Desarrollo de Sistemas de Información

Tema 3. Diseño Lógico

Marta Elena Zorrilla Pantaleón

Dpto. de Matemáticas, Estadística y Computación

Repaso modelo relacional

Marta Zorrilla

Universidad de Cantabria

El Modelo Relacional

Personal

NOMBRE	PROFESION	LOCALIDAD
Pedro	profesor	Santander
Luis	estudiante	Santander
María	estudiante	Las Palmas
Ana	estudiante	Madrid

Los datos se conciben agrupados en forma de tablas

Cada fila establece una relación entre un conjunto de valores

Los operadores de consulta generan nuevas tablas

```
SELECT NOMBRE, LOCALIDAD FROM Personal
WHERE PROFESION = 'estudiante'
```

NOMBRE	LOCALIDAD
Luis	Santander
María	Las Palmas
Ana	Madrid

El Modelo Relacional (y 2)

BANCOS

ENTIDAD	NOMBRE
0893	Santander
0059	Popular
3428	BBVA
5632	Banesto

- Toda tabla tiene una columna o conjunto de columnas que permiten identificar cada una de sus filas; éstas componen la llamada **clave principal** (**Primary Key,** PK) de la tabla.
- Los valores de la clave principal no se pueden repetir (Entidad en Bancos, y la pareja Entidad-Codigo_Oficina en Oficinas) .

OFICINAS

ENTIDAD	CODIGO_OFICINA	POBLACION	DIRECCION
0893	001	Madrid	Castellana, 73
3428	022	Las Palmas	Triana, 21
0893	022	Gáldar	R. Moreno, 3
5632	213	Oviedo	Uría, 43
0893	300	Barcelona	Diagonal, 435

- Unas tablas se refieren a otras mediante vínculos de tipo jerárquico.
- Este vínculo de referencia entre dos tablas se establece mediante columnas de igual tipo de dato en las dos tablas y se denomina **clave ajena o Foreign Key** (FK).
- La referencia de una fila de una tabla a otra de la otra tabla se produce cuando ambas tienen el mismo valor (columna Entidad en Oficinas con relación a Entidad en Bancos).

El Modelo Relacional (y 2)

TIPOS DE DATOS: cada columna de una tabla tiene asociado un tipo de dato. Existen un subconjunto estándar pero hay otros dependientes del gestor que se utilice

Cadena de caracteres (char).

Cada carácter requiere un byte para su almacenamiento.

Numérico (numeric).

Enteros: Cortos (smallint).
Largos (integer).
Decimales: definidos por su precisión y escala (decimal)
Notación científica: Simple precisión (smallfloat)
Doble precisión (float)

Fecha (date) **y hora** (datetime).

Diferentes opciones según nivel de precisión.

Objeto grande (large object, **LOB**).

Binary large object (blob).
Character large object (clob).

Tipos definidos por el usuario.

El Modelo Relacional (y 4)

ÍNDICES: estructuras de datos adicionales que permiten

- **Realizar búsquedas más ágiles en las tablas….**

  aunque suponen una sobrecarga para realizar las actualizaciones de datos

  ej: campo "título" en una tabla que recoja los libros disponibles en una biblioteca. Se justifica por ser el campo sobre el que se realizan más consultas

  Conviene definirlos sobre las columnas con FK

- **Establecer restricciones de unicidad**

  no permitir repeticiones de un valor en la columna o columnas afectadas por el índice

  ej: campo "ISBN" en una tabla que recoja los libros disponibles en una biblioteca

EJEMPLO

```
CREATE TABLE Proveedores (
 codigpro        CHAR(4)   NOT NULL   CONSTRAINT id_pro PRIMARY KEY,
 cifpro          CHAR(12)  NOT NULL   CONSTRAINT u_cif UNIQUE,
 nombrpro        CHAR(30)  NOT NULL,
 direcpro        CHAR(30)  NOT NULL,
 cpostpro        CHAR(5)    NOT NULL CHECK (cpostpro like '[0-9][0-9][0-9][0-9][0-9]'),
 localpro        CHAR(20)  NOT NULL,
 telefpro        CHAR(17)  NOT NULL,
 faxpro          CHAR(17),
 emailpro        CHAR(25),
 procepro        CHAR(5)  NOT NULL CHECK (procepro in ('UE', 'No UE')))
```

```
CREATE TABLE Articulos (
  codigart       CHAR(6)   NOT NULL   CONSTRAINT id_art PRIMARY KEY,
  descrart       CHAR(40) NOT NULL,
  preunart       DECIMAL(9,2)  NOT NULL,
  stockart       INTEGER  NOT NULL CHECK (stockart >=0),
  stockmin       INTEGER  NOT NULL CHECK (stockmin>=0),
   fecbaja       DATE)
```

```
CREATE TABLE Pedidos (
  numped          INTEGER          NOT NULL   CONSTRAINT id_ped PRIMARY KEY,
  fechaped        DATE             NOT NULL   DEFAULT getdate(),
  codigpro        CHAR(4)          NOT NULL,
  ivaped          DECIMAL(4,1)     NOT NULL CHECK (ivaped>0 and ivaped<100),
  fentrped        DATE             NOT NULL,
       CONSTRAINT f_pro FOREIGN KEY (codigpro) REFERENCES Proveedores (codigpro),
       CONSTRAINT c_fecha CHECK (fechaped<=fentrped))
```

