Teo: Pasaje a Tablas

☑ Archive ☐

Introducción

Una vez tenemos un buen modelo ER, podemos aprovechar ese modelo haciendo una transición del mismo a un modelo relacional formado por **tablas** (visto en introducción a BD) para el almacenamiento de los datos y su gestión

Modelo Relacional

Conceptos Básicos

Esquema Relacional

En este tipo de modelos tenemos un **esquema relacional** que especifica los atributos de una tabla y cómo se relaciona con otras.

Este esquema es una **representación abstracta** de la base de datos que proporciona una vista de alto nivel de cómo se deben estructurar los datos, no contiene los datos reales que se van a contener.

Ejemplo: Tenemos que una entidad Universidad tiene Instructores donde cada instructor tiene un identificado, nombre, nombre de departamento y salario. Es decir:

- Entidad: Instructor (R)
- Atributos: ID (A_1), name (A_2), dept_name (A_3), salary (A_4)

En un esquema relacional, representamos a R = (A_1, A_2, A_3, A_4) tal que

Instructor = (ID, name, dept_name, salary).

Notación: nombres de esquema relacional comienzan con mayúscula.

Relación o Instancia Relacional

Las relaciones se refieren a las tablas que almacenan los datos organizados en filas y columnas. Cada fila de la tabla representa un registro o tupla individual, y cada columna representa un atributo o campo de datos específicos.

Las relaciones son estructuras de datos reales que contienen un conjunto de tuplas y valores de atributos. Solemos definirlas como el producto cartesiano del conjunto de las columnas de una tabla:

$$D_1 \times D_2 \times \ldots \times D_{n-1} \times D_n$$

Ejemplo: Tenemos una tabla Persona con 3 columnas: nombre, edad, DNI. La relación sería:

- Relación: persona $\subseteq$ string x integer x integer
- Tupla de persona: (“Jorge Pérez”, 51, 18003567)

Notación: nombres de relaciones comienzan con minúscula,

💡 **Relación: ER y en Modelo Relacional**
- En el modelo relacional, las relaciones se refieren a **tablas** que almacenan datos organizados en filas y columnas.
- En el modelo ER se refiere a una **conexión semántica** entre entidades.
Ambos conceptos no tienen nada que ver.

Unión de conceptos

Ahora, esquema relacional y relación nos sirve para definir **una relación r con esquema de relación R**

- $r(R)$ = r es una relación con esquema de relación R

Ejemplo: tenemos persona(Persona) donde Persona = (nombre, edad, DNI) tal que persona(nombre, edad, DNI).

Esta unión es importante porque:

- Los conceptos asociados a **esquemas de relación** ahora se asocian también a **relaciones** o **tablas de la BD**, por ejemplo: atributos y dominios, claves primarias y foráneas.
- Los dominios de los atributos en los esquemas relacionales pasan a ser los dominios las tablas. Esto quiere decir que los dominios de los atributos en el esquema establecen las restricciones y los tipos de datos permitidos para los valores almacenados en las tablas.

  Ejemplo: tenemos un esquema Persona con atributos (Nombre: string, Edad: int, DNI: int). Si creamos la relación persona tal que persona(Persona), entonces, en nuestra tabla de la BD, los atributos Nombre, Edad y DNI tendrán el mismo tipo de datos que en esquema.
- Simplifica la comprensión de una tabla ya que las columnas tienen los nombres de un esquema de relación.

Tipos de Atributos

El conjunto de valores permitidos para cada atributo se llama **dominio del atributo**. Este conjunto de valores es atómico: representan valores indivisibles

Que representen valores indivisibles quiere decir que un atributo no puede ser unión de información que podría estar en atributos diferentes.. Si tenemos un atributo “nombre”, guardaremos allí nombre y, apellido si es necesario, en otro atributo. No tendremos en el atributo “nombre” algo como “Juan Bratti”, ya que dificulta la realización de consultas.

