



FACULTAD DE
CIENCIAS EXACTAS
UNICEN

Tecnicatura Universitaria en Desarrollo de Aplicaciones Informáticas (TUDAI)

Base de Datos

Tema 3: Desarrollo de una Base de Datos

Creación y Actualización de Esquemas

2
0
2
5

Ciclo de vida de una BD

- La creación de una BD, generalmente es una operación difícil, larga y costosa, que no puede improvisarse
- Las decisiones relativas a todo el proceso de creación de una BD no involucran sólo a los informáticos sino a varios sectores de la empresa
- Es necesario que las organizaciones tengan un *plan de trabajo* detallado para la **concepción** de la BD
- Comienza la etapa de **Diseño lógico y físico** de la BD para continuar luego con la **Carga y optimización** de la misma

Metodología

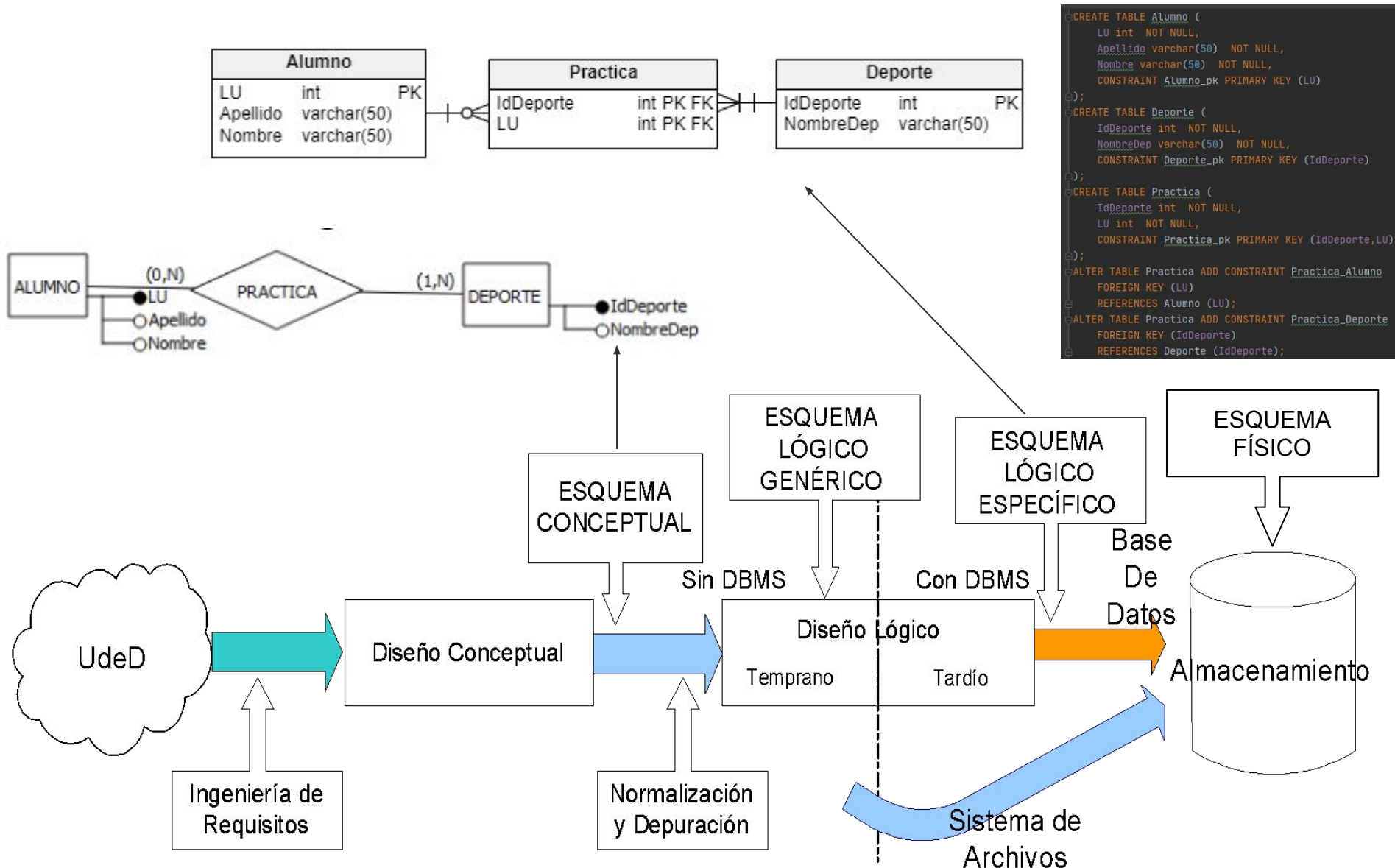
Vamos a seguir una **Metodología para el Desarrollo de una BD Relacional**

Transformaremos nuestro DEREExt en 2 pasos

- Primer paso: transformación a un **modelo lógico relacional**
- Segundo paso: transformación del modelo lógico relacional al **modelo físico de la BD** (tablas, restricciones, etc.)

Para ésta última transformación utilizamos un lenguaje específico de los DBMS, denominado **SQL**

Etapas en el Diseño de Datos



Esquemas de Base de Datos


- Una base de datos relacional consiste en un **conjunto de tablas**, a cada una de las cuales se le asigna un nombre exclusivo dentro del **esquema de cada usuario**
- El conjunto de objetos (tablas, vistas, procedimientos) conforman el esquema de un usuario
- Cada **fila** de la tabla representa una relación entre un conjunto de valores. Dado que cada tabla es un conjunto de dichas relaciones, hay una fuerte correspondencia entre ***el concepto de tabla y el concepto matemático de relación***, del que toma su nombre el modelo de datos relacional

Esquemas de Base de Datos

- Cada tabla posee un conjunto de columnas cabeceras (**atributos**) cuyo nombre debe ser único dentro de la tabla
- Para cada atributo hay un conjunto de valores permitidos, llamado **dominio** de ese atributo.
- Las columnas **pueden ser de distintos tipos**:
numéricos (edad, cantidad de hijos), alfanuméricos (nombre, dirección), fechas (fecha de nacimiento, fecha de ingreso a la compañía), booleano (posee auto propio, cumplió el servicio militar)
- <https://www.postgresql.org/docs/current/datatype.html>

Esquema de Base de Datos

- Todos los datos registrados **en una columna deben ser del mismo tipo**



Recordar todos los nombres
DEBEN ser únicos dentro de
un mismo esquema (no son
nada más que las tablas!!!!)

Ejemplo

MÉDICO

Nombre de la tabla (relación)

Nombre de la columna (atributo)

Esquema de una tabla o cabecera - (comprensión)

Fila (tupla)

Valor o estado de de la tabla (extensión)