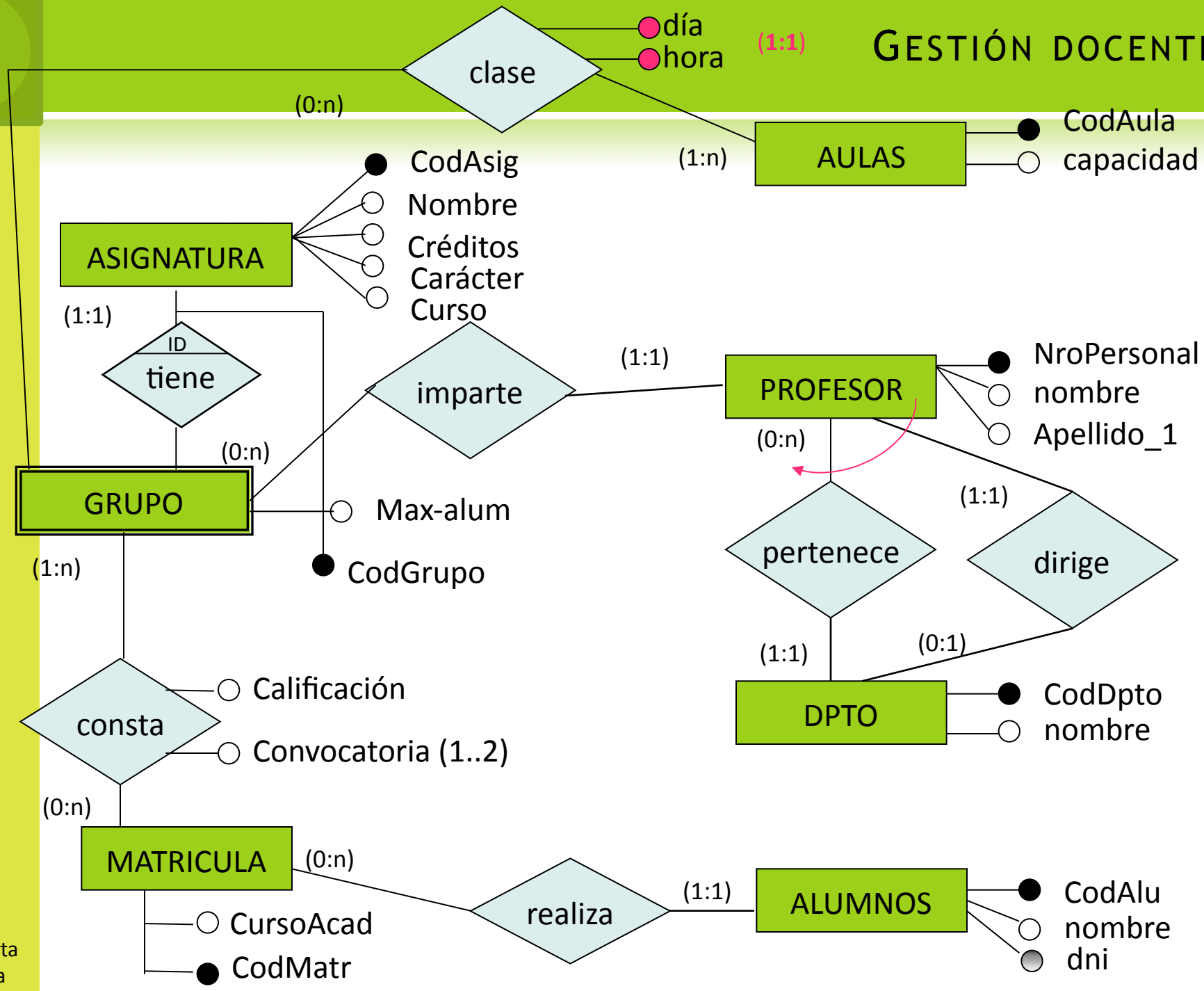
```
CREATE TABLE Lineas (
  numped          INTEGER NOT NULL,
  numlin          SMALLINT NOT NULL,
  codigart        CHAR(6)  NOT NULL,
  unilin          DECIMAL (6,0)      NOT NULL,
  preunlin        DECIMAL (9,2)      NOT NULL,
  desculin        DECIMAL (4,1)      NOT NULL CHECK (desculin>=0 and desculin<=100),
  totallin      AS ([preunlin] * [unilin] * (1 –[desculin] / 100)),
       CONSTRAINT id_lin PRIMARY KEY (numped, numlin),
       CONSTRAINT f_ped FOREIGN KEY (numped) REFERENCES Pedidos (numped),
       CONSTRAINT f_art FOREIGN KEY (codigart) REFERENCES Articulos (codigart))
```

Diseño Lógico: Del modelo ER al modelo relacional

Marta Zorrilla

Universidad de Cantabria

GESTIÓN DOCENTE



ENTIDADES

- ENTIDADES → TABLAS
  - Se conservan los atributos y la clave principal.
  - Claves candidatas, establecer restricción de unicidad.
  - Atributos compuestos → un campo por atributo
  - Atributos derivados → columnas calculadas
  - Atributos multivaluados → nueva tabla
  - Restricciones sobre atributos → lenguaje lógico

- Ej:

Asignatura (CodAsig, nombre, créditos, carácter, curso)

Alumno (CodAlu, nombre, dni)

Aula (CodAula, capacidad)

Profesor (NroPersonal, nombre, apellido_1)

Dpto (CodDpto, nombre)

Matricula (CodMatr, cursoAcad)

ENTIDADES DÉBILES

- ENTIDADES DÉBILES → TABLAS
  - Conserva todos sus atributos y se añade la clave principal de la entidad fuerte de la que depende.
  - Si la relación es de identificación, la clave principal la forma algún atributo de la entidad débil y la clave principal de la entidad fuerte.
  - Si la relación es de existencia, tendrá su propia clave, pero se establecerá borrado y actualización en cascada con respecto la entidad fuerte (también se podría restringir).

- Ej:

  Grupo (<u>CodAsig, CodGrupo</u>, max_alum)

RELACIONES

- RELACIONES → TABLAS
  - La tabla a la que da lugar tendrá como atributos las claves principales de las entidades que se conectan y los atributos de la relación.
  - La elección de la clave principal depende de la cardinalidad de la relación

M:N La PK estará formada, al menos, por las PK de las entidades que relaciona. Dimensión temporal.

CONSTA (CodMatr, CodAsig, CodGrupo, Convocatoria, calificación)

1:N La PK estará formada por la PK de la entidad que participa con cardinalidad N

PERTENECE (CodProf, CodDpto)

1:1 La PK estará formada por una de las PK de las entidades que relaciona. La otra actuará como clave candidata.

DIRIGE (CodProf, CodDpto)

Esquema obtenido

Asignatura (CodAsig, nombre, créditos, carácter, curso)

Alumno (CodAlu, nombre, dni)

Aula (CodAula, capacidad)

Profesor (NroPersonal, nombre, apellido_1)

Dpto (CodDpto, nombre)

Matricula (CodMatr, cursoacad)

Grupo (CodAsig, CodGrp, max_alum)

CONSTA (CodMatr, CodAsig, CodGrupo, Convocatoria, calificación)

PERTENECE (NroPersonal, CodDpto)

DIRIGE (NroPersonal, CodDpto)

IMPARTE (CodAsig, CodGrupo, NroPersonal)

CLASE (CodAsig, CodGrupo, CodAula, hora, dia)

REALIZA (NumMatr,CodAlum)

N:N todo clave si más de un grupo en aula o no se quieren recoger huecos libres

ESQUEMA OBTENIDO: REDUCCIÓN DE TABLAS

Asignatura (CodAsig, nombre, creditos, carácter, curso)

Alumno (CodAlu, nombre, dni)

Aula (CodAula, capacidad)

Profesor (NroPersonal, nombre, apellido1) → Profesor (NroPersonal, nombre, apellido1, CodDpto)

Dpto (CodDpto, nombre) → Dpto (CodDpto, nombre, CodProfDirige)

Matricula (CodMatr, cursoacad) → Matricula (CodMatr, cursoacad,CodAlu)

Grupo (CodAsig, CodGrp, max_alum) → Grupo (CodAsig,CodGrp,max_alum,NroPersonal)

CONSTA (CodMatr, CodAsig, CodGrp, Convocatoria, calificación)

PERTENECE (NroPersonal, CodDpto)