Básicamente, **no tener atributos con información compuesta**.

Además de mantener atomicidad, también debemos:

- no tener atributos multivalorados
- no tener atributos compuestos

Si queremos representar que un atributo no tiene valor, usamos el valor especial **null** que es un miembro de **todo dominio**.

Terminología hasta ahora

Informal Terms		Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row fila		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Bases de Datos Relacionales

Aquí tenemos dos conceptos:

- Esquema de una BD relacional: conjunto de esquemas de relación R_1, ..., R_n. Cada esquema (como ya vimos) definirá la estructura lógica y atributos de una

tabla en la base de datos.

- Una base de datos relacional consiste de múltiples relaciones $r_1(R_1)$, ..., $r_n(R_n)$. Cada relación (como ya vimos) representará una tabla en la base de datos.

Conceptos Básicos

Flujo de Desarrollo e Interacción de una BD Relacional

1. Definimos el esquema de relacion de la base de datos.
2. Creamos las bases de datos relacionales (o tablas)
3. Poblamos las tablas con datos reales o iniciales insertando registros o tuplas.
4. Consultamos y manipulamos los datos a gusto.

Nomenclaturas

- Nombre de relación: en minúsculas $r, s, u, r_1, r_2, \dots$.
- Nombre de esquema de relación: en mayúsculas $R, S, U, R_1, R_2, \dots$.
- Notación t[A]: : Sea t ∈ r , r(R), A ∈ R . t[A] es el valor de t en A (o también, t[A] representa el valor de un atributo específico A en la tupla t)
- Notación t[R]: Sea t ∈ r , r(R). t[i] es el valor de t en atributo iésimo de R.

Superclaves

En el modelo relacional, una superclave es un conjunto de uno o más atributos de un esquema de relación que tiene la propiedad de identificar de manera única cada tupla (fila) en esa relación. Esto significa (como vimos varias veces), dada una superclave, no puede haber dos tuplas en la relación que tengan los mismos valores para todos los atributos de la superclave.

Usando al terminología y notación vista hasta ahora, podríamos decir que:

> **Superclaves**: Sea K ⊆ R , R esquema de relación; K es una superclave de R si los valores para K son suficientes para identificar una tupla única en cada posible relación r(R).

Claves Candidatas

En específico, una clave candidata es un conjunto de atributos minimal, tal que no pueda eliminarse ningún atributo de la clave sin perder su capacidad de identificar de manera única las tuplas.

Clave Primaria

De entre las claves candidatas, se elige una para ser la clave primaria de la relación. Esta clave es la que se usará como principal identificador único de las tuplas en la relación.

Clave Foránea

Tenemos una clave foránea cuando hay un atributo o conjunto de atributos en una relación (tabla) que hace referencia a la clave primaria de otra relación. En otras palabras, es un atributo que establece una relación entre dos tablas en la base de datos.

- Relación Referenciante: la relación que contiene la clave foránea se denomina “relación referenciante”.
- Relación Referenciada: la relación cuya clave primaria es referenciada por la clave foránea se denomina “relación referenciada”

Los valores de la clave foránea deben coincidir con los valores en al clave primaria de la relación referenciada.

Ejemplo:

Tenemos dos relaciones:

- instructor(ID, name, dept_name, salary)
- department(dept_name, building, budget)

El valor de dept_name en instructor debe aparecer en departement. Además, instructor es la relación referenciante y department es la relación referenciada.

