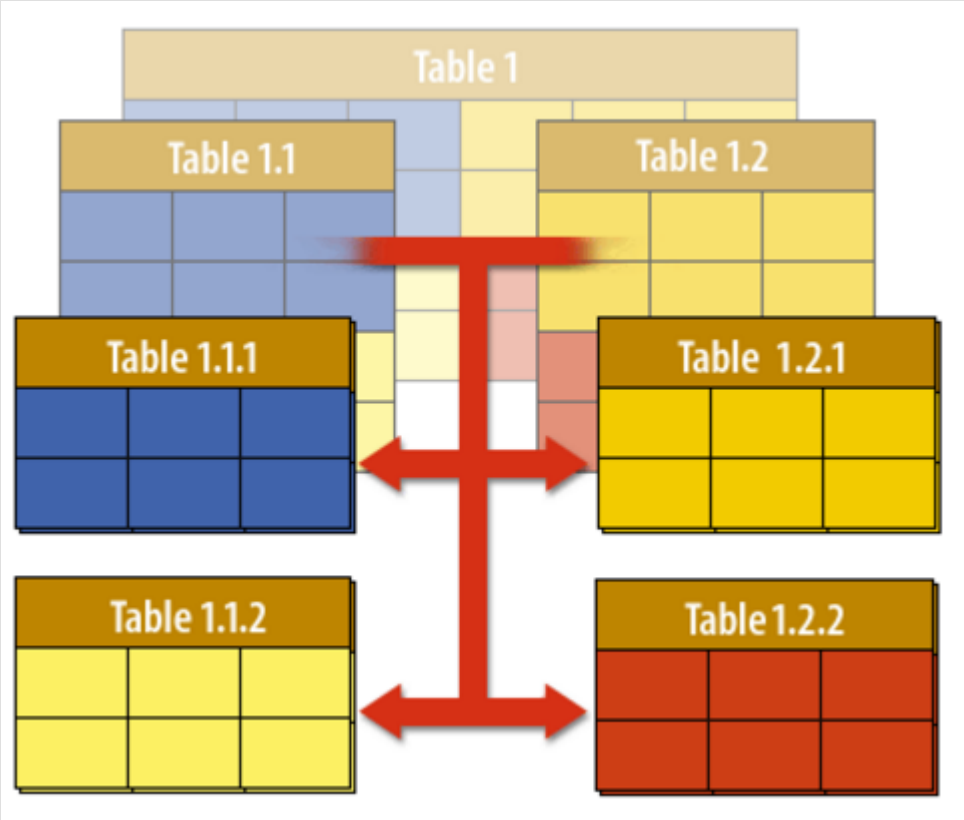


DigitalHouse > Coding School

DATA SCIENCE

SUBQUERIES
NORMALIZACIÓN

OBJETIVOS DE LA CLASE

1. **Subconsultas**
2. **DER - Normalización**
3. **Idear un modelo de datos para un problema concreto**

SUBQUERIES

- El lenguaje SQL es muy versátil y se pueden hacer cosas más allá de JOIN entre dos tablas diferentes.
- Un **SUBQUERY** o query interno o query anidado, es un query dentro de otro query SQL.
- Se pueden utilizar por ejemplo para establecer condiciones sobre el query principal y así restringir los resultados del query principal (externo), según los resultados del query secundario (interno).
- Un subquery puede ser usado dentro de sentencias SELECT, INSERT, UPDATE y DELETE, con operadores como =, <, >, >=, <=, IN, BETWEEN etc.

Sintaxis

Aquí hay un ejemplo de un subquery. La tabla resultante del subquery es usada como condición en el WHERE del query principal.

```
SELECT columna1
    FROM tabla1
    WHERE columna2 [Operador de comparación]
        (SELECT columna3
         FROM tabla2
         WHERE condición);
```

- Por ejemplo, extraigamos todas las órdenes de clientes de Francia.

```
SELECT "OrderID" FROM Orders
WHERE "CustomerID" IN
      (SELECT "CustomerID"
       FROM Customers
       WHERE "Country" = 'France');
```

OrderID
10248
10251
10265
10274
10295
10297
10311
10331

- Vean que en este caso es equivalente a hacer un JOIN de la siguiente forma.

```
SELECT "OrderID" FROM Orders
JOIN Customers ON Orders."CustomerID"=Customers."CustomerID"
WHERE Customers."Country" = 'France';
```

- Veamos otro ejemplo, aquí vamos a mostrar los 5 productos más baratos cuyos precios sean mayores a la media e imprimiremos para ellos su categoría, nombre y precio.

```
 SELECT "CategoryName","ProductName","UnitPrice"
FROM products AS p
JOIN categories AS c ON c."CategoryID" = p."CategoryID"
WHERE p."UnitPrice" > ( SELECT AVG("UnitPrice") FROM Products )
ORDER BY p."UnitPrice" ASC
LIMIT 5;
```

CategoryName	ProductName	UnitPrice
Produce	Uncle Bob's Organic Dried Pears	30
Seafood	Ikura	31
Confections	Gumbär Gummibärchen	31.23
Dairy Products	Mascarpone Fabioli	32
Meat/Poultry	Perth Pasties	32.8

PRÁCTICA GUIADA JOINS Y SUBQUERIES

MODELADO DE BASE DE DATOS

Modelado

Un modelo permite describir la **estructura lógica** de una base de datos.