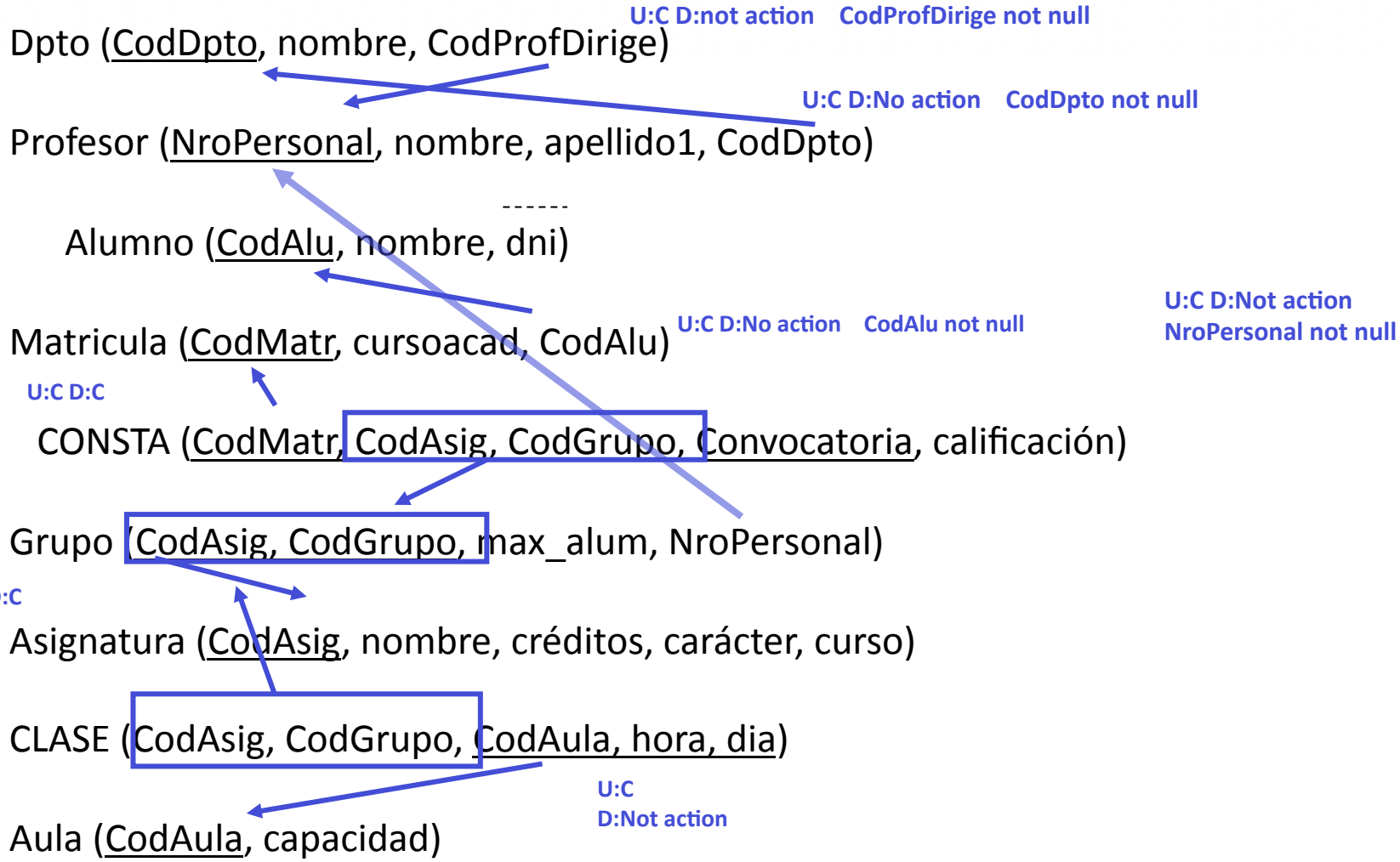
DIRIGE (NroPersonal, CodDpto)

IMPARTE (CodAsig, CodGrupo, NroPersonal)

CLASE (CodAsig, CodGrupo, CodAula, hora, dia)

REALIZA(CodMatr,CodAlu)

ESQUEMA DEFINITIVO



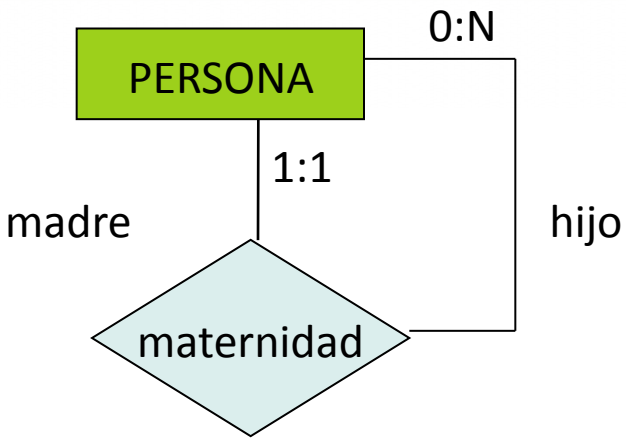
ESQUEMA DEFINITIVO

```
CREATE ASSERTION dirige_dpto AS CHECK not exists
  (SELECT * FROM Dpto WHERE CodProfDirige NOT IN
    (SELECT NroPersonal FROM Profesor where
       Profesor.CodDpto=Dpto.CodDpto ))
```

```
CREATE ASSERTION convocatorias
CHECK not exists ( select count(*) from CONSTA
   group by  Codmatr,CodAsig,CodGrp
   having count(*)>2)
```

```
CREATE ASSERTION maxAlumAula
CHECK not exists ( select count(*) from CLASE c , AULA a, GRUPO g
   where
   c.codAula=a.CodAula and c.CodAsig=g.CodAsig and
   c.CodGrupo=g.CodGrupo and max_alum>capacidad)
```

Relaciones unarias



Borrado: null
Modificación: cascada

a)

persona (Codper, nombre,… , codper_madre)

b)

Borrado y Modificación: cascada o not action

persona (Codper, nombre,…)

Nulos no permitidos

madre (Codper, codper_madre)

Generalización

- Crear una tabla por cada entidad que participa

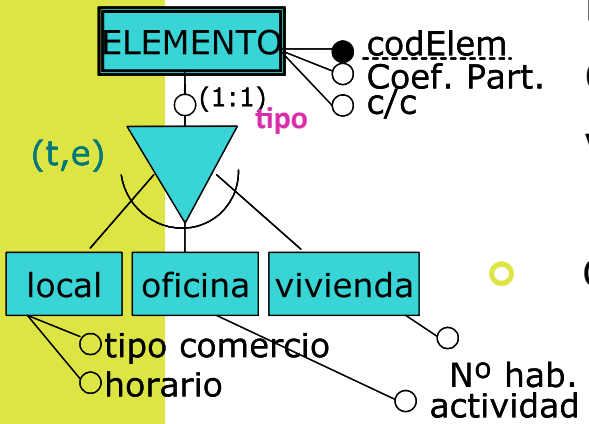
  ELEMENTO (codElem, Coef, cc)

  LOCAL(CodElem, tipo_comercio, horario)

  OFICINA(codElem, actividad)

  VIVIENDA (codElem, numHab)

- Crear una tabla para cada caso particular, eliminando la entidad de nivel superior. No frecuente.