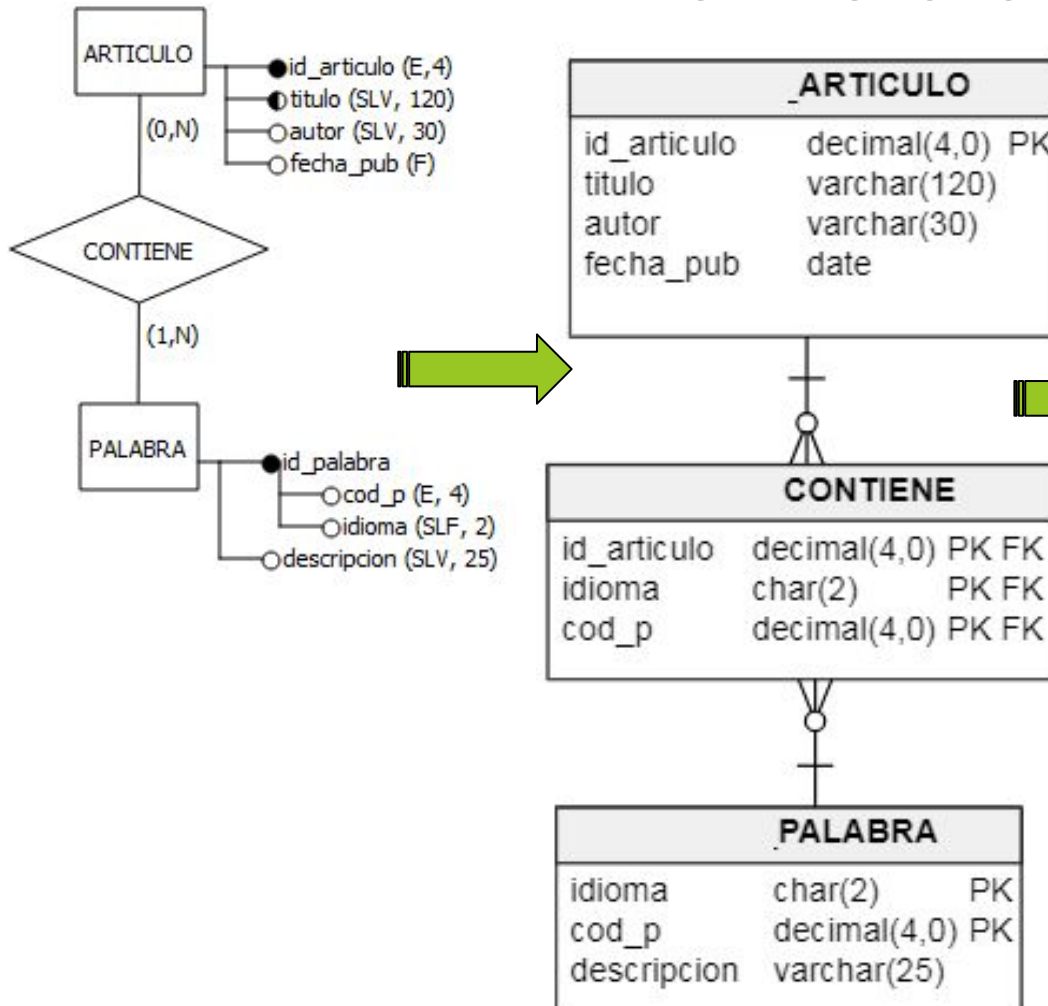
Matricula	NyApell	Especialidad	DNI	ClinicaEjerce
234555	Juan Paz	Traumatología	26456678	C. Modelo
345234	Inés Roca	Pediatría	30564865	C. Paz
365478	Pedro Jara	Traumatología	23546987	Cons. Privado
....

- Las relaciones pueden visualizarse en forma tabular
- Cada columna tiene un dominio de definición que incluye los valores posibles que puede tomar

Tablas en SQL

- En SQL no existe un orden para las filas de una tabla. Cuando se lee una tabla, las filas aparecerán en un orden aleatorio, a menos que se especifique uno
- Las columnas contienen la información de los campos de la tabla: nombre, tipo de dato y restricciones asociadas a la columna
- Las filas contiene los registros o instancias

Cómo transformamos el DERExt a Tablas?



PostgreSQL

```
1 -- Created by Vertabelo (http://vertabelo.com)
2 -- Last modification date: 2018-08-09 17:00:00
3
4 -- tables
5 -- Table: TP1_E1_ARTICULO
6 CREATE TABLE TP1_E1_ARTICULO (
7     id_articulo decimal(4,0) NOT NULL,
8     titulo varchar(120) NOT NULL,
9     autor varchar(30) NOT NULL,
10    fecha_pub date NOT NULL,
11    CONSTRAINT PK_TP1_E1_ARTICULO
12    PRIMARY KEY (id_articulo)
13 );
14
15 -- Table: TP1_E1_CONTIENE
16 CREATE TABLE TP1_E1_CONTIENE (
17     id_articulo decimal(4,0) NOT NULL,
18     idioma char(2) NOT NULL,
19     cod_p decimal(4,0) NOT NULL,
20    CONSTRAINT PK_TP1_E1_CONTIENE
21    PRIMARY KEY (id_articulo, idioma, cod_p)
22 );
```

Reglas de Transformación de Entidades

Las reglas de transformación del DERExt al Esquema Relacional de Bases de Datos para entidades son las siguientes:

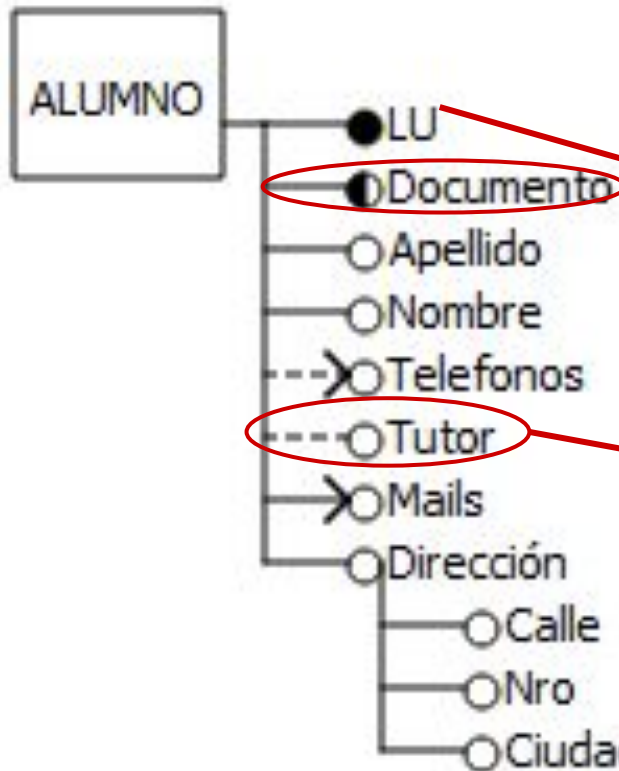
- Se crea **una tabla por cada entidad**, con el mismo nombre de la entidad
- El **identificador de la entidad** se transforma en la **clave primaria** de dicha tabla
- Todo **atributo simplemente valuado** de la entidad se transforma en un atributo de dicha tabla
- Los **atributos obligatorios** llevan una leyenda de NOT NULL

Reglas de Transformación de Entidades (Cont)

- Los **atributos compuestos** se despliegan en sus partes componentes, como si fueran univaluados.
- Los **atributos obligatorios** se indican con la leyenda NOT NULL (En Vertabelo la N significa NULL)
- Los **atributos multivaluados** se proyectan en otra tabla conjuntamente con la clave de la entidad o de la (inter)relación.

Derivación de Entidades

Aplicando las reglas anteriores...



▼ Alternate (unique) keys + Add key

Name
ALUMNO_ak_1

Comment:

