

# Diseño de Bases de Datos

---

## Modelo Relacional

# Origen e importancia

» sencillo y potente: BD como conjunto de tablas con filas y columnas; las tablas están asociadas

## ■ Origen en los 70

- Edgar F. Codd [[CACM 1970](#)]: definición y consultas
- Substituye a modelos más antiguos: jerárquico, en red

## ■ Modelo más ampliamente utilizado

- Sencilla representación de datos
- Facilidad para formular consultas

## ■ El mundo de los RBDMS

- Oracle, MySQL (Oracle), Postgres, SQL Server (Microsoft), Informix, DB2 (IBM), Sybase (SAP), ...
- [[map](#)]

...

- Modelos competidores

- Modelo objeto-relacional
- Renovación de 'viejas ideas': datos jerárquicos

- Lenguaje relacional

- definición (DDL) y manipulación (DML) de datos
- el modelo relacional soporta consultas sencillas y potentes  
IMPORTANTE: *semántica* precisa de consulta relacional
- el DBMS optimiza la operación en términos de eficiencia
- SQL-92 (revisión importante)  
SQL-99 (extensiones importantes; estándar actual)

# Definiciones

- **BD Relacional: conjunto de relaciones**
- **Relación: dos partes**
  - **Esquema: nombre de la relación, y nombre y tipo de cada columna (atributo, campo)**  
Estudiante (nia:int,nombre:string,login:string,edad:int,notam:real)
  - **Instancia: valores de tabla (registro, tupla); todas las tuplas son distintas**

nia	nombre	login	edad	notam
2354	García	garcia@med	21	7,2
9625	Aragón	aragon@eii	18	6,7
5557	Pozo	pozo@inf	23	8,9

# Creación de relaciones

- Creación de la relación/tabla, con sus atributos/campos
- Los tipos/dominios se especifican para cada campo
- El **tipado es asegurado** por el **DBMS** cuando se añaden o modifican tuplas/registros
- Es **habitual la existencia de diversas tablas** que, **eventualmente, estarán relacionadas**

```
CREATE TABLE Estudiante (  
    ni a      INTEGER,  
    nombre   CHAR(20),  
    logi n   CHAR(10),  
    edad     INTEGER,  
    notam    REAL  
)
```

```
CREATE TABLE Matricula (  
    ni a      INTEGER,  
    cod      CHAR(20),  
    nota     REAL  
)
```

# Destrucción/modificación

- La destrucción de una relación implica su borrado del esquema, y el borrado de todas sus tuplas

```
DROP TABLE Estudiante
```

- La modificación de un esquema se puede hacer añadiendo, borrando o modificando campos
- Para campos añadidos, todas las tuplas son extendidas con valor *null* es ese campo

```
ALTER TABLE Estudiante  
ADD COLUMN matr DATE
```

# Añadir, borrar y modificar tuplas

- Se puede insertar una tupla en la relación

```
INSERT INTO Estudiante (ni a, nombre, logi n, edad, notam)  
VALUES (5557, ' Roj o', ' roj o@ci e', 23, 8. 0)
```

- El borrado se realiza indicando la condición que cumplirán todas las tuplas a ser borradas

```
DELETE  
FROM Estudiante E  
WHERE  
E. nombre=' Ledesma'
```

- La modificación se realiza igualmente con una condición de selección de tuplas

```
UPDATE Estudiante E  
SET E. edad=E. edad+1  
WHERE E. ni a=5557
```

# Restricciones de integridad

- **RI = condición de validez**
  - Condiciones que tienen que cumplirse para cualquier instancia de la base de datos
  - Se especifican cuando se define el esquema
  - Se comprueban cuando se modifican las relaciones (DBMS)
- Instancia legal = satisface todas las RI
- Semántica del mundo real descrito
- Dominio, clave primaria, clave foránea
- También se soportan otras más generales



# Claves primarias y candidatas

- Clave (candidata)
  - Identificación única de cada tupla, por medio de un subconjunto mínimo de campos
- Condiciones
  - No existen dos tuplas con los mismos valores en todos los campos de la clave —implicación sobre los *null*
  - Ningún subconjunto de la clave es identificador único —superclave: {nia, nombre}
- Una candidata debe ser elegida como clave primaria (o principal)
  - ¿Quién hace esta elección?