  LOCAL(CodElem, tipo_comercio, horario , Coef, cc)

  OFICINA(codElem, actividad , Coef, cc)

  VIVIENDA (codElem, numHab , Coef, cc)

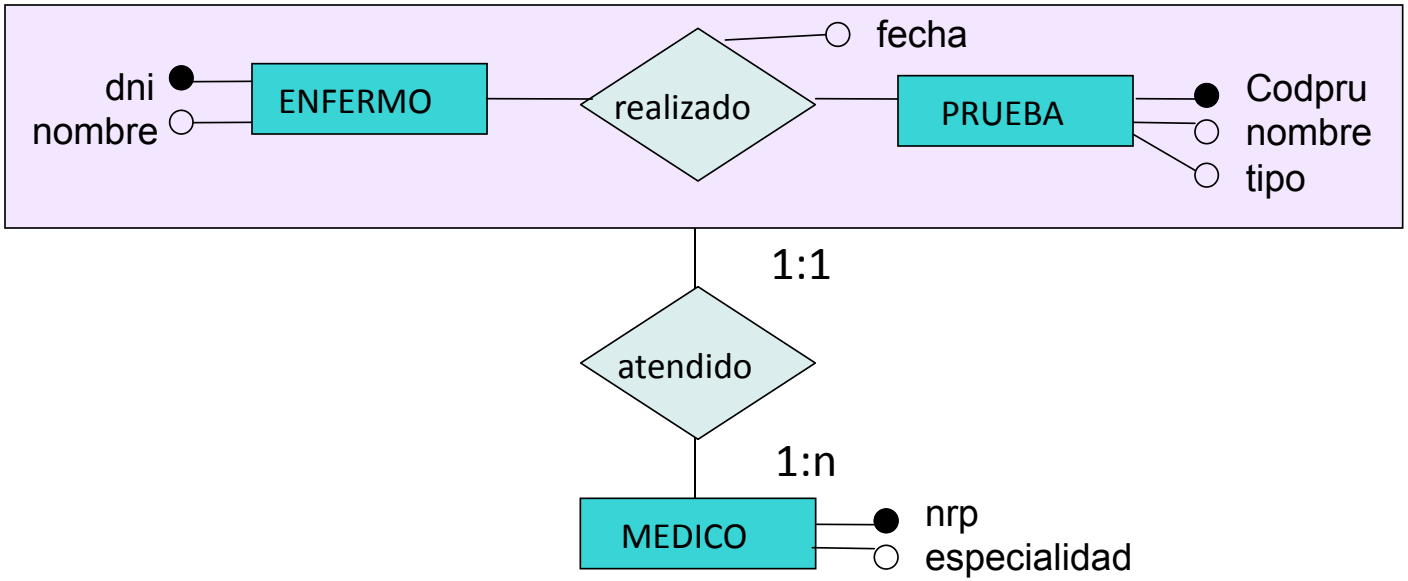
- Crear un sola relación

  ELEMENTO (CodElem, tipo, horario , tipo_comercio, actividad, numhab,Coef, cc)

Restricciones semánticas

- Totalidad/parcialidad:
  - Se controla la totalidad prohibiendo la inserción en el supertipo directamente, se hará cuando se inserte en los subtipos (disparadores)
  - La parcialidad no requiere disparadores
- Exclusividad/solapamiento
  - Se requiere un aserto (o trigger) que compruebe que si un ejemplar está en un subtipo no puede estar en otro

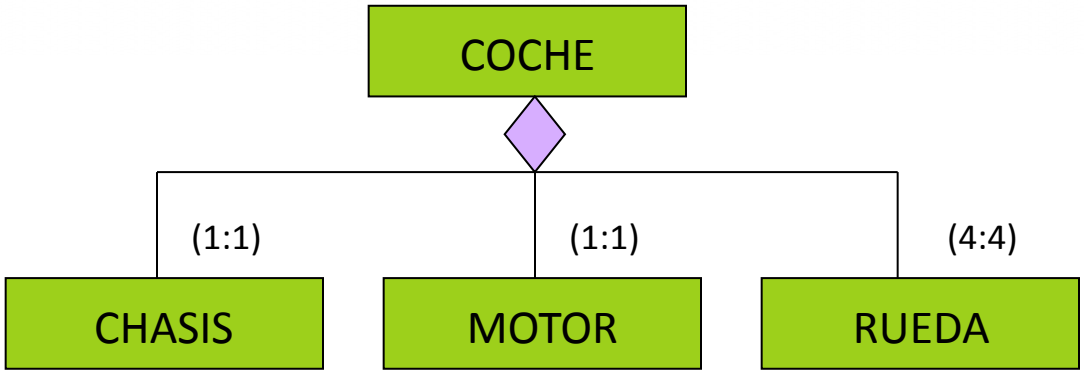
Agregación



REALIZADO(<u>dni,codpru, fecha</u>)

ATENDIDO(<u>dni,codpru, fecha,nrp</u>)

Agregación



obligatorios

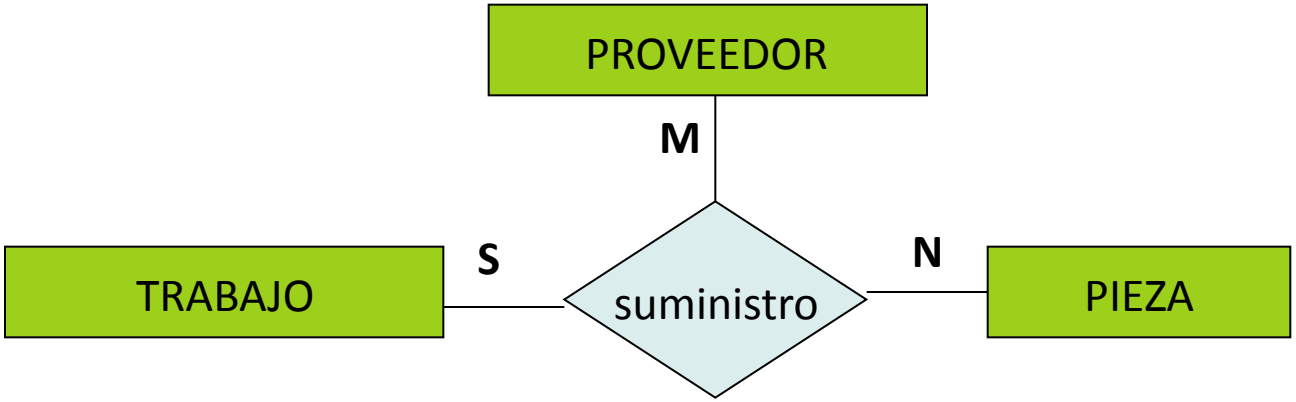
- COCHE(Idcoche, fechafabr,…, idchasis,idmotor)
- coche_ruedas(idcoche,idrueda)
- CHASIS(idchasis, nroserie, lon, anchura,…)
- MOTOR (idmotor, nroserie, rpm,…)
- RUEDA (idrueda, modelo,…)

Relaciones n-arias

- No es directo, una relación puede tener varias interpretaciones y todas ellas válidas.

