POR… *¡no en* AUTOR*!*
- Si se modifica un ISBN de un LIBRO, dicho cambio debe propagarse al atributo libro de las filas de ESCRITO_POR que le hacen referencia
  - ON UPDATE CASCADE

ESCRITO_POR(*libro*, *autor*)
Clave ajena: libro **Referencia_a** LIBRO(ISBN)
ON DELETE CASCADE **ON UPDATE** CASCADE

Integridad Referencial

ESCRITO_POR(*libro*, *autor*)
Clave ajena: autor **Referencia_a** AUTOR(id)

- Si se elimina una fila de AUTOR, pero existen libros escritos por él, no debe permitirse su borrado
  - ON DELETE NO ACTION
- Si se modifica un id de un AUTOR, dicho cambio debe propagarse al atributo autor de las filas de ESCRITO_POR que le hacen referencia
  - ON UPDATE CASCADE

ESCRITO_POR(*libro*, *autor*)
Clave ajena: autor **Referencia_a** AUTOR(id)
ON DELETE NO ACTION **ON UPDATE** CASCADE

Esquema Lógico **Específico** (**Oracle**)


```
CREATE TABLE editorial(
 nombre    VARCHAR2(15) NOT NULL,
 calle     VARCHAR2(20) NOT NULL,
 numero    NUMBER(2)    NOT NULL,
 cod_post  CHAR(6)      NULL,
 ciudad    VARCHAR2(15) NOT NULL,
 CONSTRAINT editorial_pk PRIMARY KEY(nombre)
);
```

Esquema Lógico Específico (Oracle)


```
CREATE TABLE libro(
 ISBN       CHAR(13)      NOT NULL,
 titulo      VARCHAR2(64) NOT NULL,
 año        NUMBER(4)     NOT NULL,
 edicion     NUMBER(2)     NOT NULL,
 num_copias NUMBER(3)     NOT NULL,
 editorial  VARCHAR2(15) NOT NULL,
 CONSTRAINT libro_pk PRIMARY KEY (ISBN),
 CONSTRAINT libro_fk_editorial
   FOREIGN KEY (editorial)
   REFERENCES editorial(nombre)
   -- ON DELETE NO ACTION
   -- ON UPDATE CASCADE
);
```

Esquema Lógico Específico (Oracle)


```
CREATE TABLE ejemplar(
 numero  NUMBER(2) NOT NULL,
 estante CHAR(4)   NOT NULL,
 ISBN    CHAR(13)  NOT NULL,
 CONSTRAINT ejemplar_pk PRIMARY KEY(libro, numero),
 CONSTRAINT ejemplar_fk_libro
   FOREIGN KEY (libro) REFERENCES libro(ISBN)
   -- ON DELETE CASCADE
   -- ON UPDATE CASCADE
);
```

Esquema Lógico Específico (Oracle)


```
CREATE TABLE socio(
 codigo      CHAR(4)        NOT NULL,
 nombre      VARCHAR2(30)   NOT NULL,
 telefono    NUMBER(9)      NOT NULL,
 DNI         CHAR(11)       NULL,
 penalizado  CHAR(2)        NOT NULL,
 fin_pena    DATE           NULL,
 CONSTRAINT socio_pk PRIMARY KEY(codigo),
 CONSTRAINT socio_ak UNIQUE(DNI),
 CONSTRAINT socio_penalizado_ok
     CHECK (penalizado IN ('SI', 'NO'))
);
```

Esquema Lógico Específico (Oracle)


```
CREATE TABLE prestamo(
 libro          CHAR(13)   NOT NULL,
 ejemplar       NUMBER(2)  NOT NULL,
 socio          CHAR(4)    NOT NULL,
 fecha          DATE       NOT NULL,
 maxdevolucion  DATE       NOT NULL,
 f_devolucion   DATE       NOT NULL,
 CONSTRAINT prestamo_pk
   PRIMARY KEY(libro, ejemplar, socio, fecha),
 CONSTRAINT prestamo_fk_ejemplar
   FOREIGN KEY(libro, ejemplar)
   REFERENCES ejemplar(ISBN, numero),
   -- ON DELETE NO ACTION ON UPDATE CASCADE
 CONSTRAINT prestamo_fk_socio FOREIGN KEY(socio)
   REFERENCES socio(codigo)
   -- ON DELETE NO ACTION ON UPDATE CASCADE
);
```

Esquema Lógico Específico (Oracle)


```
CREATE TABLE autor(
 id           CHAR(2)       NOT NULL,
 nombre       VARCHAR2(20)  NOT NULL,
 apellido1    VARCHAR2(15)  NOT NULL,
 apellido2    VARCHAR2(15)  NULL,
 año_nacim    NUMBER(4)     NOT NULL,
 pais         VARCHAR2(30)  NOT NULL,
 num_premios NUMBER(2)     DEFAULT 0 NOT NULL,
 CONSTRAINT autor_pk PRIMARY KEY(id),
 CONSTRAINT autor_num_premios_ok
     CHECK (num_premios>=0 AND num_premios<99)
);
```

Esquema Lógico Específico (Oracle)


```
CREATE TABLE escrito_por(
 libro CHAR(13) NOT NULL,
 autor CHAR(2)  NOT NULL,
 CONSTRAINT escrito_por_pk PRIMARY KEY(libro, autor),
 CONSTRAINT escrito_por_fk_libro
   FOREIGN KEY (libro) REFERENCES libro(ISBN),
   -- ON DELETE CASCADE
   -- ON UPDATE CASCADE
 CONSTRAINT escrito_por_fk_autor
   FOREIGN KEY(autor) REFERENCES autor(id)
   -- ON DELETE NO ACTION
   -- ON UPDATE CASCADE
);
```

Esquema Lógico Específico (Oracle)


```
/* Derivado: LIBRO.num_copias
Esta sentencia deberá ejecutarse una vez
introducidos los datos en las tablas:
UPDATE libro L
SET num_copias = (SELECT COUNT(*)
                  FROM ejemplar E
                  WHERE E.ISBN = L.ISBN);
*/

/* Valor por defecto calculado:
PRESTAMO.maxdevolucion = fecha + 15
En las instrucciones INSERT INTO prestamo ha de
darse el valor SYSDATE a fecha y el valor SYSDATE+15
a maxdevolucion */
```

Sabremos redactar sentencias como esta cuando dominemos las sentencias SELECT y UPDATE de SQL

Esquema Lógico Específico (Oracle)


```
/* Valor por defecto calculado: SOCIO.fin_pena
Una de estas sentencias se debe ejecutar cuando se
debe penalizar a un socio X, que ha devuelto tarde un
ejemplar libro, es decir, el día que lo devuelve:

UPDATE socio
  SET fin_pena = SYSDATE + 10
WHERE codigo = X;

UPDATE socio S
  SET fin_pena =(SELECT f_devolucion+10
                 FROM prestamo P
                 WHERE f_devolucion = SYSDATE
                   AND P.socio = S.codigo)
WHERE S.codigo = X;
*/
```

Sabremos redactar estas sentencias cuando dominemos las sentencias SELECT y UPDATE de SQL