En esta clase veremos **dos modelos:**

Modelo
Entidad Relación

Modelo
Relacional

Modelo **Entidad Relación**

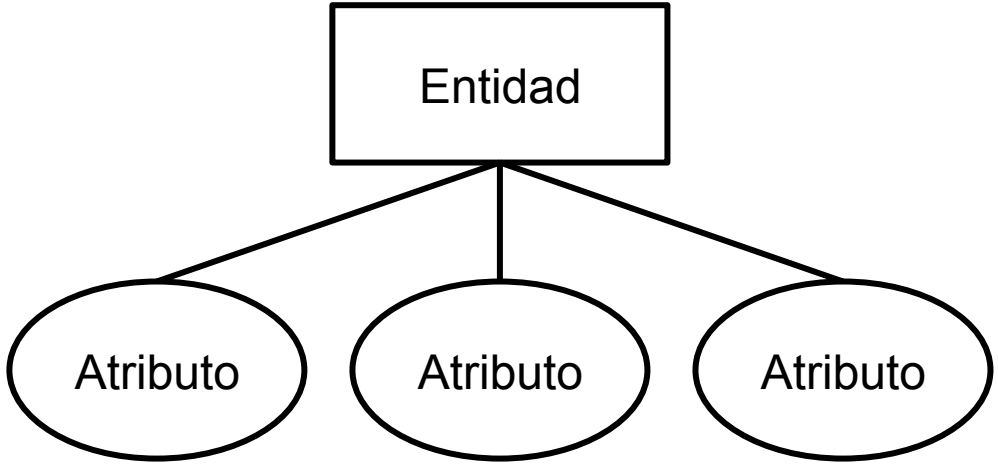
Lenguaje que describe de una base de datos las **entidades** que participan en el problema y las **relaciones** que existen entre ellas.

Entidad

Concepto del **mundo real** que se describe en una base de datos

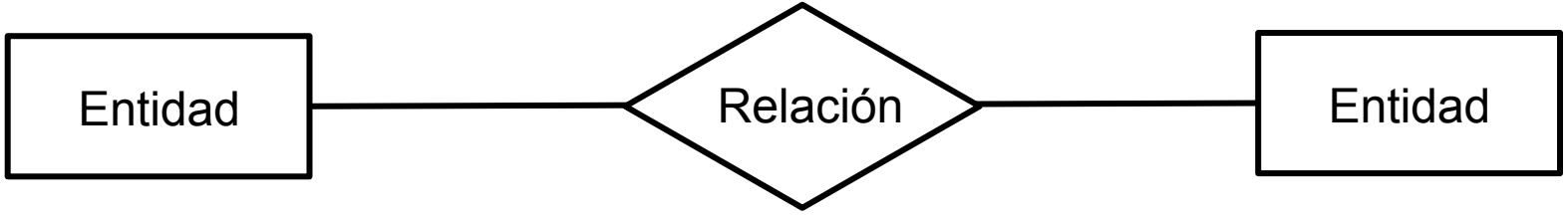
Entidad

- Pueden ser **concretas o abstractas.**
- Se representan con un **rectángulo**.
- Sus **atributos se representan con un círculo.**



Relación

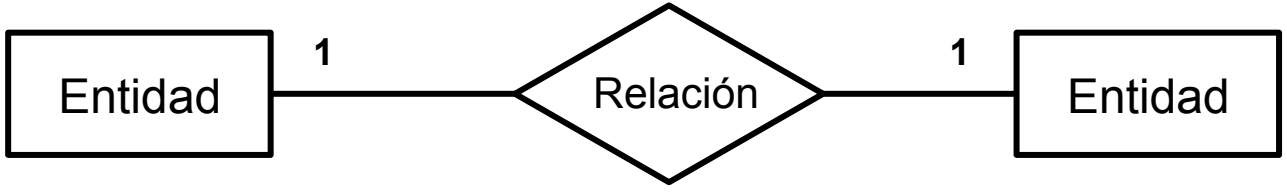
- **Conexión lógica** entre entidades.
- Pueden tener **atributos** propios.
- Se representa con un **rombo**.



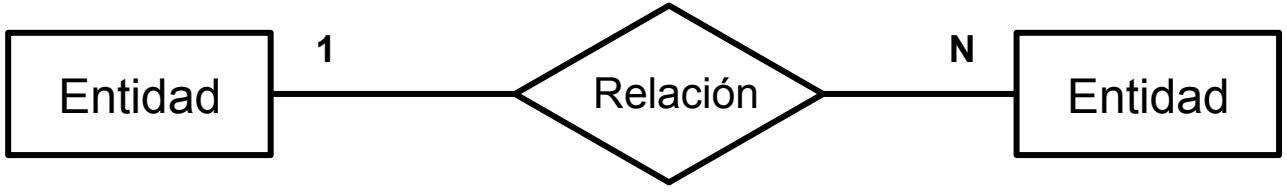
Cardinalidad

Número de entidades con la cual otra entidad puede asociarse mediante una relación.