Estudiante (nia, nombre, login, edad, notam)

...

- Las claves candidatas se especifican con **UNIQUE**, y la que es elegida como primaria con **PRIMARY KEY**

```
CREATE TABLE Matricula (  
    ni a    INTEGER,  
    cod     CHAR(20),  
    nota    REAL,  
    PRIMARY KEY (ni a, cod)  
)
```

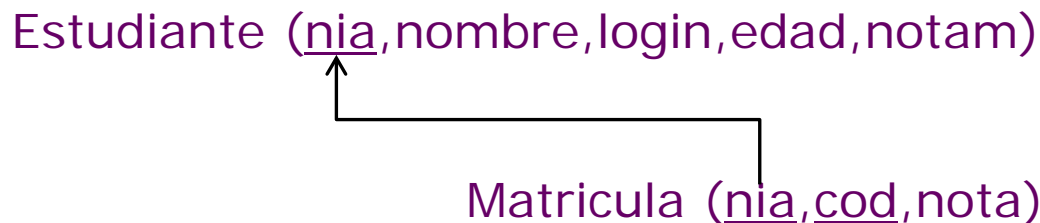
```
CREATE TABLE Estudiante (  
    ni a    INTEGER,  
    nombre  CHAR(20),  
    logi n  CHAR(10),  
    edad    INTEGER,  
    notam   REAL,  
    UNIQUE (nombre, edad),  
    CONSTRAINT claveEst PRIMARY KEY (ni a)  
)
```

nombrado de la restricción, lo que permite identificar errores

# Claves foráneas; integridad ref.

## ■ Clave foránea (o externa)

- Conjunto de campos en una relación que sirven para *referenciar* tuplas en otra relación —puntero de asociación
- La referenciada debe ser clave primaria, para asegurar que se hace referencia a una única tupla



## ■ Integridad referencial

- Imposición de las restricciones referenciales: sólo los estudiantes listados en Estudiante son admitidos en la Matrícula de cursos

...

```
CREATE TABLE Matricula (  
    ni a    INTEGER,  
    cod     CHAR(20),  
    nota    REAL,  
    PRIMARY KEY (ni a, cod),  
    FOREIGN KEY (ni a) REFERENCES Estudiante  
)
```

- Una clave foránea puede hacer referencia a la misma relación en la que se encuentra
- El *null* —desconocido/no-aplicable— cumple la restricción de clave foránea, pero no la de clave primaria

# Cumplimiento de la integridad ref.

- **Inserción** con referencia no existente
  - **NO ACTION**: rechazar la inserción de una matrícula con *nia* no existente en la tabla de estudiantes
- **Borrado** de una tupla referenciada
  - (default) **NO ACTION**: rechazar el borrado de un estudiante que tiene entradas en matrícula
  - **CASCADE**: borrar el estudiante, y todas las matrículas que le referencian
  - **SET DEFAULT**: asignar esas matrículas a un *nia* por defecto
  - (en SQL) **SET NULL**: asignar el *nia* de esas matrículas al valor especial *null*—conflicto con PK
- **Actualización de la clave primaria**
  - Mismo funcionamiento

...

```
CREATE TABLE Matricula (  
    ni a    INTEGER,  
    cod     CHAR(20),  
    nota    REAL,  
    PRIMARY KEY (ni a, cod),  
    FOREIGN KEY (ni a) REFERENCES Estudiante  
        ON DELETE CASCADE  
        ON UPDATE SET DEFAULT  
)
```

- La acción 'cascade' se propaga *en cascada* por siguientes relaciones que referencian a la tupla borrada o actualizada
- ¿Se puede utilizar en una relación que se autoreferencia?