- Depende de la cardinalidad:
  - N:M:S, la PK el conjunto de las PK de cada relación
  - N:M:1, la PK será el conjunto de las PK con cardinalidad distinta de 1
  - N:1:1, probablemente se dividirá en dos relaciones 1:n

DISTINTAS INTERPRETACIONES



N:M:S SUMINISTRO (CodProv, CodPieza, CodTrab)

N:M:1 SUMINISTRO (CodProv, CodPieza, CodTrab)

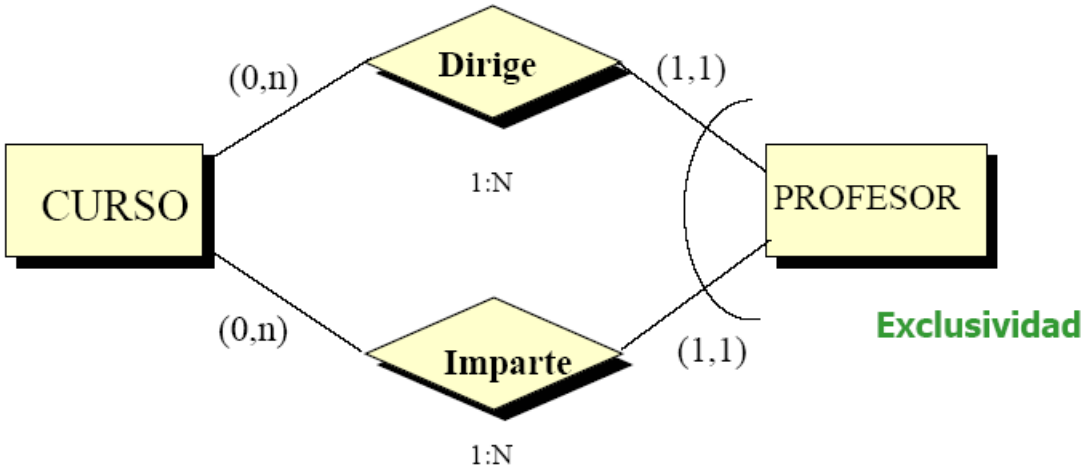
N:1:1 SUM1 (CodPieza, CodProv)

SUM2 (CodPieza, CodTrab)

A TENER EN CUENTA

- Existen restricciones del modelo ER que deben transformarse al modelo relacional mediante check, asertos o disparadores
  - Cardinalidades mínimas en relaciones N:M y 1:N (cuando no se controla con NOT NULL)
  - Cardinalidades máximas
  - Restringir el valor de un determinado campo
  - Condición que han de cumplir los campos de una determinada tabla
  - Exclusividad e inclusividad entre relaciones
  - Exclusividad en generalizaciones
  - Insertado y borrado en generalizaciones
  - Atributos derivados (cuando no es campo calculado)

EJ: TRANSFORMACIÓN DE RESTRICCIONES

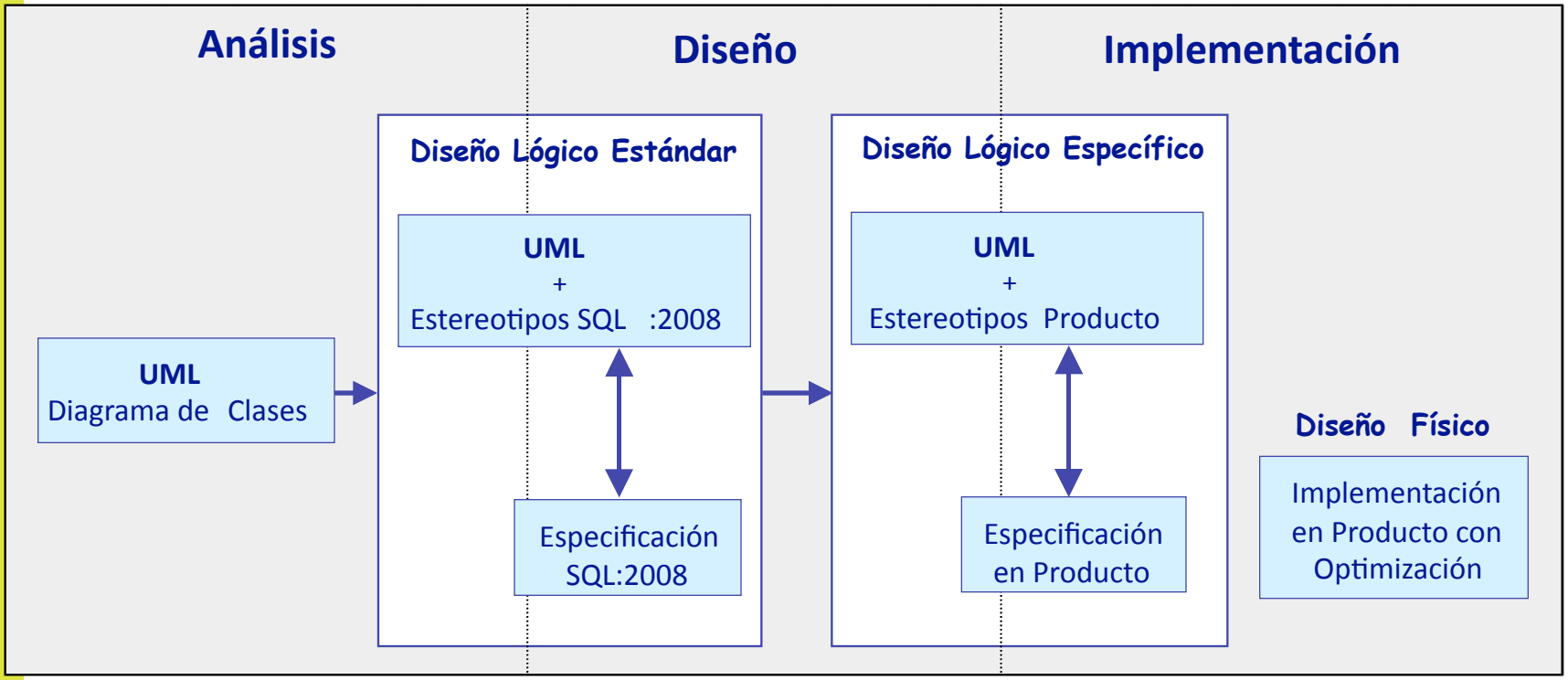



```
CREATE TABLE Curso (
        Cod_Curso Codigos_Cursos,
        Nombre Nombres, .... ,
        Cod_prof_dirige Cods_profesores,
        Cod_prof_imparte Cods_profesores,
        PRIMARY KEY (Cod_Curso)
        FOREIGN KEY (Cod_prof_dirige) REFERENCES Profesor
                ON UPDATE CASCADE
        FOREIGN KEY (Cod_prof_imparte) REFERENCES Profesor
                ON UPDATE CASCADE
        CHECK (( Cod_prof_dirige NOT IN (SELECT Cod_prof_imparte FROM Curso))
                AND
                ( Cod_prof_imparte NOT IN (SELECT Cod_prof_dirige FROM Curso)));
```