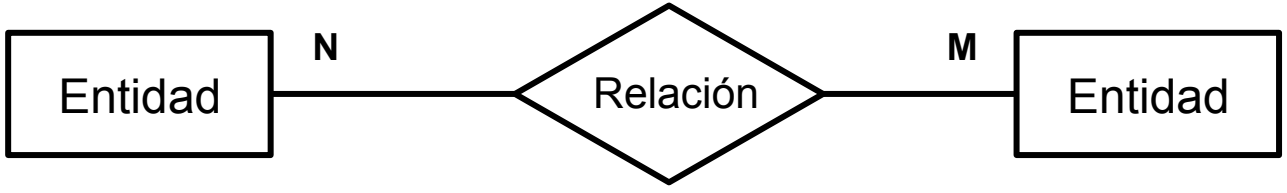
1:1



1:N



N:M



Metodología

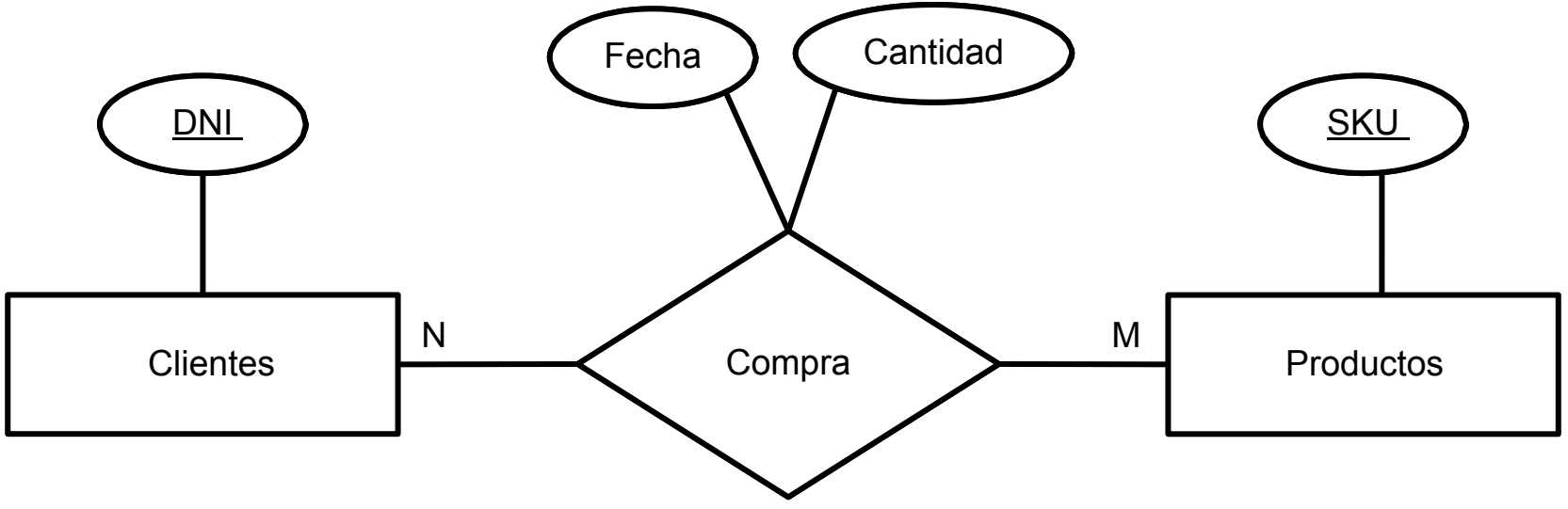
- **Identificar entidades:**
  - Definir objetos del mundo a representar.
- **Identificar atributos:**
  - Definir las “propiedades” de cada entidad.
- Determinar la **clave primaria** de cada entidad.
- Identificar **relaciones** entre las entidades.
- Señalar **cardinalidad** entre entidades.

Ejemplo

Se quiere representar la compra de **productos** por **clientes**.

- Un **cliente** tiene un **DNI.**
- Un **producto** tiene un **código de producto (SKU)**.
- Un cliente puede comprar muchos productos y un producto puede ser adquirido por muchos clientes.
- Nos interesa registrar la fecha de compra y la cantidad de unidades que el cliente compró de ese producto.

Ejemplo



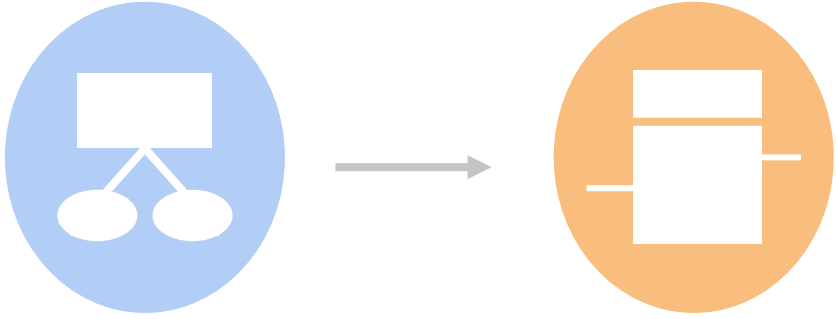
Modelo **Relacional**

Las **entidades y relaciones** se representan con **tablas**.
Las **tablas** contienen sus **atributos**.
Se detallan las **claves primarias y foráneas**.