# Otras restricciones


## ■ Restricciones de columna

- **NOT NULL**: no admite nulos
- **CHECK**: condición de integridad

```
CREATE TABLE Estudiante (  
    ...  
    notam REAL not null,  
    ...  
    CONSTRAINT notamos CHECK (notam>=0)  
)
```

## ■ Aserciones

## ■ Disparadores



necesitaremos conocer  
más sobre la potencia de  
las consultas en SQL

# Tipos de datos

<predefined type> ::=

<character string type> [...]

| <national character string type>

| <binary large object string type>

| <bit string type>

| <numeric type>

| <boolean type>

| <datetime type>

| <interval type>

SQL-99

SQL-99

<domain definition> ::=

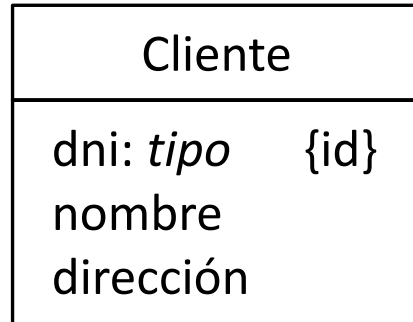
CREATE DOMAIN <domain name> [ AS ] <data type> [...]  
[<domain constraint>] [...]

```
CREATE DOMAIN color AS VARCHAR(10)
CHECK (VALUE IN ('rojo', 'verde', 'azul'));
```



# Diseño lógico: ER -> Relacional

- ERD



- ER

Cliente (dni, nombre, dirección)

- SQL

```
CREATE TABLE Cliente (  
    dni          CHAR(10),  
    nombre       CHAR(20),  
    dirección    CHAR(50),  
    PRIMARY KEY (dni))
```

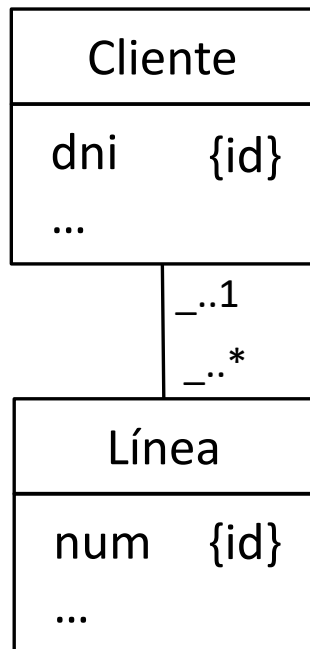
# Tipos de relación -> tablas

## ■ Cardinalidad máxima: 1—1

- Comprobamos si se pueden poner en términos de una única entidad

## ■ 1—\*

- Referenciamos en el lado \*, el valor (único) del lado 1



Cliente (dni,...)

Línea (num,dniC,...)

```
CREATE TABLE Cliente (  
    dni      CHAR(10), ...,  
    PRIMARY KEY (dni))
```

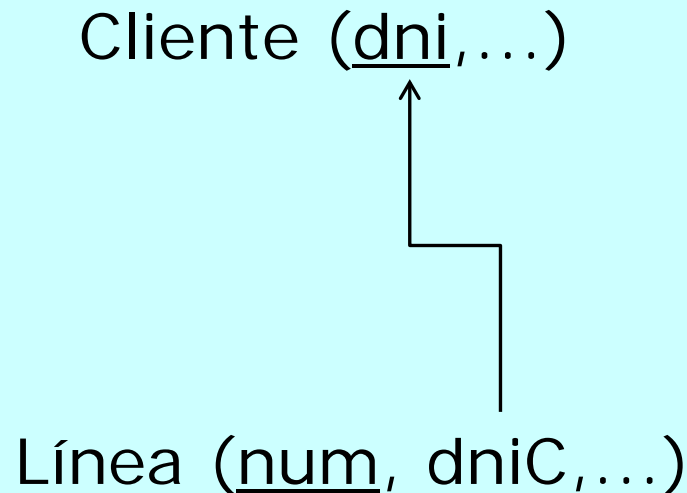
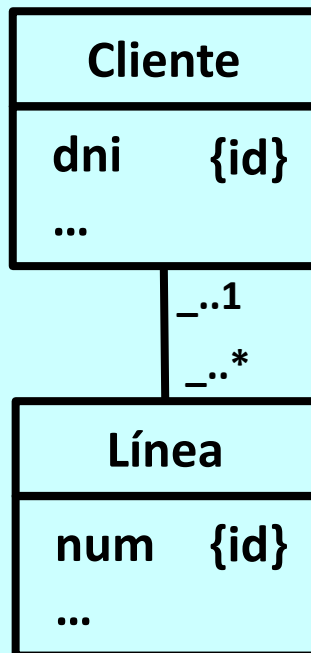
```
CREATE TABLE Línea (  
    num      INTEGER,  
    dniC     CHAR(10), ...,  
    PRIMARY KEY (num)  
    FOREIGN KEY (dniC)  
        REFERENCES Cliente)
```