Columns

choose column... + Add

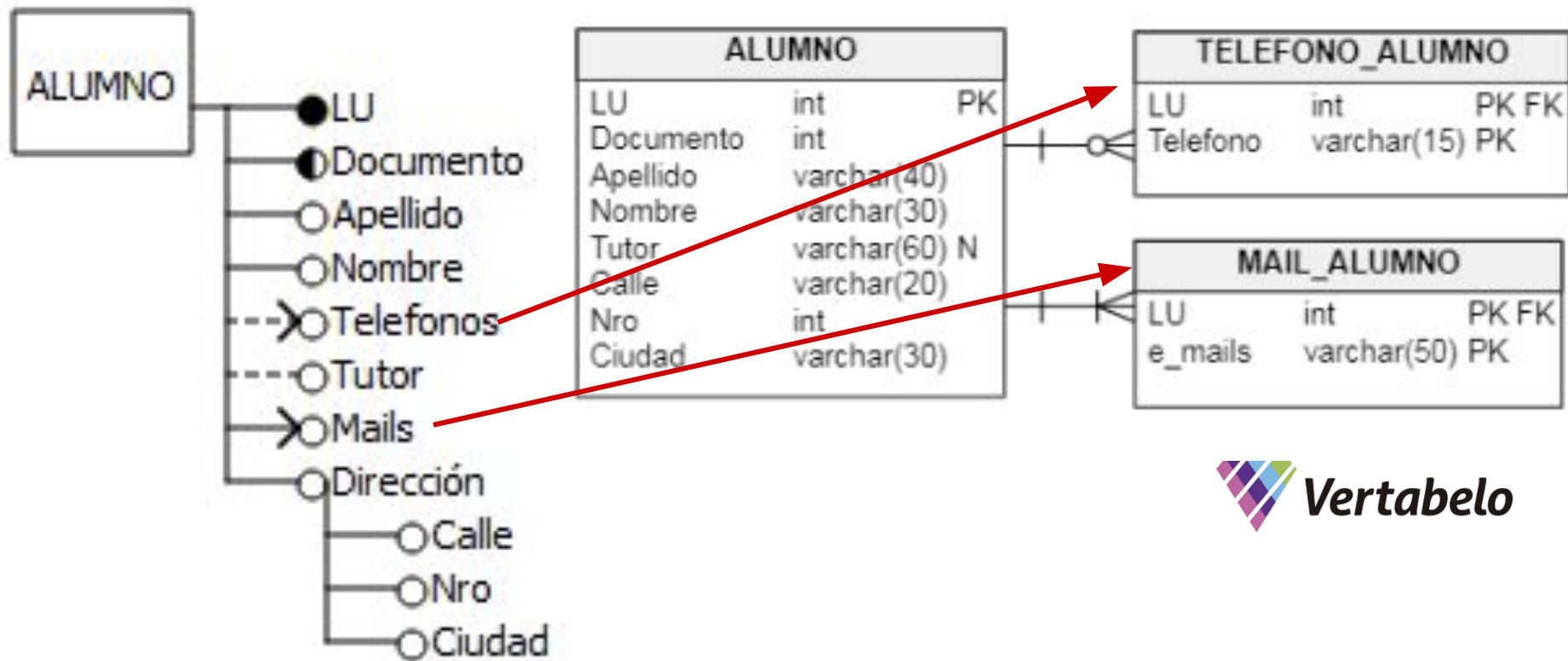
Documento x

ALUMNO		
LU	int	PK
Documento	int	
Apellido	varchar(40)	
Nombre	varchar(30)	
Tutor	varchar(60)	N
Calle	varchar(20)	
Nro	int	
Ciudad	varchar(30)	

BD_03_1_Vertabelo

Derivación de Entidades

Aplicando las reglas anteriores...



Creación de Tablas

- Cada columna debe tener un determinado tipo de dato.
- El tipo de dato limita el conjunto de valores posibles que se pueden asignar a una columna

ALUMNO		
LU	int	PK
Document	int	
Apellido	varchar(40)	
Nombre	varchar(30)	
Tutor	varchar(60)	N
Calle	varchar(20)	
Nro	int	
Ciudad	varchar(30)	

```
CREATE TABLE ALUMNO(  
    LU            integer      NOT NULL,  
    Documento    integer      NOT NULL,  
    Apellido     varchar(40)   NOT NULL,  
    Nombre       varchar(30)   NOT NULL,  
    Tutor        varchar(60),  
    Calle        varchar(20)   NOT NULL,  
    Nro          integer      NOT NULL,  
    Ciudad       varchar(30)   NOT NULL,  
    CONSTRAINT PK_ALUMNO PRIMARY  
    KEY (LU));
```

O puede colocarse la definición de la clave primaria en sentencia aparte

```
ALTER TABLE ALUMNO  
ADD CONSTRAINT PK_ALUMNO PRIMARY  
KEY (LU);
```

Tipos de Datos

Definición de Datos en PostgreSQL

<https://www.postgresql.org/docs/17/ddl.html>

Tipos de datos Postgresql

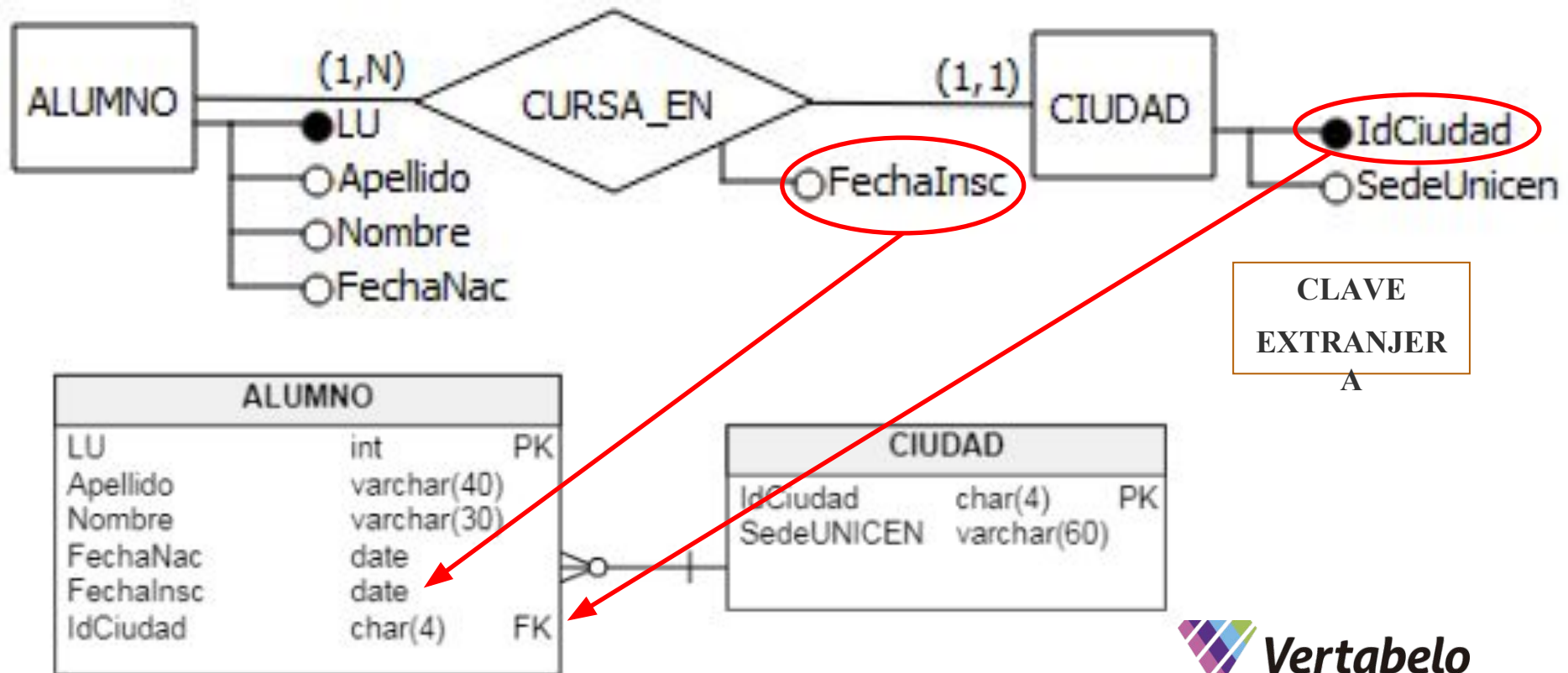
<https://www.postgresql.org/docs/17/datatype.html>

Tipos de datos compuestos

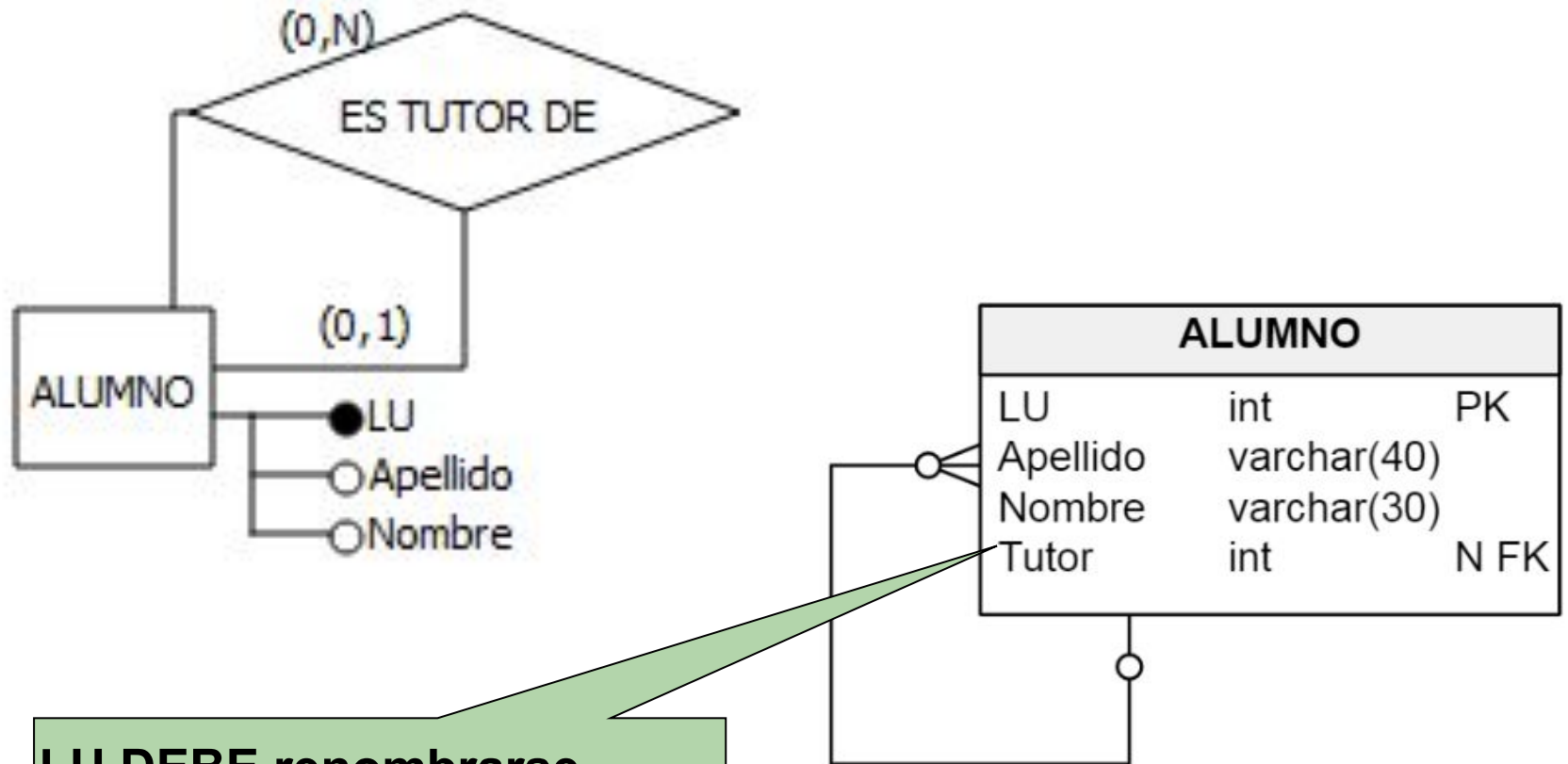
<https://www.postgresql.org/docs/17/rowtypes.html>