Claves

- **Clave**:
  Columna o grupo de columnas que identifica unívocamente a cada fila.
- **Clave Candidata:**
  Columna o grupo de columnas que tienen el potencial de ser clave.
- **Clave Primaria:**
  Clave candidata elegida para identificar las filas de una tabla.
- **Clave Foránea:**
  Columna o grupo de columnas que hacen referencia a otra tabla.

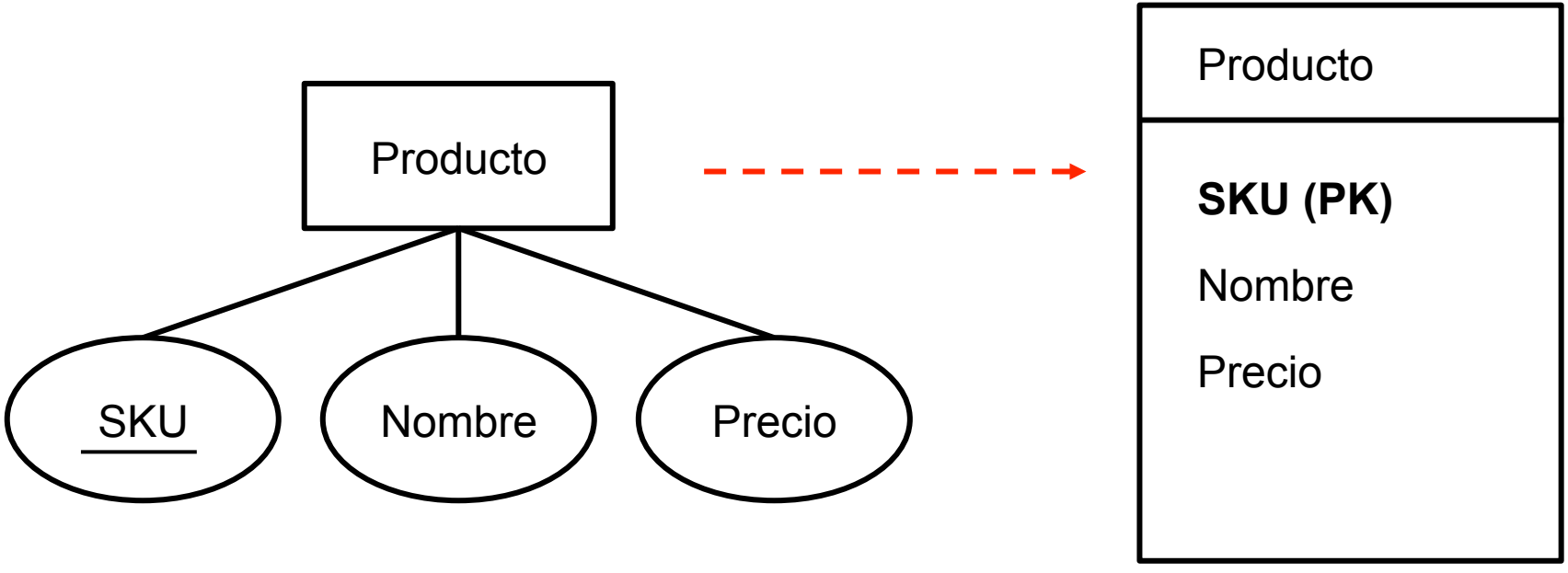
Modelos



Del modelo **Entidad Relación** al modelo **Relacional**

Entidades

Se representan con tablas. Los atributos de la entidad ahora serán atributos en la tabla.



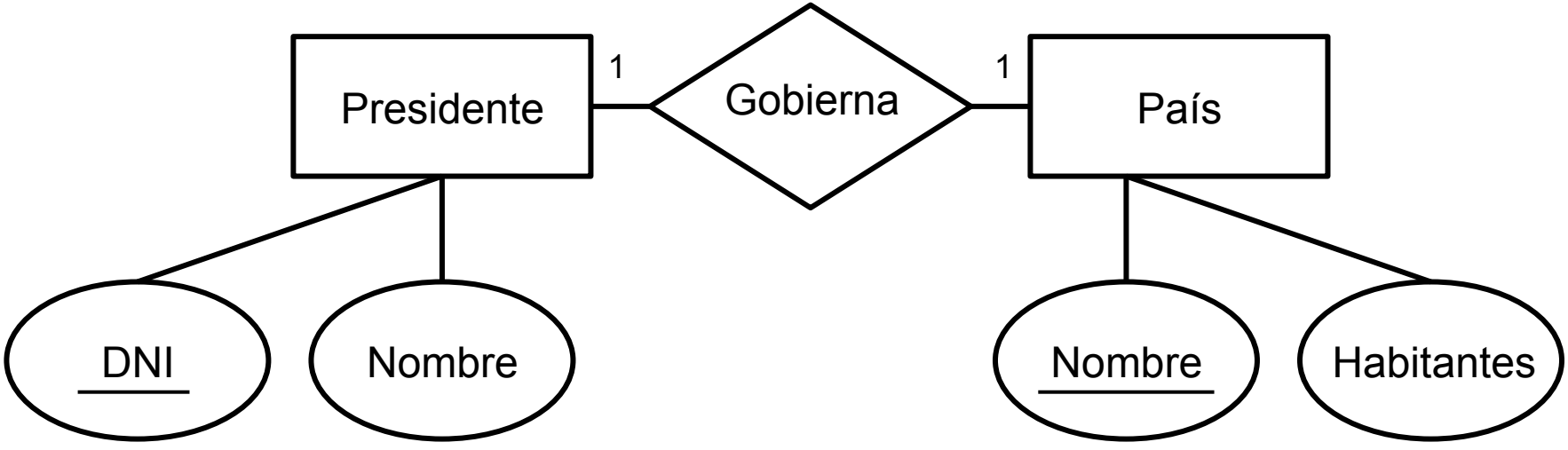
Relaciones **1 a 1**

La relación que existía en el modelo entidad relación, ahora se convierte en un atributo de alguna de las tablas.