Exclusividad

Reglas de transformación UML a Relacional. Perfil UML

Etapas del diseño



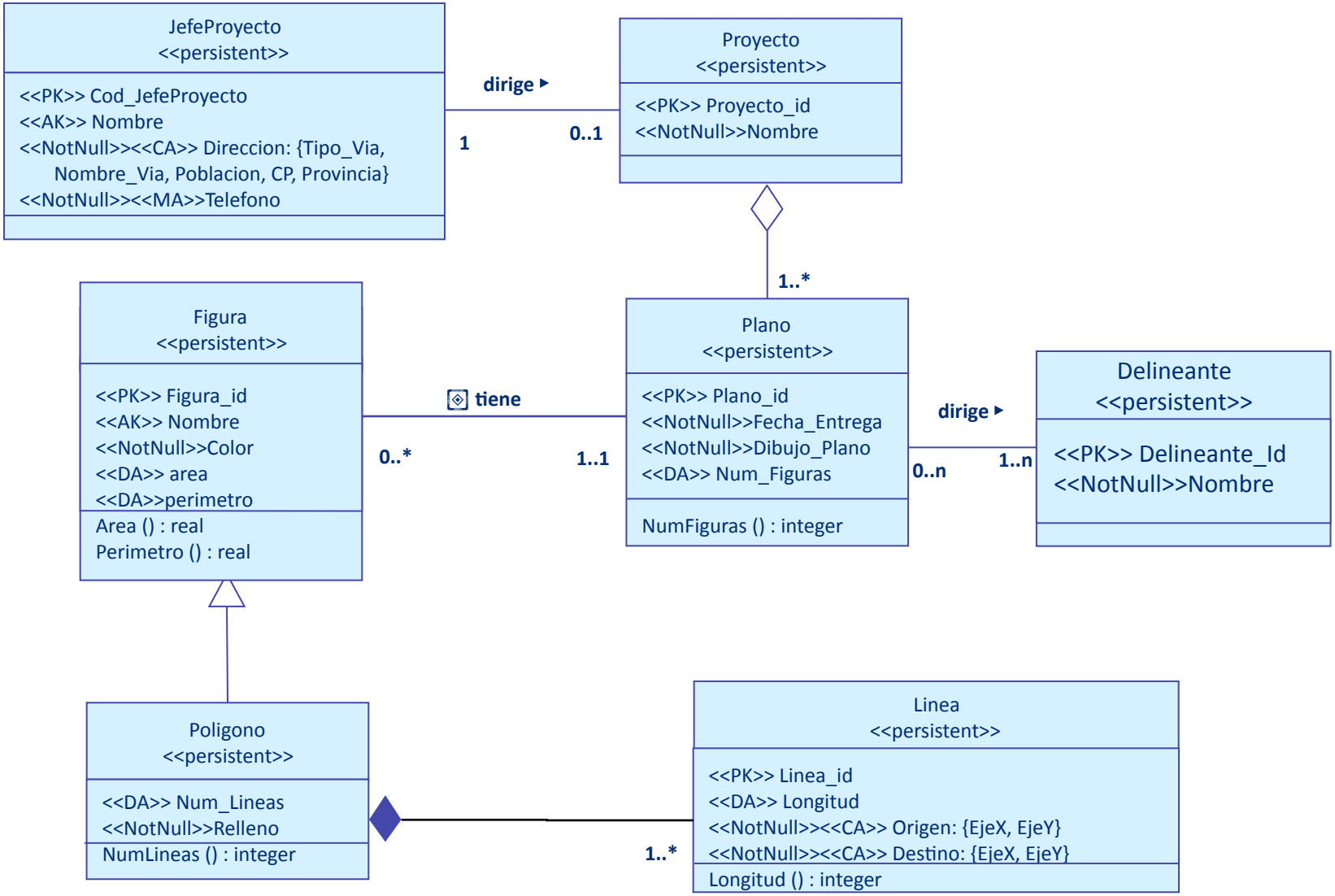
Estereotipos para el diseño de BD Relacionales

Elemento BD	Elemento UML	Estereotipo
Base de datos	Componente	**<<Database>>**
Esquema	Paquete	**<<Schema>>**
Tabla	Clase	**<<Table>>**
Vista	Clase	**<<View>>**
Índice	Clase	**<<Index>>**
Columna	Atributo	**<<Column>>**
Clave ppal	Atributo	**<<PK>>**
Clave ajena	Atributo	**<<FK>>**
Clave alternativa	Atributo	**<<AK>>**
Atrib. Multievaluado	Atributo	**<<MA>>**
Atrib. derivado	Atributo	**<<DA>>**
Atrib. Compuesto	Atributo	**<<CA>>**
Requerido	Atributo	**<<NOT NULL>>**
Único	Restricción	**<<UNIQUE>>**
Restr. Dominio	Restricción	**<<CHECK>>**
Regla negocio	Restricción	**<<TRIGGER>>**

Ejemplo

- Un estudio de arquitectura desea crear una base de datos para gestionar sus proyectos.
  - Cada proyecto tiene un código y un nombre. Un proyecto tiene uno y solo un jefe de proyecto y un jefe de proyecto sólo puede estar involucrado en un proyecto o en ninguno.
  - De cada jefe de proyecto se desean recoger sus datos personales (código, nombre, dirección y varios teléfonos). Un jefe de proyecto se identifica por un código. No hay dos nombres de jefe de proyecto iguales.
  - Un proyecto se compone de una serie de planos, pero éstos se quieren guardar de modo independiente al proyecto. Es decir, si en un momento dado se dejara de trabajar en un proyecto, se desea mantener la información de los planos asociados.
  - De los planos se desea guardar su número de identificación, la fecha de entrega y un dibujo del plano general con información acerca del número de figuras que contiene. Además se indicarán los delineantes que participan en su confección.
  - Los planos tienen figuras. De cada figura se desea conocer, el identificador, el nombre (único), el color, el área y el perímetro. Algunas de las figuras son polígonos y de estos se desea conocer si están rellenos o no, el número de líneas que tienen y las líneas que lo forman.
  - De cada línea que forma parte de un polígono se desea conocer el punto de origen y el de fin (según sus coordenadas, X e Y), así como la longitud. Cada línea tiene un identificador que permite diferenciarlo del resto. La longitud de la línea se puede calcular a partir de sus puntos origen y final.
  - Todos los campos se establecen como requeridos

Ejemplo en perfil UML



Reglas de transformación: clases

1. Cada clase (no afectada por jerarquía) mapea en tabla

SQL:2008

```
CREATE TABLE Plano
( Plano_Id          INTEGER,
  Fecha_Entrega     DATE,
  Dibujo_Plano      BLOB);
```

Reglas de transformación: clases-Atributos

PERFIL UML	SQL:2008
Atributo	Atributo del tipo
<<PK>>	PK en la tabla
<<AK>>	UNIQUE en la tabla
<<CA>>	Un campo para cada atributo en la tabla
<<MA>>	Tabla
<<DA>>	Campo calculado - función/ Campo físico actualizado por disparador
Niveles de visibilidad	---
<<Método>>	Función / Procedimiento

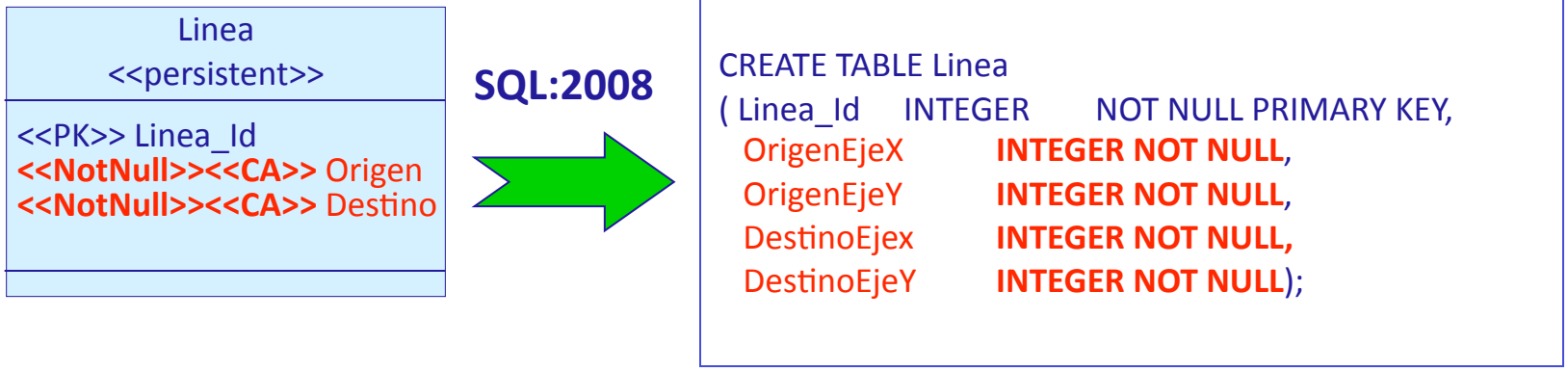
Reglas de transformación: clases-Claves Principales y Alternativas