# CLAVE FORÁNEA EN RELACIÓN 1—\*

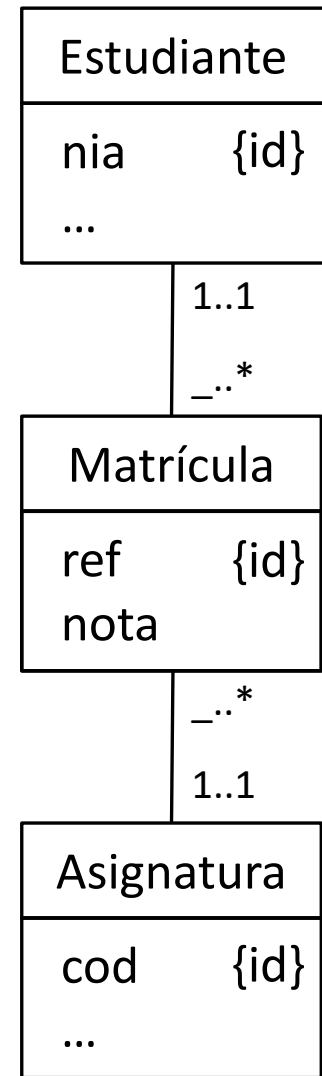
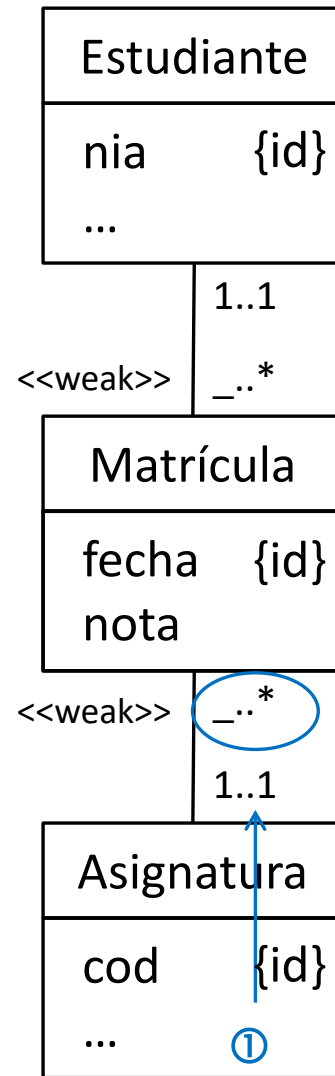
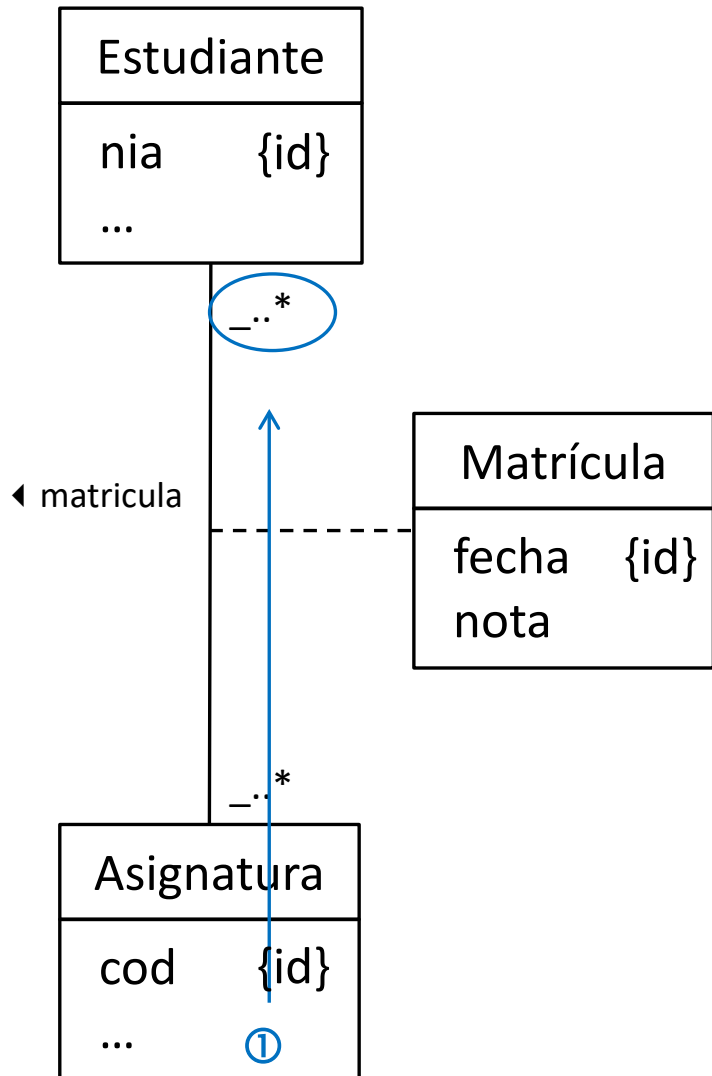