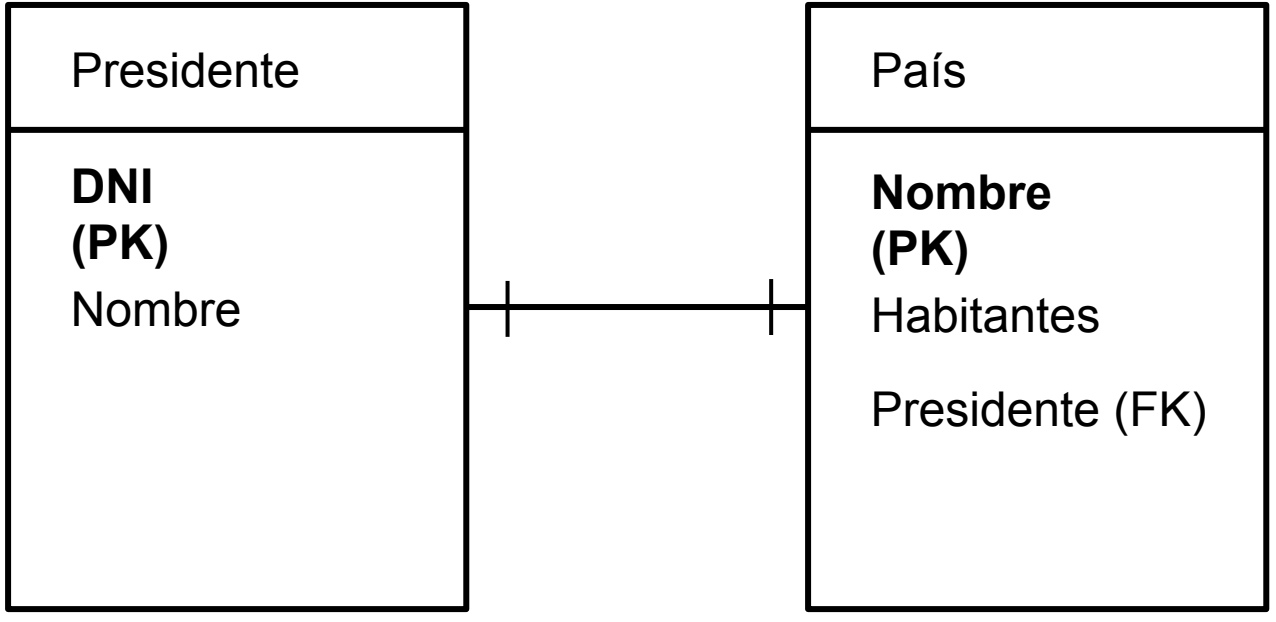
Ejemplo

- Un **presidente** gobierna un solo **País**.
- Del presidente nos interesa el **nombre, el apellido y el DNI**.
- Del **País** nos interesa el **nombre** y la **cantidad de habitantes.**