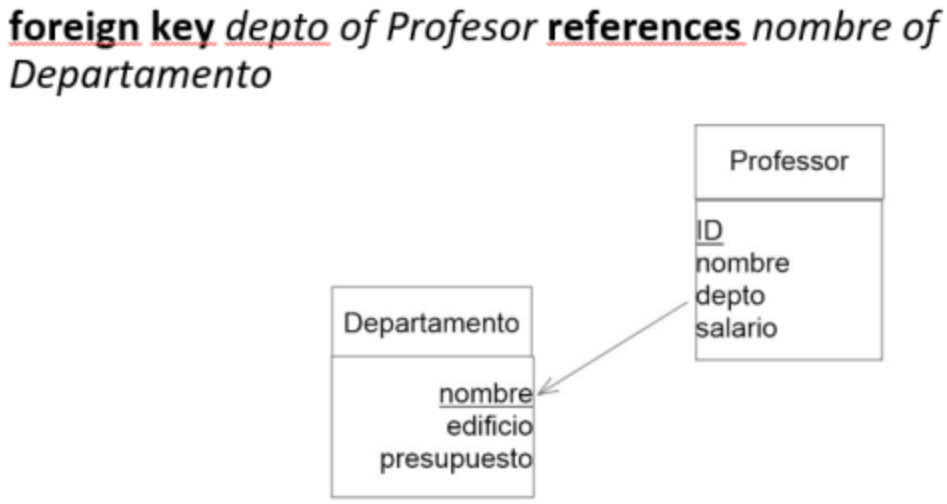
Ejercicio:

Tenemos que takes tiene 5 claves:

1. ID: es una clave foránea de ID de student.

2. course_id: es una clave foránea de section
3. sec_id: es una clave foránea de section
4. semester: es una clave foránea de section
5. year: es una clave foránea de section

Interpretación gráfica de FK



Diseño de BD Relacional

El diseño de una BD relacional es un proceso importante para garantizar que los datos se almacenen de manera eficiente y coherente. Para lograr un buen diseño de BD, nos centraremos en lograr un buen diseño de modelos ER para evitar problemas como:

- Almacenar toda la información en una sola relación.
    1. Esto repite información (dos estudiantes con el mismo instructor
    2. Hace que se necesiten valores nulos (representar un estudiante sin supervisor)

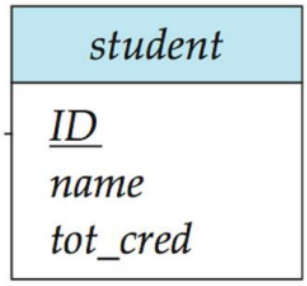
Entonces, si tenemos un buen diseño de un esquema de BD en un modelado ER, necesitaremos traducir este diagrama en un modelo relacional! Para esto, tenemos **reglas de traducción** que suelen resolver un porcentaje alto del problema (aunque las reglas no contemplan todos los casos).

Traducción ER → Relacional

La idea de representar entidades, relaciones y atributos del modelo ER consiste en tablas relacionales con claves primarias y restricciones de clave foránea en el modelo relacional. Algunas pautas son:

Reglas de Traducción Automáticas | Entidades

- **Regla CE1:** Una CE fuerte sin atributos multivalorados ni compuestos se traduce en una tabla donde los atributos se convierten en columnas y la clave primaria del CE se convierte en la de la tabla.



Se mapea a:

student(ID, name, tot_cred)

- **Regla CE2:** Un CE fuerte que tiene atributos o subatributos (porque podrían ser compuestos) no multivalorados, se traduce en una tabla con los mismos atributos simples y los subatributos hoja de los atributos compuestos

Se mapea a:

biblioteca(nombre, calle, número, ciudad)

- **Regla CE3:** Un CE fuerte que no tiene atributos compuestos pero si multivalorado se traduce en una tabla para la entidad donde los atributos no multivalorado son las

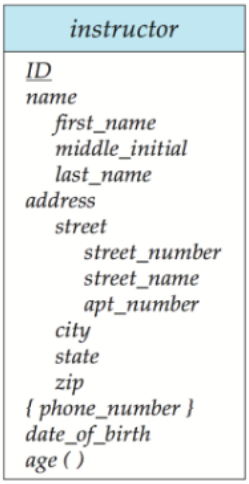
columnas, y otra tabla para la entidad donde la FK y el atributo multivalorado son las columnas, siendo la FK la PK de la primer tabla.