Essentials

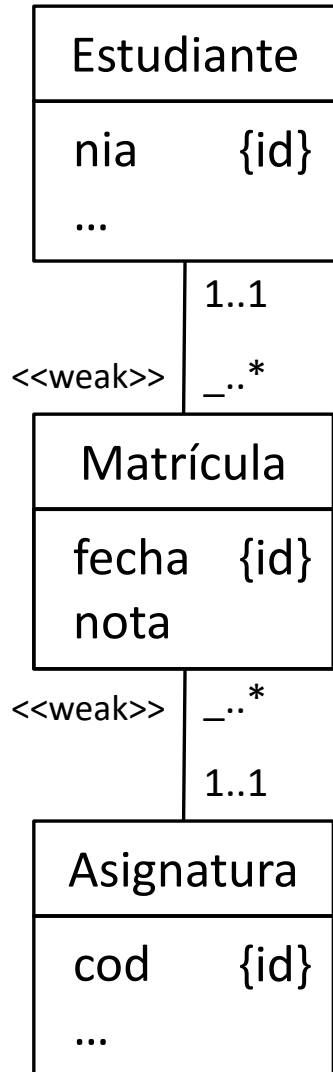
- En el lado \*,  
apuntando al lado 1 (valor único)



\* \_\_\_\_ \*



...