Derivación de Relaciones Binarias 1:N

Los atributos identificadores de la entidad (clave de la relación) del 'lado 1', se agregan como atributos en la tabla correspondiente a la entidad del 'lado N' → constituyen una **clave extranjera**



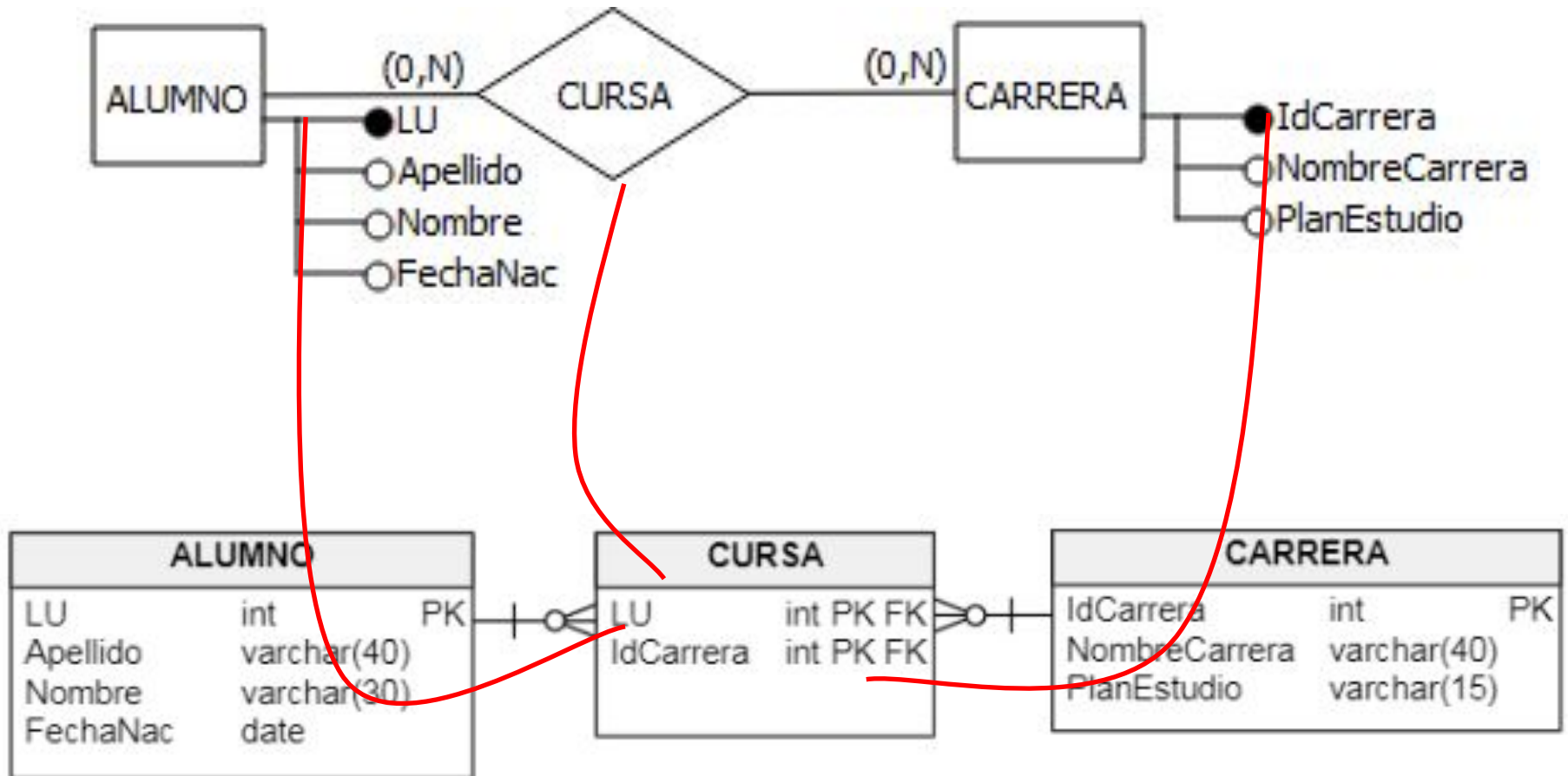
Derivación de Relaciones Unarias 1:N (o N:1)



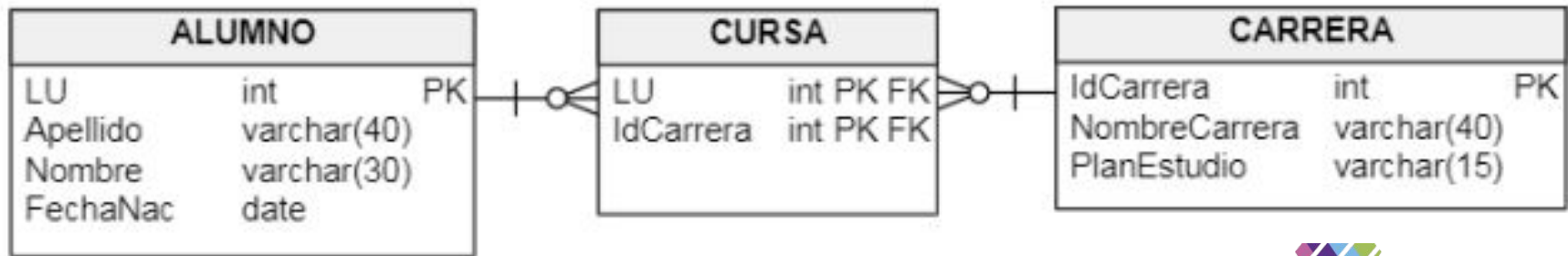
Derivación de Relaciones Unarias y Binarias N:N

- Se crea una **nueva tabla**, cuya clave es la yuxtaposición de los identificadores (claves) de cada una de las entidades participantes.
- Nombre de tabla: nombre indicado en el rombo, o puede renombrarse.
- Cada una de las claves, por separado es una clave extranjera referida a la tabla(entidad) de la cual proviene.