Se mapea a:

libro(título, ISBN, editorial, edición)
libro-autor(ISBN, autor)
For *libro-autor* **foreign key** *ISBN* **references** *libro*

- Atributos derivados: generalmente no se almacenan en columnas ya que sus valores se pueden calcular cuando sea necesario.

Ejercicio:

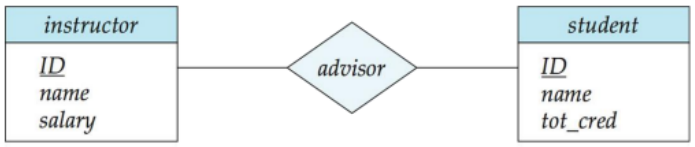


instructor(ID, first_name, middle_name, last_name, date_of_birth)
addresses(ID, street_number, street_name, apt_number, city, state, zip)
phones(ID, phone_number)
FOR phones FOREIGN KEY ID references instructor
FOR addresses FOREIGN KEY ID references instructor

omito age ya que es derivado y puede calcularse mediante consultas específicas

Reglas de Traducción Automáticas | Relaciones

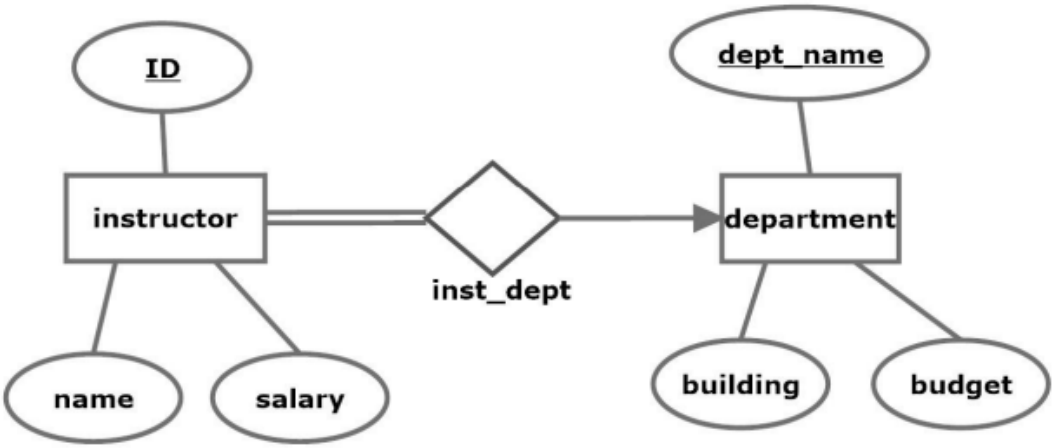
- **Regla CR1:** Un CR varios-varios se convierte en una tabla en donde se toman las PK de los CE como FK de la tabla de la CR. Si la CR tiene atributos, éstos se tomarán como columnas de la tabla de la CR. Las CE tendrán sus tablas aparte.



Se mapea a:

Advisor = (i-id, s-id)
For *Advisor* **foreign key** *i-id* **references** *instructor*
For *Advisor* **foreign key** *s-id* **references** *student*

- **Regla CR2:** Un CR uno-varios o varios-uno que es **total** en lado varios se traduce dos tablas (uno para cada CE) pero en el CE del lado varios, se agrega la clave primaria del lado uno (como FK).