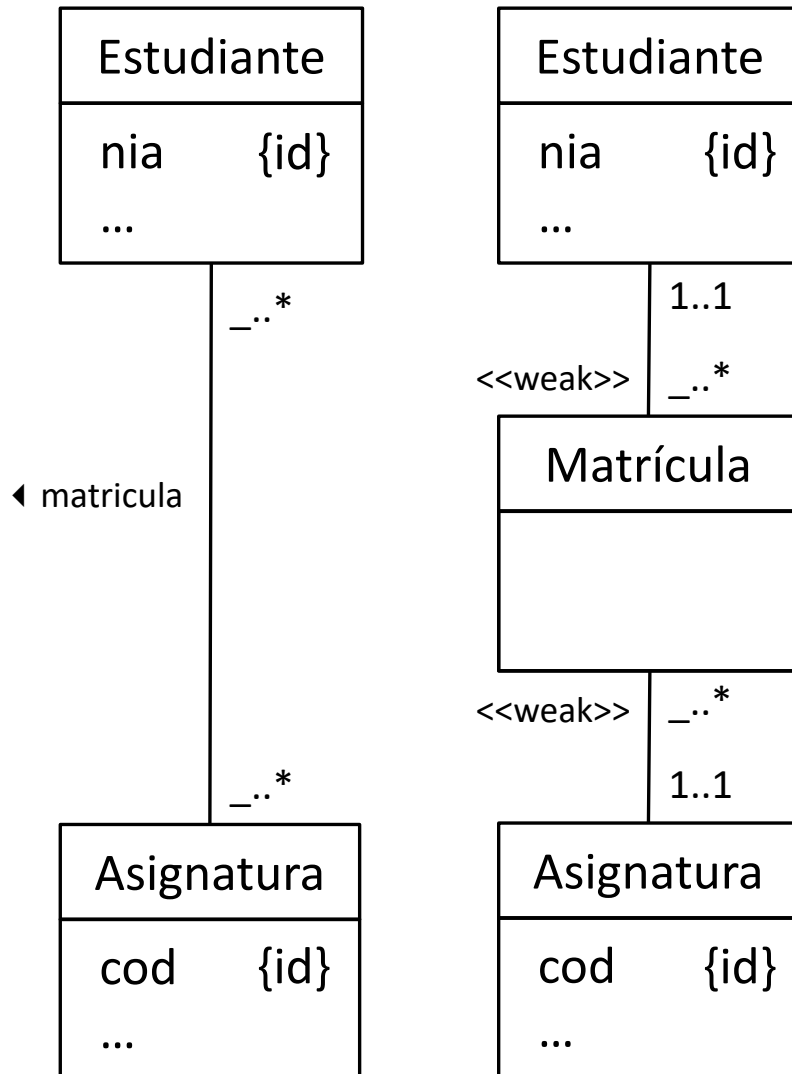
Estudiante (nia,...)

Matrícula (niaE,codA,fecha,nota)

Asignatura (cod,...)

```
CREATE TABLE Estudiante (...)  
CREATE TABLE Asignatura (...)  
CREATE TABLE Matrícula (  
    niaE    INTEGER,  
    codA    INTEGER,  
    fecha   DATE,  
    nota    REAL,  
    PRIMARY KEY (niaE, codA, fecha),  
    FOREIGN KEY (niaE) REFERENCES Estudiante,  
    FOREIGN KEY (codA) REFERENCES Asignatura)
```

...



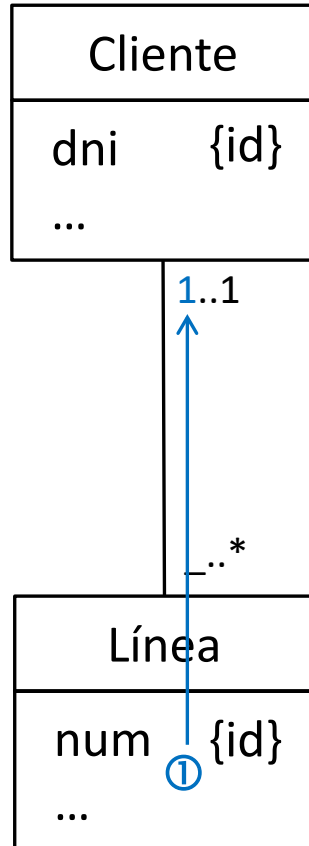
Estudiante (nia,...)

Matrícula (niaE,codA)

Asignatura (cod,...)

```
CREATE TABLE Estudiante (... )
CREATE TABLE Asignatura (... )
CREATE TABLE Matricula (
    niaE    INTEGER,
    codA    INTEGER,
    PRIMARY KEY (niaE, codA),
    FOREIGN KEY (niaE)
        REFERENCES Estudiante,
    FOREIGN KEY (codA)
        REFERENCES Asignatura)
```