Derivación de Relaciones Binarias N:N



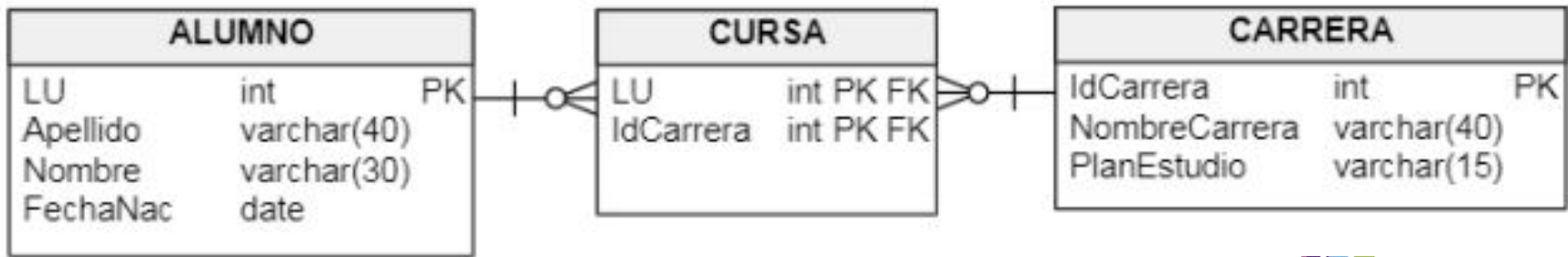
Creación de Tablas



```
CREATE TABLE ALUMNO(  
    LU            integer      NOT NULL,  
    Apellido     varchar(30)  NOT NULL,  
    Nombre       varchar(30)  NOT NULL,  
    FechaNac     date,  
    CONSTRAINT PK_ALUMNO PRIMARY KEY (LU));
```

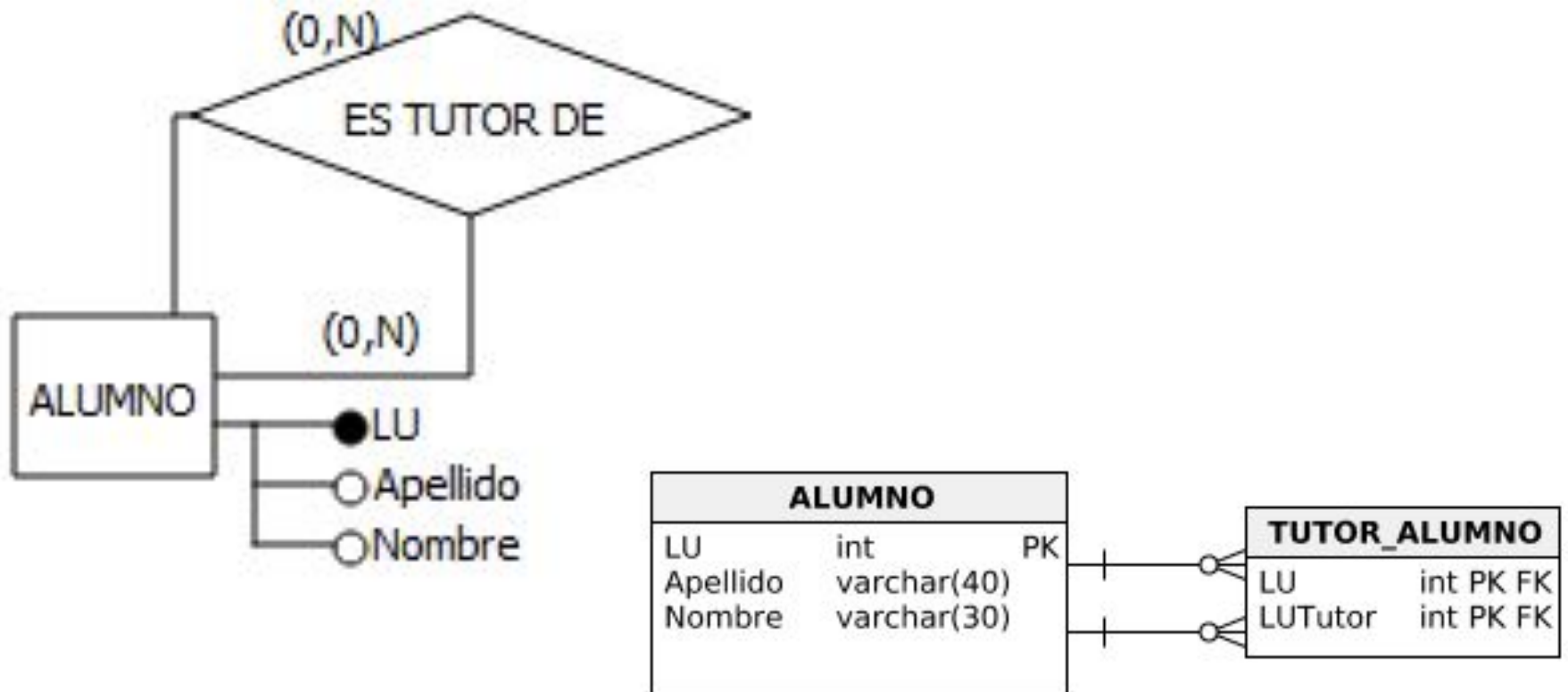
```
CREATE TABLE CARRERA(  
    IdCarrera     integer      NOT NULL,  
    NombreCarrera varchar(40)  NOT NULL,  
    PlanEstudio   varchar(15)  NOT NULL,  
    CONSTRAINT PK_CARRERA PRIMARY KEY (IdCarrera));
```

Creación de Tablas

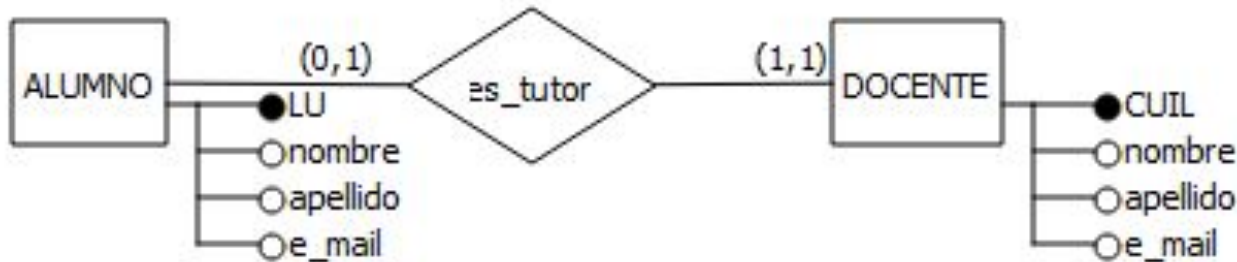


```
CREATE TABLE CURSA(  
    LU          integer NOT NULL,  
    IdCarrera  integer NOT NULL,  
    CONSTRAINT PK_CURSA PRIMARY KEY (LU, IdCarrera));  
  
ALTER TABLE CURSA ADD CONSTRAINT FK_CURSA_CARRERA  
    FOREIGN KEY (IdCarrera)  
    REFERENCES CARRERA(IdCarrera);  
  
ALTER TABLE CURSA ADD CONSTRAINT FK_CURSA_ALUMNO  
    FOREIGN KEY (LU)  
    REFERENCES ALUMNO(LU);
```

Derivación de Relaciones Unarias N:N

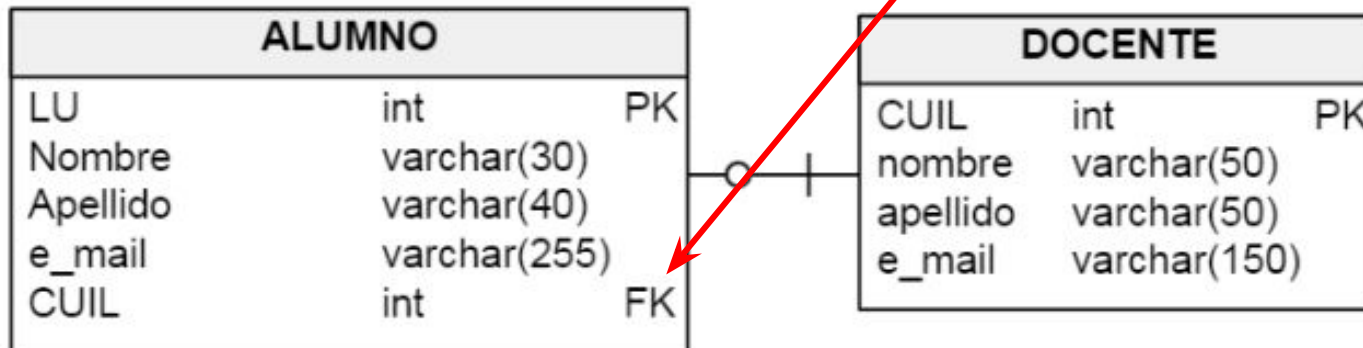


Derivación de Relaciones Binarias 1:1



No olvidarse de crear la ALTERNATIVE KEY

Screenshot of a database management tool showing the 'Alternate (unique) keys' dialog. The 'Name' field contains 'AK1_ALUMNO'. The 'Comment' field is empty. The 'Columns' section shows a dropdown menu with 'choose column...' and a '+ Add' button. Below the dropdown, the column 'CUIL' is listed with a close button 'x'.