Figura <<persistent>>
<<PK>> Figura_Id **<<AK>>** Nombre Color

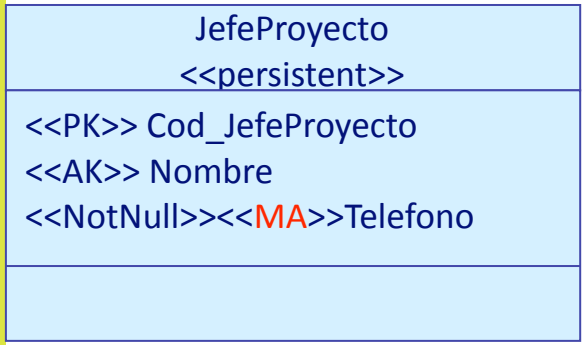
SQL:2008

```
CREATE TABLE Figura
( Figura_Id          INTEGER NOT NULL,
  Nombre             VARCHAR(30) NOT NULL,
  Color              VARCHAR(15),
 PRIMARY KEY (Figura_Id),
 UNIQUE (Nombre));
```

Reglas de transformación: clases-Atributos compuestos



Reglas de transformación: clases-Atributos multivaluados

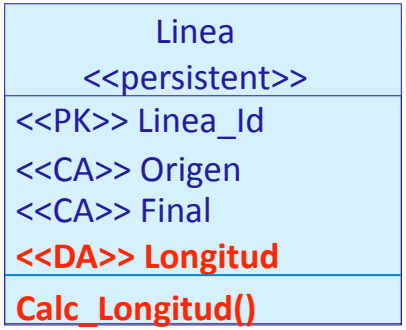


SQL:2008

```
CREATE TABLE JefeProyecto
( Cod_JefeProyecto        INTEGER NOT NULL,
 Nombre                   CHAR(20) NOT NULL);

CREATE TABLE Jefe_tfnos
(Cod_JefeProyecto              INTEGER NOT NULL,
  Telefono                     CHAR(20) NOT NULL,
  UNIQUE (Cod_JefeProyecto , Telefono),
   FOREIGN KEY (Cod_JefeProyecto) REFERENCES
            JefeProyecto (Cod_JefeProyecto));
```

Reglas de transformación: clases-Atributos derivados

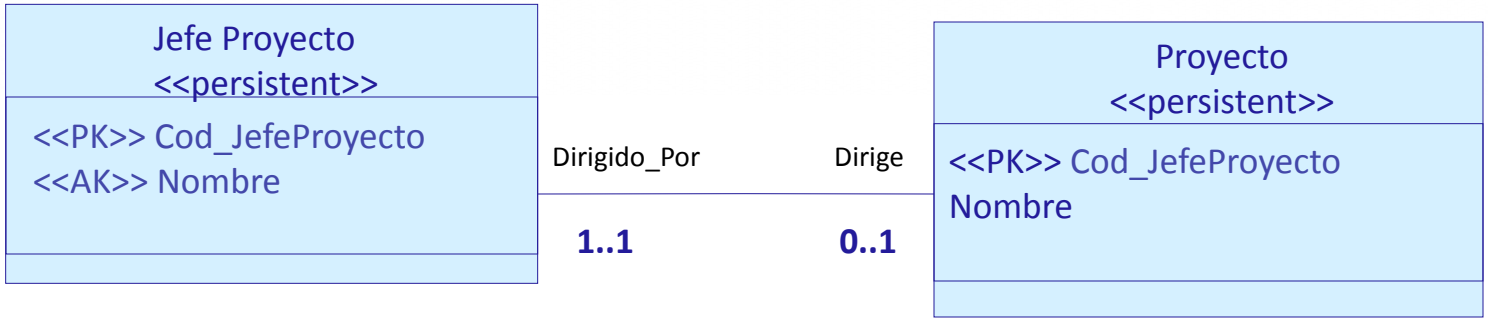


SQL:2008

```
CREATE TABLE Linea
( Linea_Id    INTEGER PRIMARY KEY,
   OrigenEjeX        INTEGER,
   OrigenEjeY        INTEGER,
   DestinoEjex       INTEGER,
   DestinoEjeY       INTEGER,
   Longitud AS  Calc_Longitud (OrigenEjeX, OrigenEjeY,
                     DestinoEjeX, DestinoEjeY);

CREATE FUNCTION Calc_Longitud (OrigenEjeX int,
OrigenEjeY, DestinoEjeX, DestinoEjeY) RETURNS INTEGER
BEGIN
 .....
END;
```

Reglas de transformación: asociación

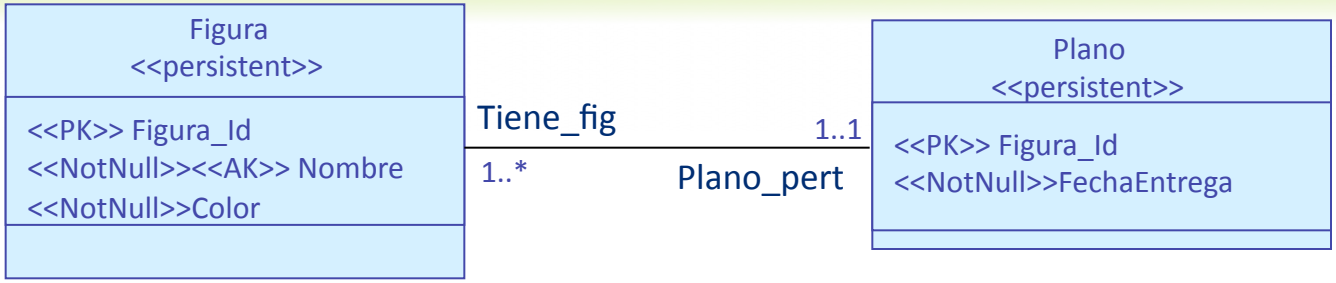


SQL:2008

```
CREATE TABLE JefeProyecto
(Cod_JefeProyecto INT NOT NULL PRIMARY KEY,
 Nombre           VARCHAR(30) NOT NULL,
 UNIQUE(Nombre));
```

```
CREATE TABLE Proyecto
( Proyecto_Id   INT NOT NULL PRIMARY KEY,
  Nombre        VARCHAR(30),
  Dirigido_Por  INT NOT NULL,
  FOREIGN KEY Dirigido_Por REFERENCES
    JefeProyecto (Cod_JefeProyecto));
```