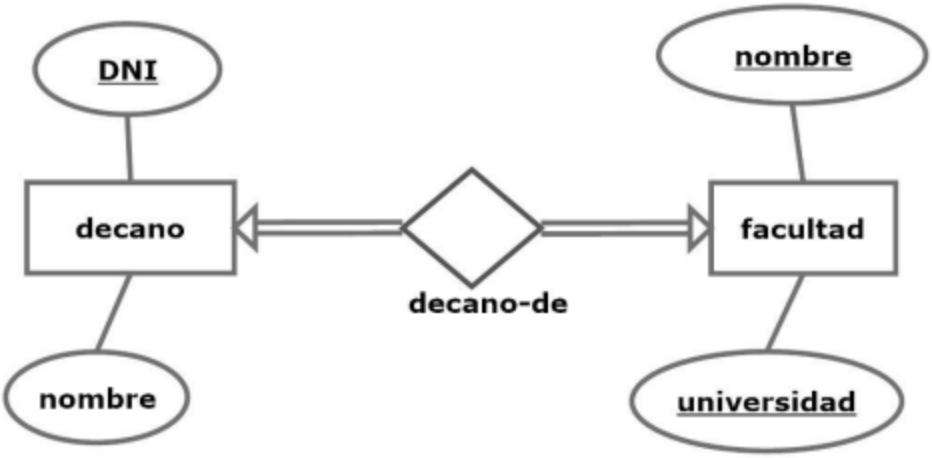


CR inst_dept se mapea a:

instructor(ID, name, salary, dept_name)
For *instructor* **foreign key** *dept_name* **references** *department*

Si la participación no es total en el lado varios, puede resultar en valores nulos.

- **Regla CR3:** Un CR uno-uno con participación total en ambos lados se traduce en una tabla de uno de los CE sumando los atributos PK como FK del otro CE (que tiene su otra tabla).



El CR decano_de se mapea a:

Las siguientes respuestas son igualmente válidas:

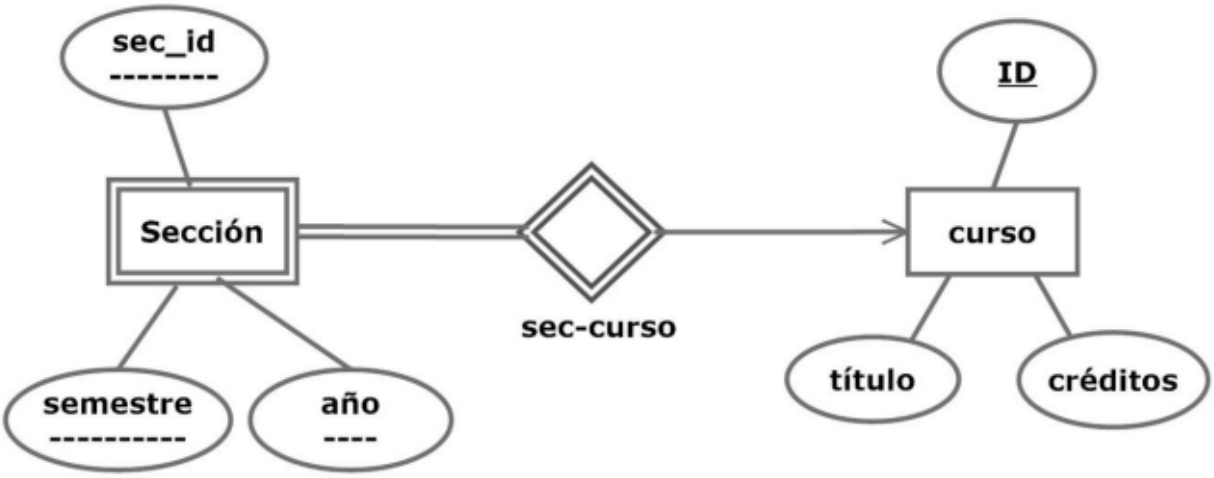
decano(DNI, nombre, nombreFacultad, universidad)
For *decano* **foreign key** *nombreFacultad, universidad*
references *facultad*

y

facultad (nombre, universidad, DNI)
For *facultad* **foreign key** *DNI* **references** *decano*

Reglas de Traducción Automáticas | Entidades Débiles

- **Regla CED:** se traduce a una tabla que sus columnas contiene a la PK del CE fuerte/identificador como FK más los atributos (no multivalorados) del CE débil.