Modelo **Entidad Relación**



Modelo **Relacional**



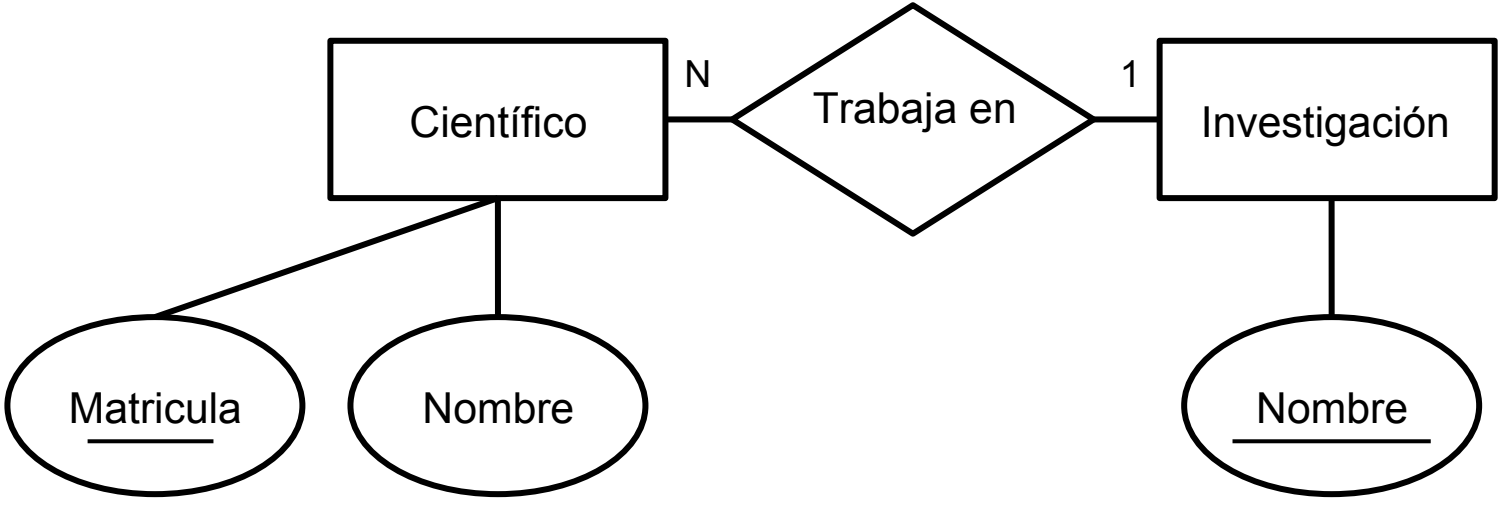
Relaciones **1 a N**

La relación que existía en el modelo entidad relación, ahora se convierte en un atributo de la tabla que posee la cardinalidad N.

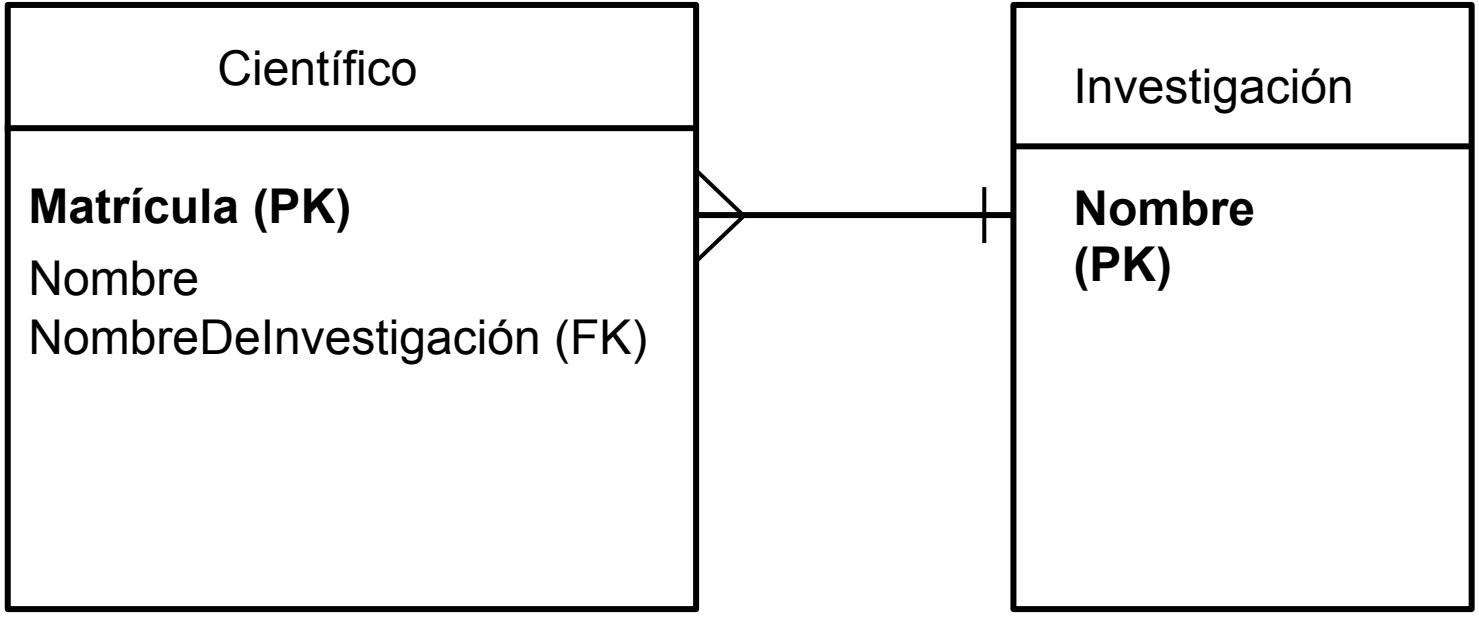
Ejemplo:

- Un científico trabaja en una única investigación
- Varios científicos pueden trabajar en una investigación.
- Cada investigación tiene un nombre único.
- Cada científico tiene un número de matrícula y un nombre.