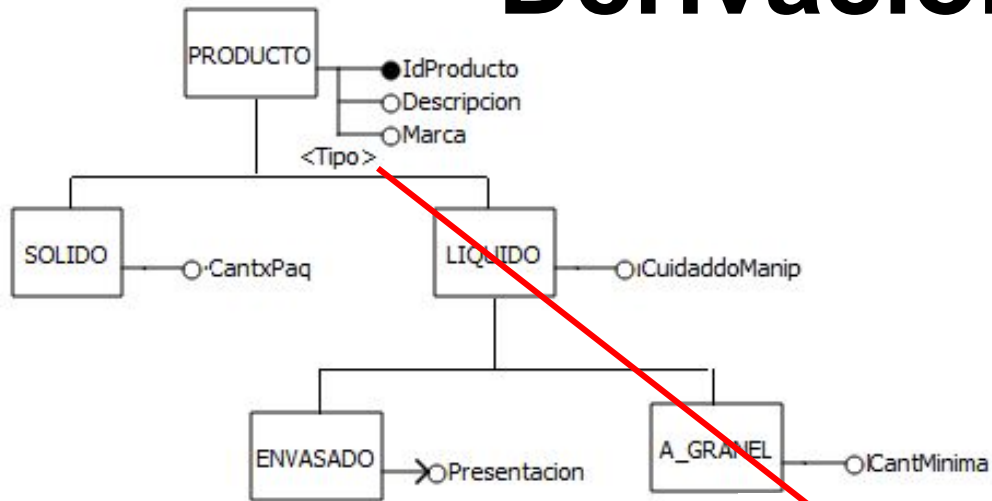
Derivación de Atributos en Relaciones

- Los atributos de una relación pueden ser del mismo tipo que los de una entidad.
- Si la relación que describen es designativa (1:N) entonces se incluyen en la tabla del lado N, derivándolos en forma análoga a los de las entidades.
- Si la relación es asociativa (binaria N:N o ternaria), se derivan en la tabla producto de la relación, también de forma análoga a los de las entidades.

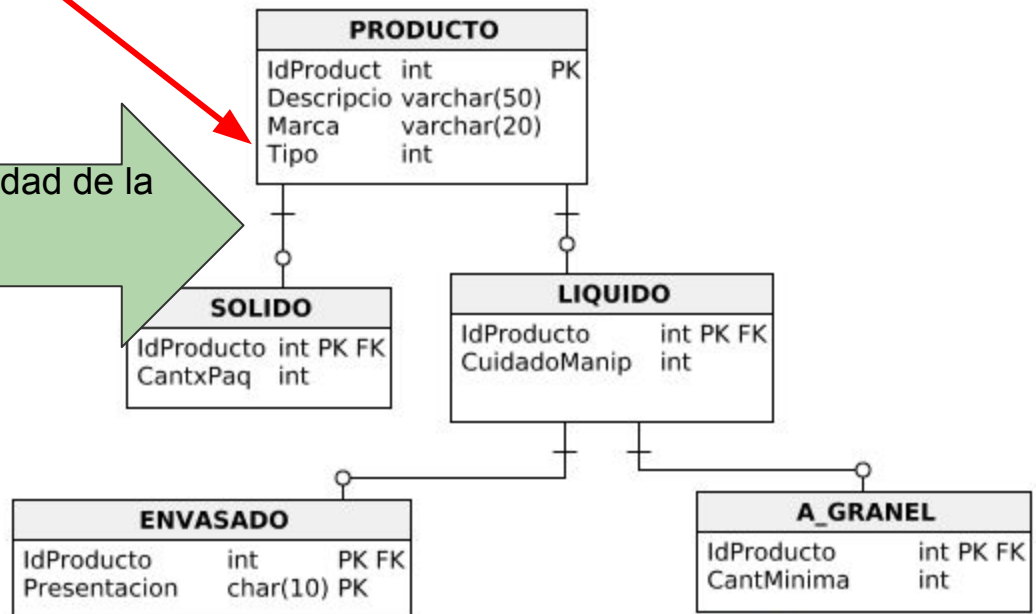
Derivación de Jerarquías

- Se crea una tabla por la **entidad supertipo** (con los atributos en común incluido su identificador) y una tabla por cada una de las **entidades subtipo** (con los atributos propios).
- La **clave de la tabla subtipo** es la clave de la tabla del **supertipo**.
- Para las jerarquías exclusivas, que deben incluir el **atributo discriminante (tipo)**, éste se debe agregar a la tabla correspondiente a la entidad supertipo

Derivación de Jerarquías



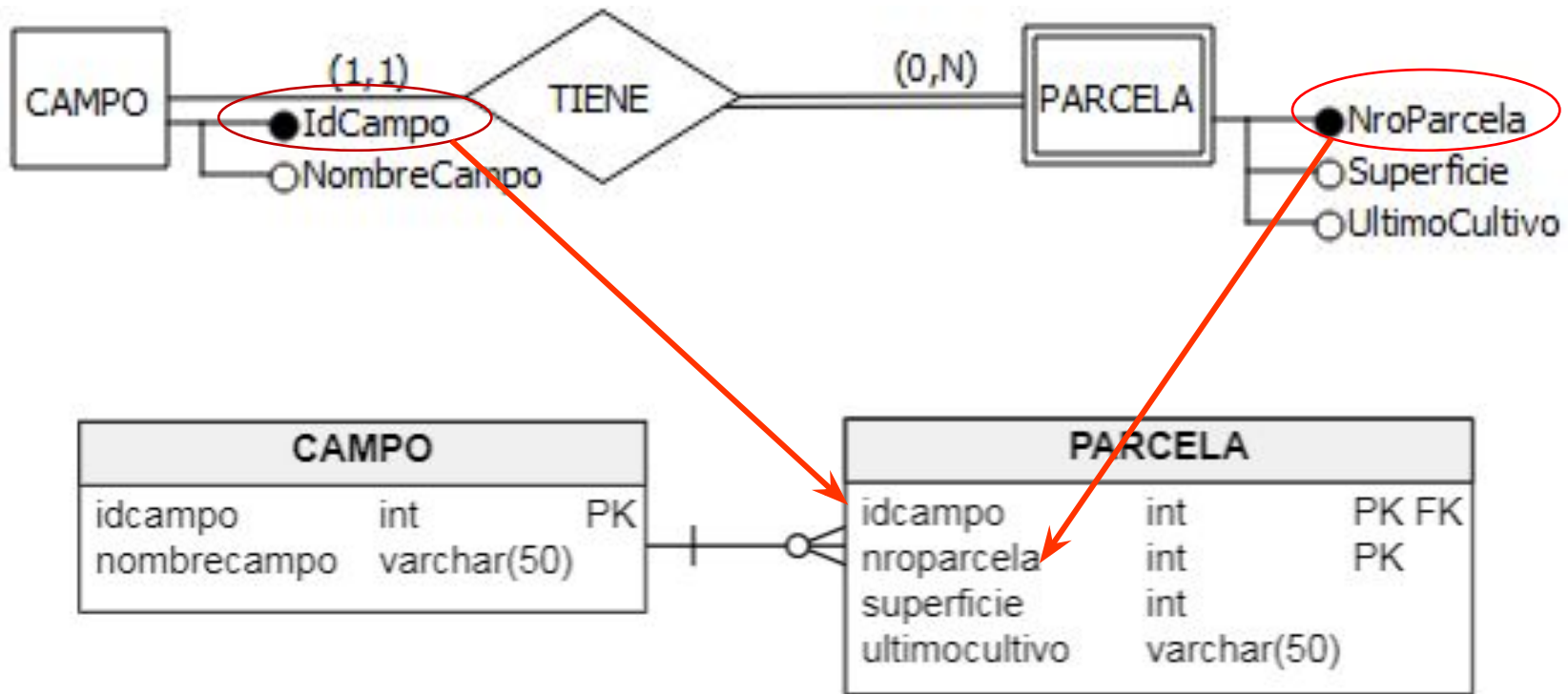
No olvidarse de cambiar la cardinalidad de la relación!!!