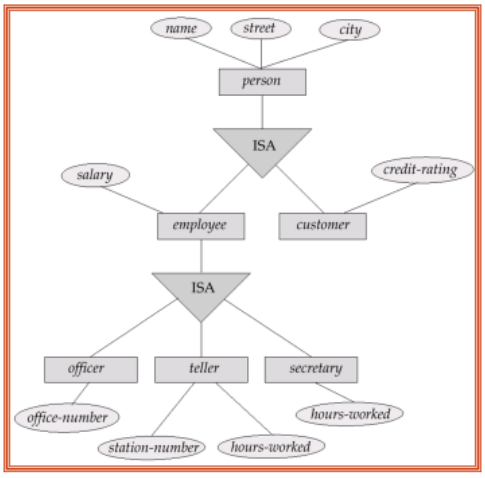


El CE débil sección se mapea a:

sección(ID, sec_id, semestre, año)
For *sección* **foreign key** *ID* **references** *curso*

Reglas de Traducción Automáticas | Generalizaciones

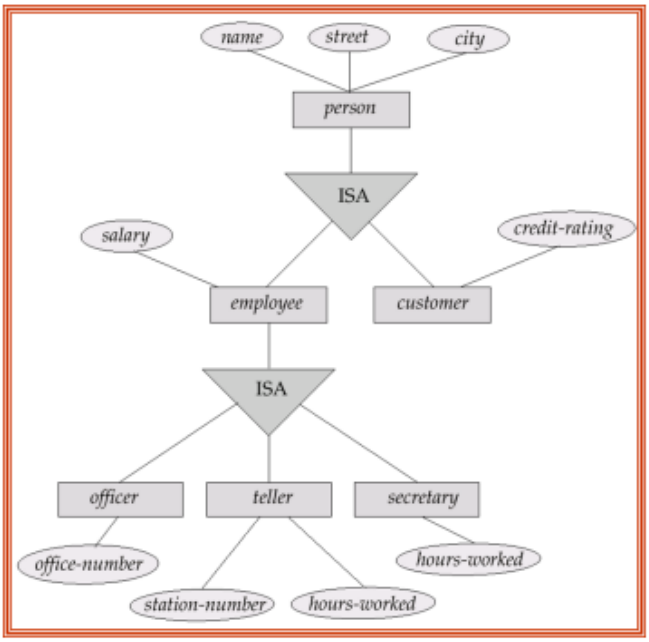
- **Idea G1:** se forma una tabla para el CE de más alto nivel y luego se hacen tablas para cada especialización del CE de alto nivel en donde se incluye: PK del CE de alto nivel + atributos locales de cada especialización.



Primera idea de mapeo de los CE que participan en generalización.

person(name, street, city)
customer(name, credit-rating)
Employee(name, salary)

- **Idea G2:** tener una tabla para cada CE (alto nivel y especialización) cada uno con los atributos locales y heredados del CE de más alto nivel. Esto es para evitar tener que revisar más de una tabla para obtener información. Así tendríamos las tablas con toda la info completa.



Usando la idea G2 los CE se mapean a:

person(<u>name</u>, street, city)
customer(<u>name</u>, street, city, credit-rating)
Employee(<u>name</u>, street, city, salary)

- **Idea G3:** para evitar redundancia cuando tenemos que una CE contiene a otra y la generalización es total y disjunta. Se traduce en una tabla para cada CE especialización con los atributos locales y heredados, no se hace la tabla para la generalización.

Resumen

- Generalización total y disjunta: Idea G3.
  - como no hay instancias huérfanas, no hace falta almacenar información de la CE de más alto nivel.
- Generalización no total y disjunta: Idea G2.
  - no hay redundancias
  - no hay que consultar dos tablas para obtener info de una especialización
- Generalización no disjunta: Idea G1.

- Tengo que acceder a dos tablas para conocer toda info de un CE de especialización.

> 💡 Puede haber pros y contras en todas las propuestas.

Ver ejemplos de traducción de Generalizaciones en modelos relacionales en las filminas.