Relaciones **1 a N**



Relaciones **1 a N**

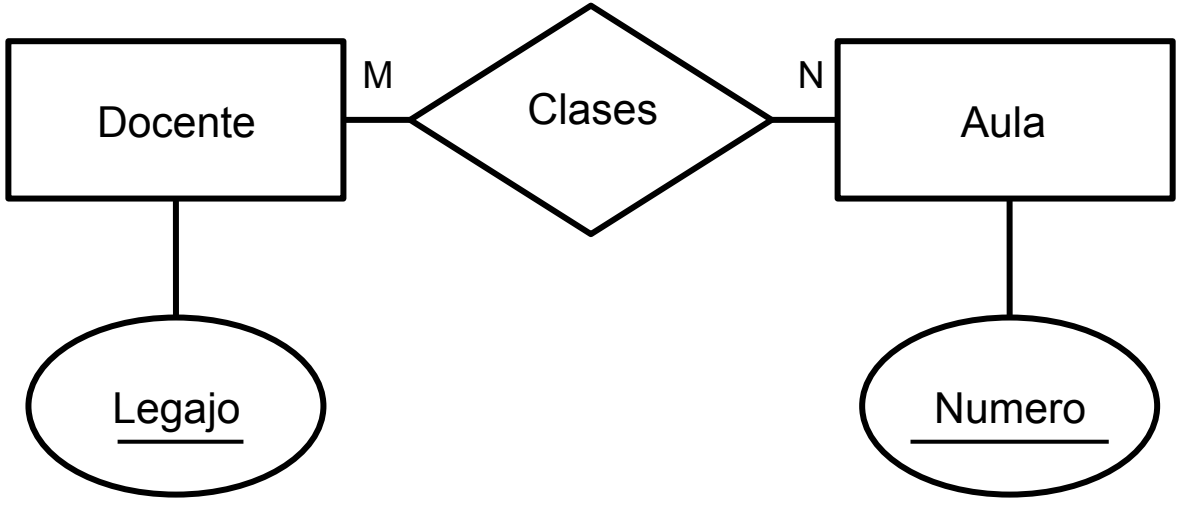


Relaciones **N a M** (muchos a muchos)

Ejemplo:

- Un docente puede dar clase en varias aulas y en varias aulas pueden dar clases varios docentes.
- Cada aula tiene un número distintivo
- Cada docente tiene un número de legajo distintivo.

Relaciones **N a M** (muchos a muchos)

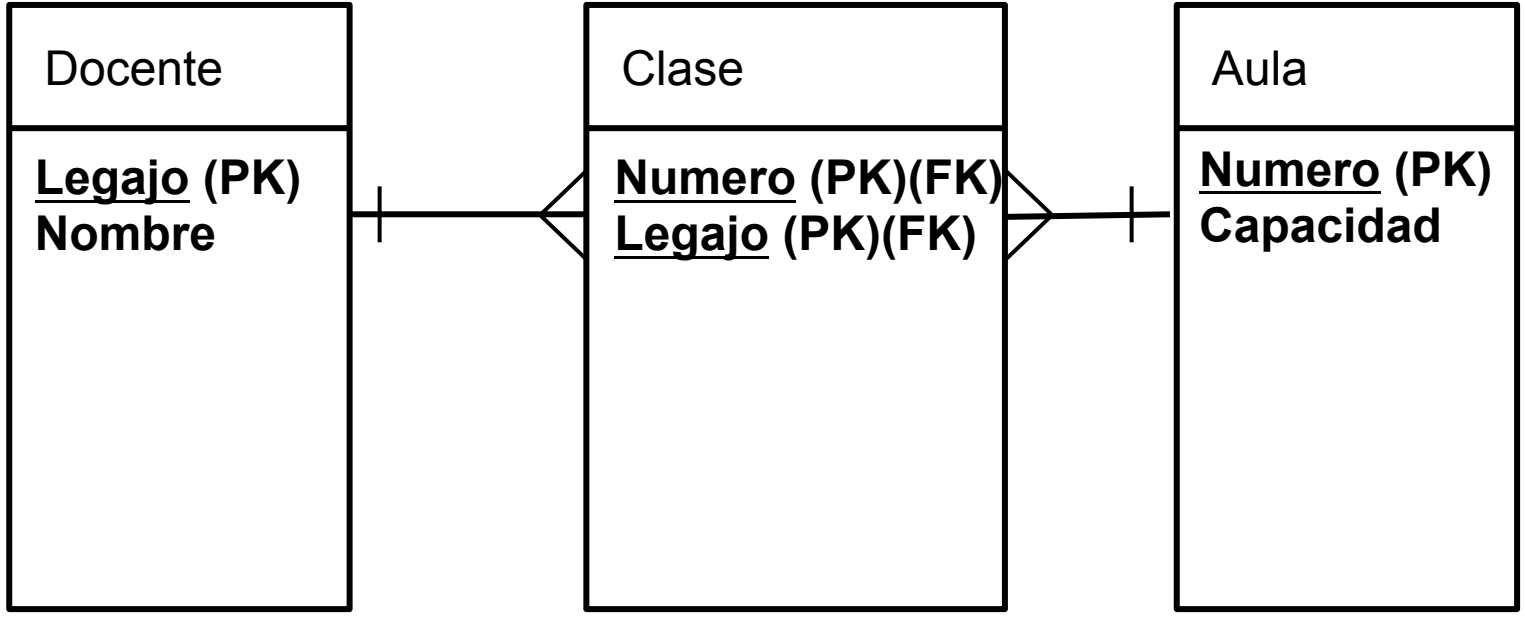


Relaciones **N a M** (muchos a muchos)