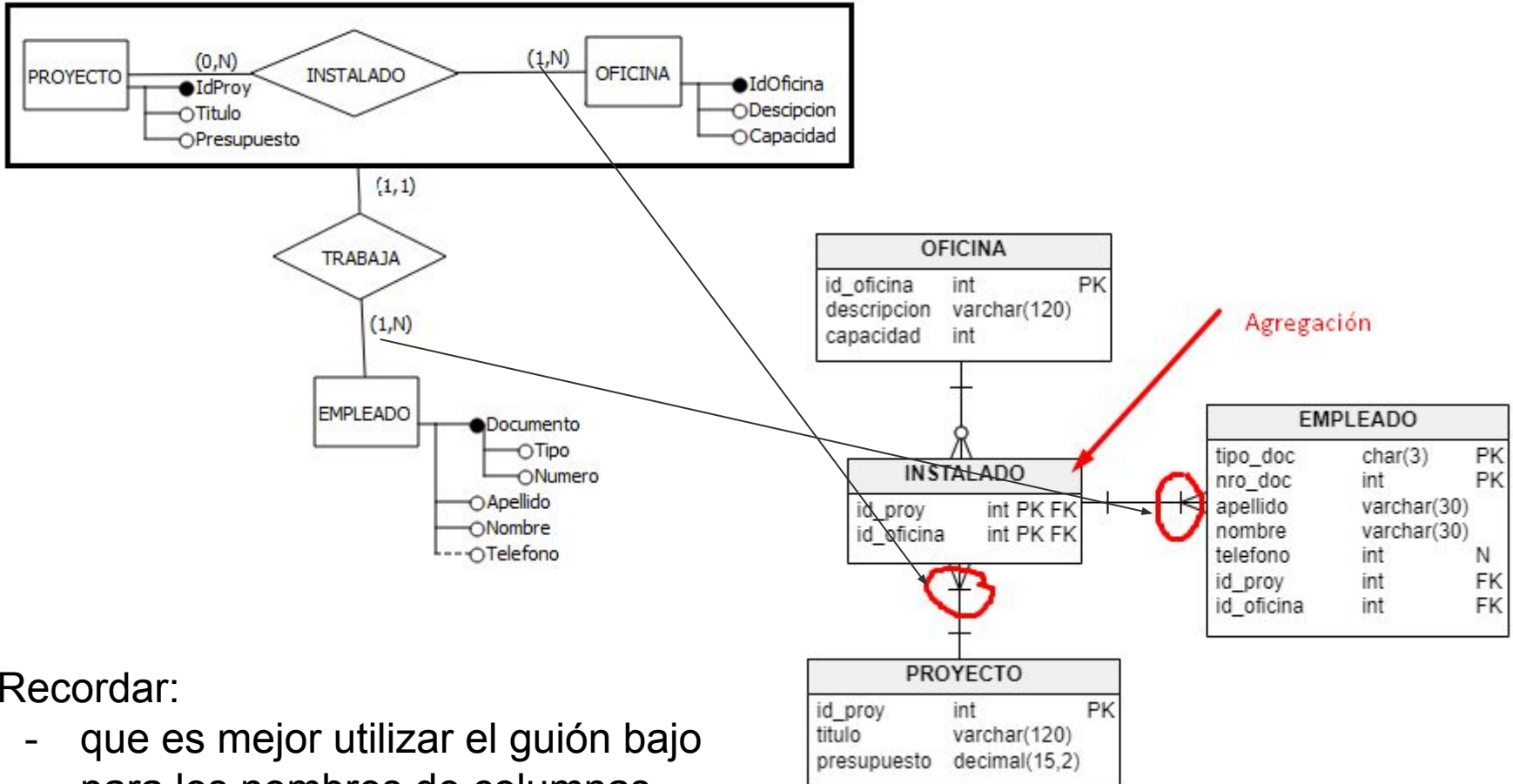
Derivación de Entidades Débiles

Entidades Débiles: tienen dependencia de existencia y de identificación.

Su clave se forma con el identificador propio (clave parcial) más el identificador (clave) de la entidad fuerte,



Derivación de Agregaciones



Recordar:

- que es mejor utilizar el guión bajo para los nombres de columnas
- modificar las cardinalidades mínimas en las relaciones

Lenguaje SQL

La definición de los datos se realiza a través de sentencia SQL

- SQL son las siglas de Structured Query Language (Lenguaje de Consulta Estructurado)
- Sus comandos permiten definir la semántica del esquema relacional: que tablas o relaciones se establecen, sus posibles valores (dominios), asociaciones, restricciones, etc.
- Los datos o información de dichas tablas las guarda el SGBD en tablas propias denominadas tablas de **metadatos**.
- El nombre de las tablas deben ser único dentro de cada esquema.
- Una tabla en una base de datos relacional es similar a una tabla en papel, posee columnas y filas.



Structured Query Language

Es un lenguaje para definición y manipulación de datos

- desarrollado inicialmente en laboratorios de investigación de **IBM**
- se transformó en estándar en 1986 (SQL-86) y ha tenido numerosas revisiones
- SQL:1999 incorporó triggers y características OO (SQL3)
- Nuevas versiones: incorporan nuevas características, no sustancialmente diferentes de SQL3

Es Declarativo: se indica qué datos se requieren, sin especificar cómo

- **Lenguaje de Definición de Datos (DDL):** permite crear y modificar el esquema de la base, tablas, restricciones, vistas, etc.
- **Lenguaje de Manejo de Datos (DML):** permite consultar y actualizar datos (inserción, modificación, eliminación)



Structured Query Language

Otras posibilidades:

- **SQL empotrado:** Manipulación de datos empotrada desde un lenguaje anfitrión
- **SQL Procedural:** Lenguaje de programación para escribir PSM (Persistent Stored Modules): Triggers, Funciones y Stored Procedures
- **Lenguajes de 4º Generación (4GL):** Generadores de formularios, informes, gráficos



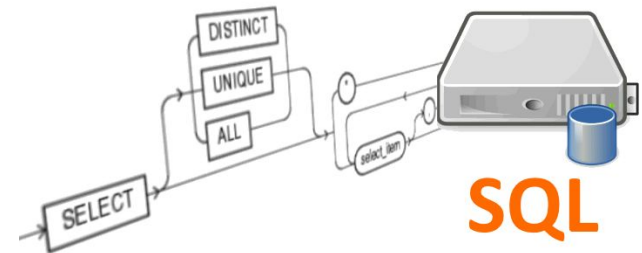
Structured Query Language

Por ejemplo algunas sentencias SQL para la definición de datos (DDL-Data Definition Language):

CREATE TABLE <nom_tabla> (....); → creación de una tabla

ALTER TABLE <nom_tabla> ... ; → modificación de una tabla

DROP TABLE <nom_tabla> ; → eliminación de una tabla





Structured Query Language

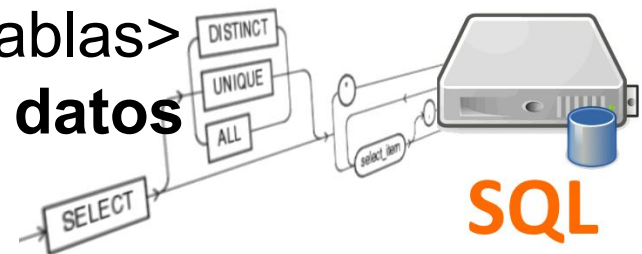
Por ejemplo algunas sentencias para el manejo de datos (DML - Data Manipulation Language):

INSERT INTO <nom_tabla> (...) **VALUES** (....); →
inserción de registros

UPDATE <nom_tabla> **SET** (...) **WHERE** (....); →
modificación de registros

DELETE FROM <nom_tabla> **WHERE** (....); → borrado
de registros

SELECT <lista_atrib> **FROM** <lista_tablas>
WHERE (....); → consulta de datos



Sintaxis para las Sentencias SQL

- ***{Alternativas}***, entre llaves se colocarán los términos que pueden repetirse, es decir darse una o más veces en la misma sentencia

[Opcional], entre corchetes se colocarán las palabras que son opcionales en la sentencia, es decir que pueden colocarse o pueden obviarse

Sentencia Create Table

```
CREATE TABLE [ IF NOT EXISTS ] nombre_tabla (  
  
    { nombre_columna tipo_dato [NOT NULL | NULL | DEFAULT default_expr ]  
    [ restricción_de_columna [ ... ], }  
    { restricción_de_tabla }  
  
    );
```