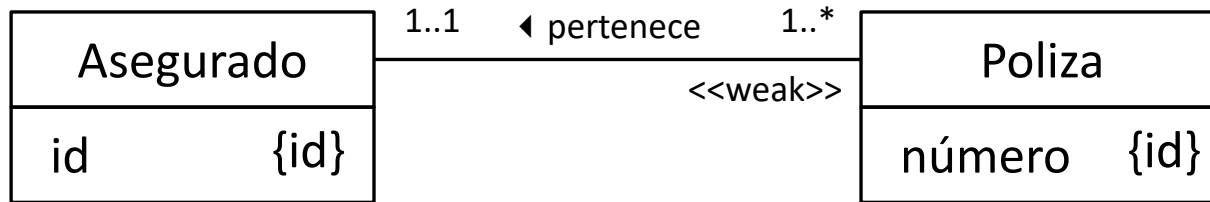
# Traducción de la opcionalidad



```
CREATE TABLE Línea (  
    num          INTEGER,  
    dni C        CHAR(10) not null,  
    . . . ,  
    PRIMARY KEY (num),  
    FOREIGN KEY (dni C)  
        REFERENCES Cliente)  
    ON DELETE NO ACTION)
```

- la opcionalidad **0..\_** supone la existencia de nulos (*default*)
- ¿representación explícita en el modelo relacional?
- la obligatoriedad cliente — (**1..\_**) línea debe tratarse con restricciones **CHECK**

# Entidades débiles



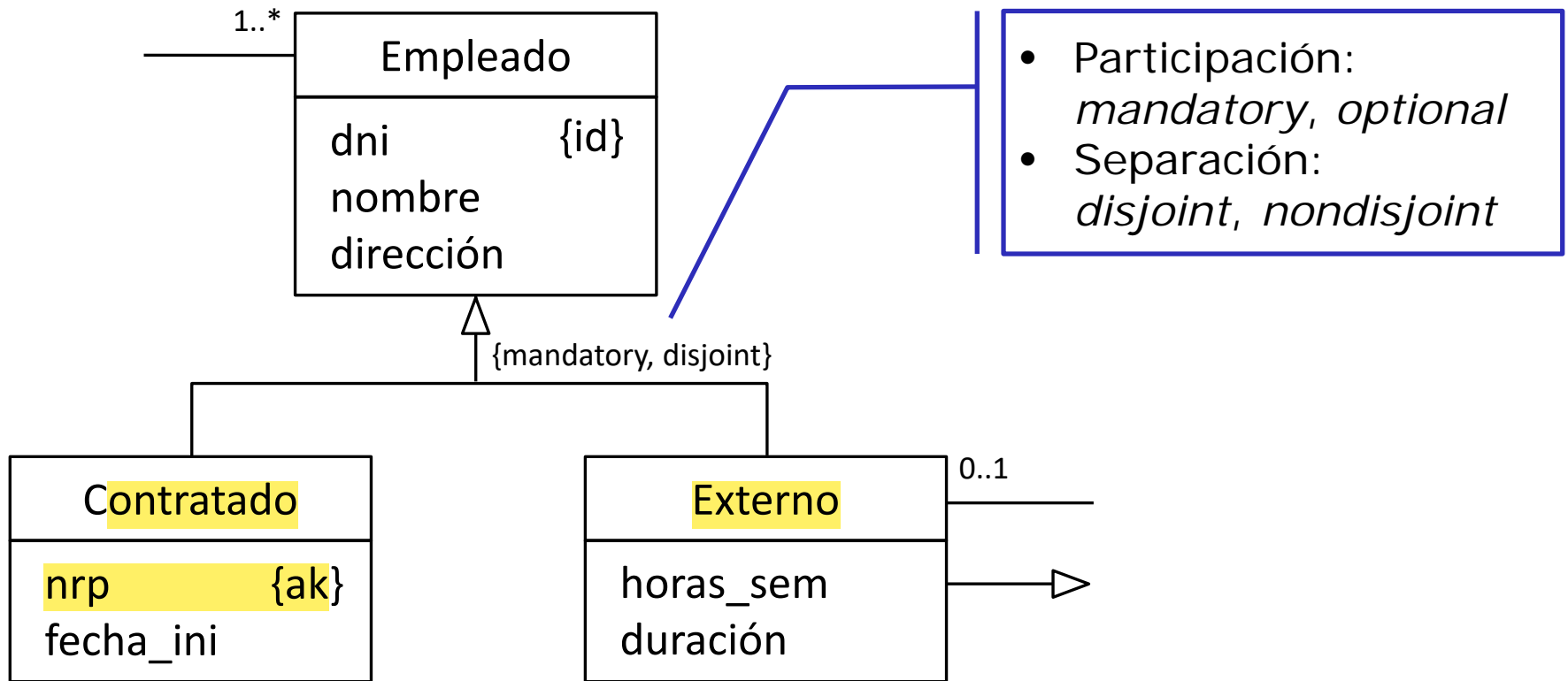
- combinación de: (1) identificación con campos de otra entidad, (2) relación obligatoria
- FK no nula que forma parte de la PK