Reglas de transformación: asociación

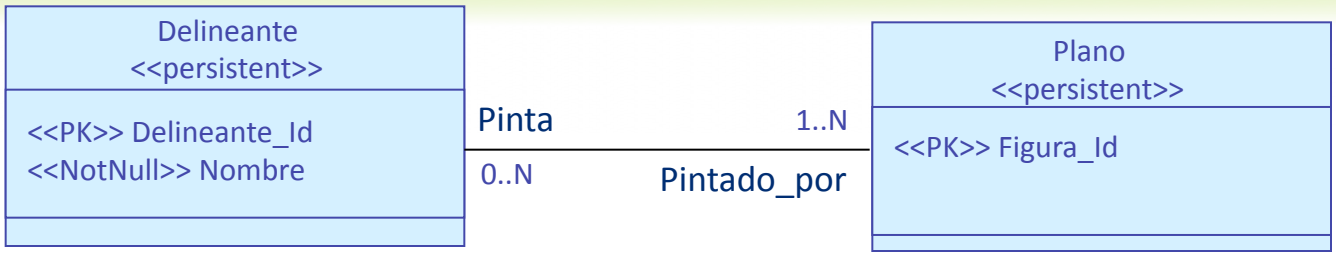


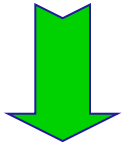
SQL:2008

```
CREATE TABLE Figura
( Figura_Id      INTEGER NOT NULL PRIMARY KEY,
  Nombre         VARCHAR(30) NOT NULL UNIQUE,
  Color  VARCHAR(10) NOT NULL,
  Plano_Pert     INTEGER NOT NULL  FOREING KEY
                        REFERENCES Plano(Plano_id));
```

```
CREATE TABLE Plano
( Plano_Id         INTEGER NOT NULL PRIMARY KEY,
  FechaEntrega  DATE NOT NULL);
```

Reglas de transformación: asociación





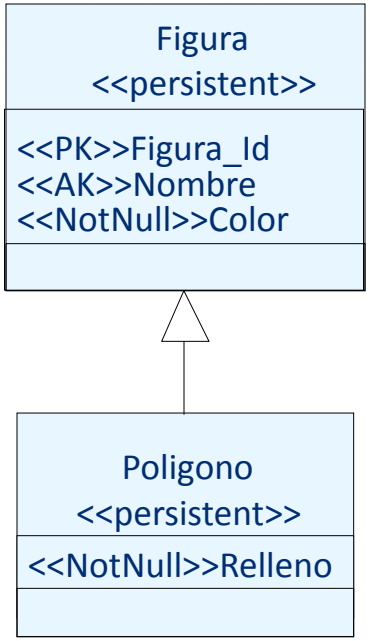
SQL:2008

```
CREATE TABLE Delineante
( Delineante_id  INTEGER NOT NULL PRIMARY KEY,
  Nombre         CHAR(20) NOT NULL);
```

```
CREATE TABLE Plano
( Plano_Id       INTEGER NOT NULL PRIMARY KEY);
```

```
CREATE TABLE Delineante_Plano
( Plano_Id       INTEGER NOT NULL,
  Delineante_id INTEGER NOT NULL,
  PRIMARY KEY(Plano_id, Delineante_id),
  FOREING KEY (Plano_id) REFERENCES Plano (Plano_id),
  FOREING KEY (Delineante_id) REFERENCES Delineante (Delineante_id)   );
```

Reglas de transformación: Generalización

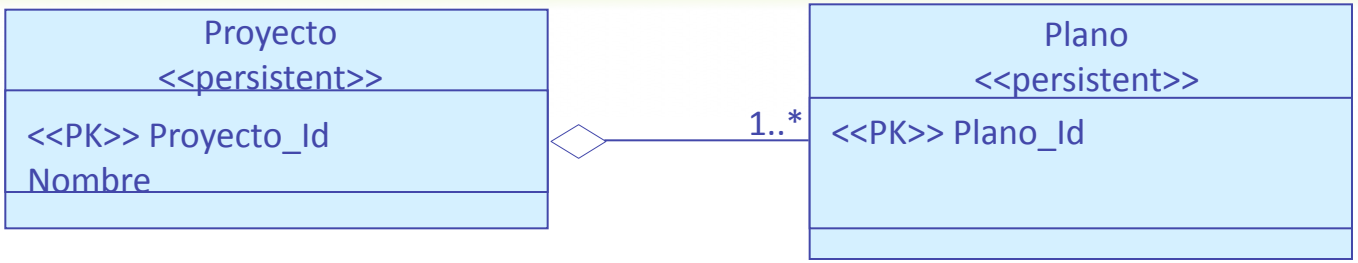


SQL:2008

```
CREATE TABLE Figura
(Figura_Id INTEGER NOT NULL PRIMARY KEY,
  Nombre VARCHAR(30) NOT NULL UNIQUE,
  Color VARCHAR(10) NOT NULL);

CREATE TABLE Poligono
(Figura_Id  INTEGER PRIMARY KEY,
 Relleno      BOOLEAN NOT NULL,
 FOREIGN KEY (Figura_id) REFERENCES Figura(Figura_id));
```

Reglas de transformación: Agregación

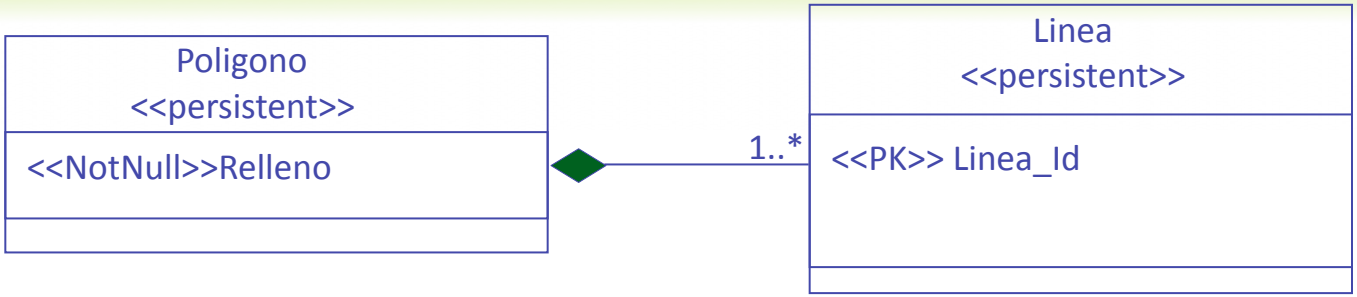


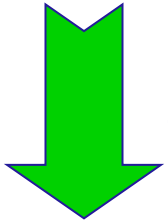
SQL:2008

```
CREATE TABLE Proyecto
( Proyecto_Id     INTEGER NOT NULL PRIMARY KEY,
  Nombre          VARCHAR2(30));

CREATE TABLE Plano
( Plano_id        INTEGER PRIMARY KEY,
   Proyecto_id    INTEGER NOT NULL FOREIGN KEY
                          REFERENCES Proyecto (Proyecto_id);
```

Reglas de transformación: Composición





SQL:2008

```
CREATE TABLE Poligono
( Figura_ID   INTEGER NOT NULL PRIMARY KEY,
  Relleno      BOOLEAN NOT NULL);

 CREATE TABLE Linea
(Linea_ID      INTEGER NOT NULL PRIMARY KEY,
 Figura_id     INTEGER NOT NULL
      FOREIGN KEY REFERENCES Poligono (Figura_Id)
            ON DELETE CASCADE ON UPDATE CASCADE);
```