IF NOT EXISTS significa: SI NO EXISTE, por lo tanto, esto es útil para validar que la tabla sea creada en caso de que no exista, si existe y se ejecutara la sentencia sin el IF NOT EXISTS la sentencia daría error.

donde restricción_de_columna es:

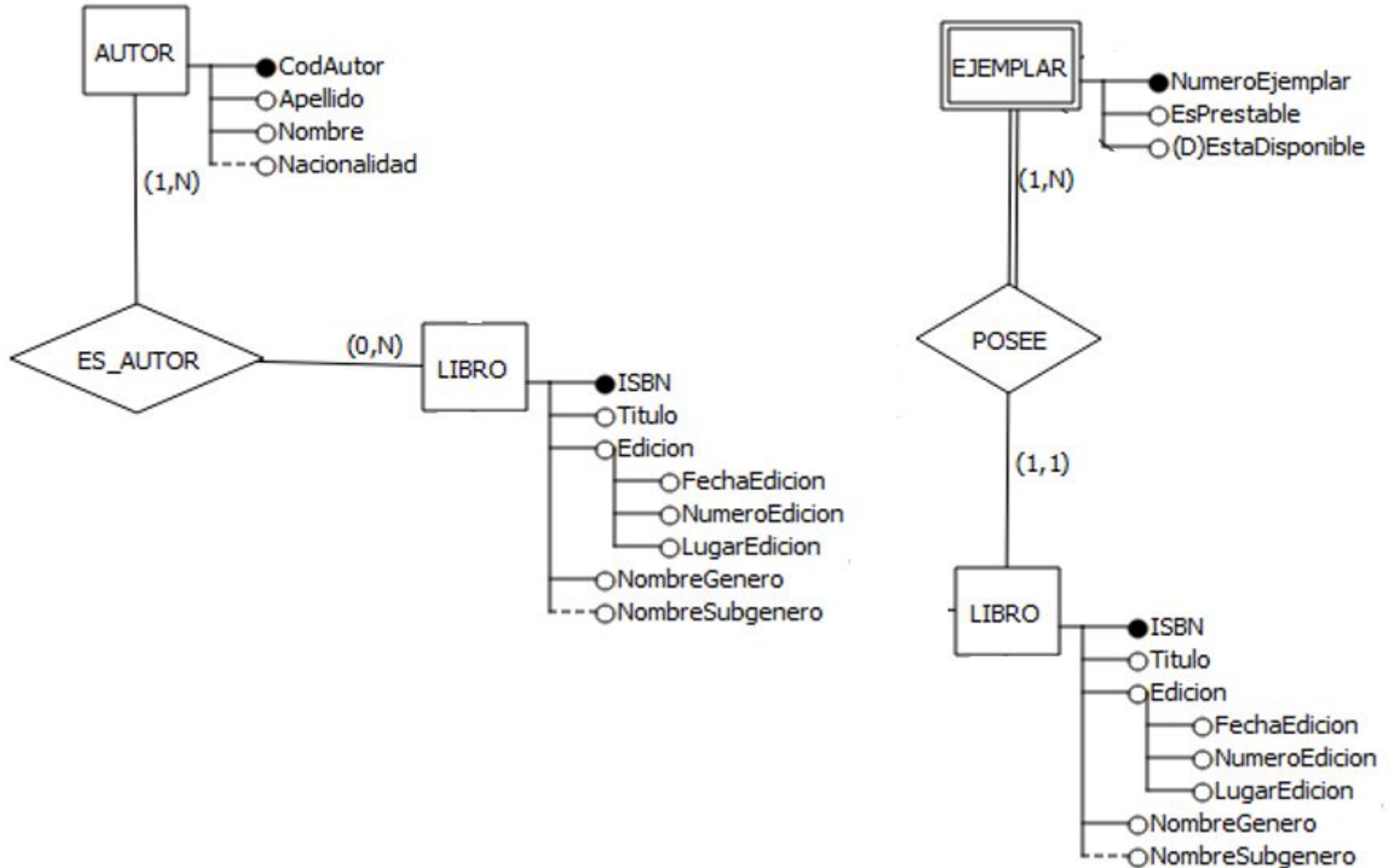
```
[ CONSTRAINT constraint_name ]  
{ UNIQUE index_parameters |  
  PRIMARY KEY index_parameters |  
  REFERENCES reftable [ ( refcolumn ) ] }
```

Sentencia Create Table

y restricción_de_tabla es:

```
[ CONSTRAINT constraint_name ]  
{ UNIQUE ( column_name [, ... ] ) |  
  PRIMARY KEY ( column_name [, ... ] ) |  
  FOREIGN KEY ( column_name [, ... ] ) }  
}
```

Ejercicios



Modificación de Tablas

Una vez creada una tabla es posible realizar algunas modificaciones en su definición (ej, agregar o quitar columnas, especificar valores por defecto, incorporar restricciones o quitarlas, etc. **La sintaxis de PostgreSQL**

`ALTER TABLE tabla ADD COLUMN columna tipo;` *Agrega una columna*

`ALTER TABLE tabla DROP COLUMN columna;` *Borra una columna*

`ALTER TABLE tabla RENAME COLUMN
 Columna_vieja TO columna_nueva;` *Renombra una columna*

Modificación de Tablas

ALTER TABLE *tabla* *columna* TYPE *nuevo_tipo*; *Cambia el tipo de dato*

ALTER TABLE *tabla* ALTER COLUMN *columna*
[SET DEFAULT *value* | DROP DEFAULT] *Asigna o elimina el valor por defecto*

ALTER TABLE *tabla* ALTER COLUMN *columna*
[SET NOT NULL | DROP NOT NULL] *Asigna o elimina la restricción de nulidad*

ALTER TABLE *tabla* *Agrega una restricción*
ADD CONSTRAINT *nombre definición_de_constraint*

Modificación de Tablas - Ejemplos

ALTER TABLE Alumno

ADD COLUMN condicion VARCHAR(10) DEFAULT 'Regular';

→ *Incorpora una nueva columna en Alumno con un valor por defecto*

ALTER TABLE Instituto

DROP COLUMN encargado;

→ *elimina la columna encargado de Alumno*

ALTER TABLE Curso

ADD CONSTRAINT U_tit UNIQUE (titulo);

→ *define una restricción de unicidad para el título del Curso*

ALTER TABLE Ofrece

DROP CONSTRAINT Fk_Ofrece_Cur;

→ *elimina la restricción de clave extranjera en Ofrece*

Modificación de Tablas - Ejemplos

```
ALTER TABLE Alumno ALTER COLUMN condicion  
SET DEFAULT 'Libre';
```

→ *Define un valor por defecto para una columna*

```
ALTER TABLE Alumno ALTER COLUMN condicion  
DROP DEFAULT;
```

→ *Elimina la definición del valor por defecto para una columna*

```
ALTER TABLE Alumno ALTER COLUMN tutor  
SET NOT NULL;
```

→ *Define una restricción de nulidad para la columna (OJO)!!!*

```
ALTER TABLE Alumno ALTER COLUMN tutor  
DROP NOT NULL;
```

→ *Elimina una restricción de nulidad para la columna*

```
ALTER TABLE Alumno condicion TYPE VARCHAR(30);
```

Modificación de Tablas - Ejemplos

```
ALTER TABLE Alumno
```

```
ADD CONSTRAINT Pk_Alumno PRIMARY KEY (LIBRETA) ;
```

→ *Incorpora la restricción de clave primaria a la tabla Alumno;*

```
ALTER TABLE Alumno
```

```
ADD CONSTRAINT Fk_Alumno_Univ FOREIGN KEY (nom_univ)  
REFERENCES Universidad (nom_univ) ;
```

→ *Incorpora la restricción de clave extranjera a la tabla Alumno que referencia a la tabla Universidad;*

Borrado de Tablas

DROP TABLE nombre_tabla [CASCADE | RESTRICT]

→ *Se elimina la definición de la tabla y todas las filas que contiene*

- Si es RESTRICT, se rechaza si hay objetos definidos a partir de la tabla (es la opción por defecto)
- Si es CASCADE, se eliminan todos los objetos dependientes de la tabla (también los objetos que dependan a su vez de ellos)

→ *tener precaución en su uso*



Carga de datos

Sitio para generar datos aleatoriamente

<https://www.mockaroo.com/>