Asegurado (id,...)  
↑  
Póliza (num, idA,...)

```
CREATE TABLE Asegurado (...)  
CREATE TABLE Poliza (  
    num        INTEGER,  
    idA        CHAR(10) not null,  
    ... ,  
    PRIMARY KEY (num, idA),  
    FOREIGN KEY (idA)  
        REFERENCES Asegurado  
        ON DELETE CASCADE)
```



# Jerarquías ISA



# ISA, opciones

## ■ 1 tabla

Empleado (dni, nombre, dirección, nrp, fechaIni, horasSem, duracion, FK1)

- ¿cómo distingo a los contratados de los externos?
- ¿qué pongo en nrp cuando introduzco un externo?

## ■ 2 tablas

Empleado\_Cont (dni, nombre, dirección, nrp, fechaIni, FK1)

Empleado\_Exte (dni, nombre, dirección, horasSem, duración, FK1)

- ¿es un problema la repetición de lo común?
- ¿qué ocurre si llega una relación \*—1 a empleado?
- ¿y si quiero saber el número total de empleados?

# ISA, opciones

- 3 tablas

Empleado (dni, nombre, dirección, FK1)

Empleado\_Cont (dni, nrp, fechaIni)

Empleado\_Exte (dni, horasSem, duración)

- ¿es un problema la repetición del *dni*?
- ¿qué ocurre si quiero saber el nombre de un *nrp*?

- 2\* tablas

Empleado (dni, nombre, dirección, nrp, fechaIni, FK1)

Empleado\_Exte (dni, horasSem, duración)

- ¿cuándo sería esto adecuado?

...

» es una guía de elección, pero puede haber otros factores (cantidad de nulos y de tuplas, relaciones, consultas / combinación de tablas)

## ■ Tabla para la superclase

- si tiene asociaciones **1:\*** (lado 1); las tablas que incluyen las claves foráneas deben apuntar a una única tabla
- si es **optional**, para los que solo pertenecen a la superclase
- si es **nondisjoint**, para evitar repeticiones
- discriminador: explícito o implícito

## ■ Tablas para las subclases

- diferencias entre ellas; todas en una tabla daría lugar a nulos y restricciones
- diferencia con la superclase; puede haber tablas para cada combinación superclase/subclase

# TABLAS PARA ISA



Essentials

---

- Tabla para la superclase
  - si es apuntada por clave foránea
  - si es optional y/o nondisjoint
- Tablas para las subclases
  - si hay diferencias que implicarían *demasiados* nulos y restricciones

# Vistas y seguridad

## ■ Definición de la vista

- *Tabla* que no almacena tuplas, las cuales se obtienen a partir de una *definición* sobre tablas base
- Permite reestructurar el esquema lógico base, e implementar políticas de acceso restringido a los datos

```
CREATE VIEW Estudi anteMH (ni a, notam)  
AS SELECT E. ni a, E. notam  
FROM Estudi antes E  
WHERE E. notam >= 9.0
```

dando acceso a través de la vista, no se pueden ver los nombres, login o edades

no se puede acceder al resto de alumnos

```
CREATE VIEW Persona AS  
SELECT * FROM Persona_Fi si ca  
UNION  
SELECT * FROM Persona_Juri di ca
```

# ¿Qué nos falta?

- Principalmente, la manipulación y consulta de datos de una BD
  - Necesitaremos lenguajes de consulta (QL)
  - QLs
    - “*reales*” para la implementación (SQL)
    - formales para la conceptualización (*RA, RC*)
  - ¡ Primero los conceptos !
    - *fundamentar* lo que significa manipular relaciones para obtener los resultados de las consultas
  - ¡ Después las implementaciones !