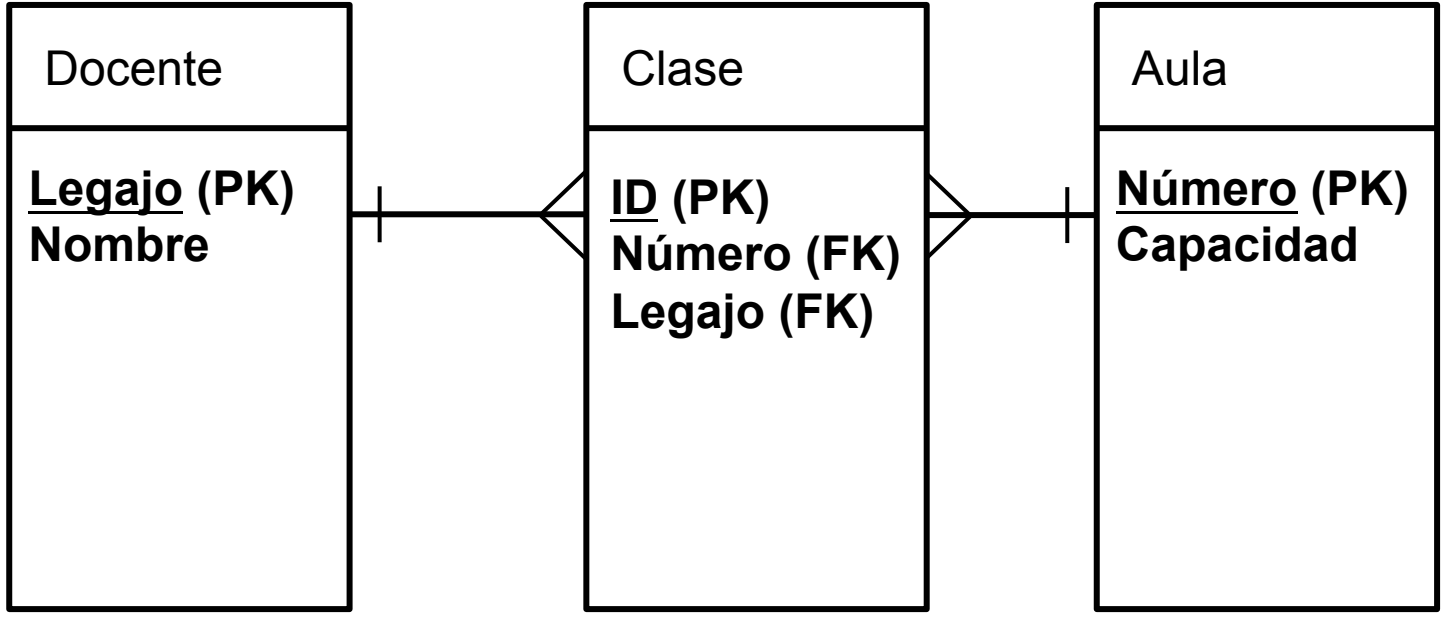
¿Cómo se transforma al modelo relacional?

En el caso de las relaciones de muchos a muchos para mantener la unicidad se requiere una nueva **tabla intermedia** que va a **representar la relación** muchos a muchos.

Relaciones **N a M** (muchos a muchos)



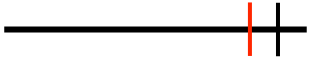
Relaciones **N a M** (muchos a muchos)



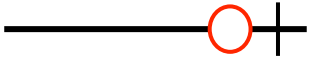
Relaciones **Cardinales** (opcionales / obligatorias)

Una entidad puede estar relacionada con otra de dos maneras.

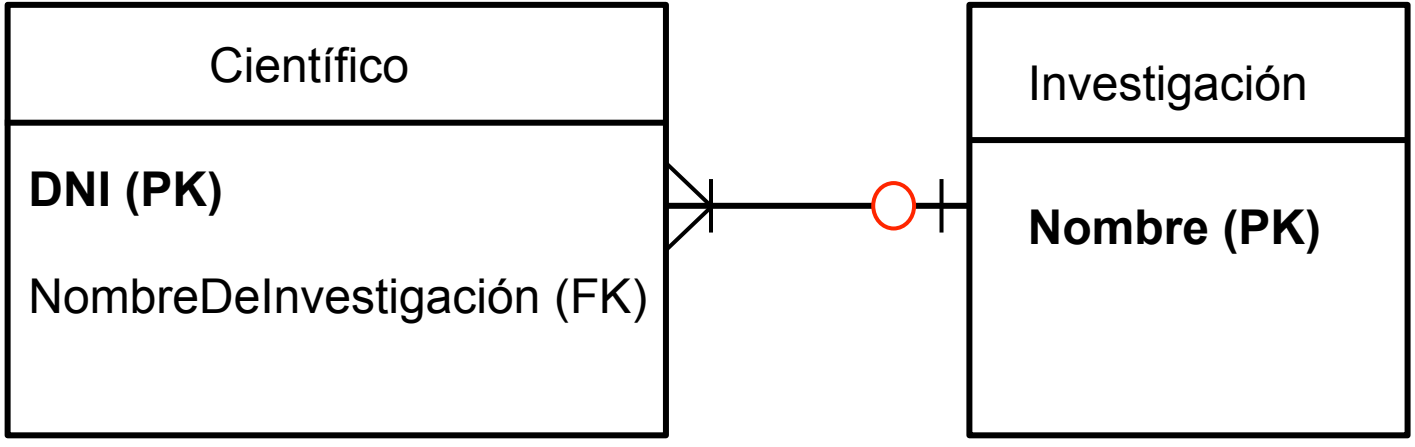
- ➔ Obligatoria



- ➔ Opcional

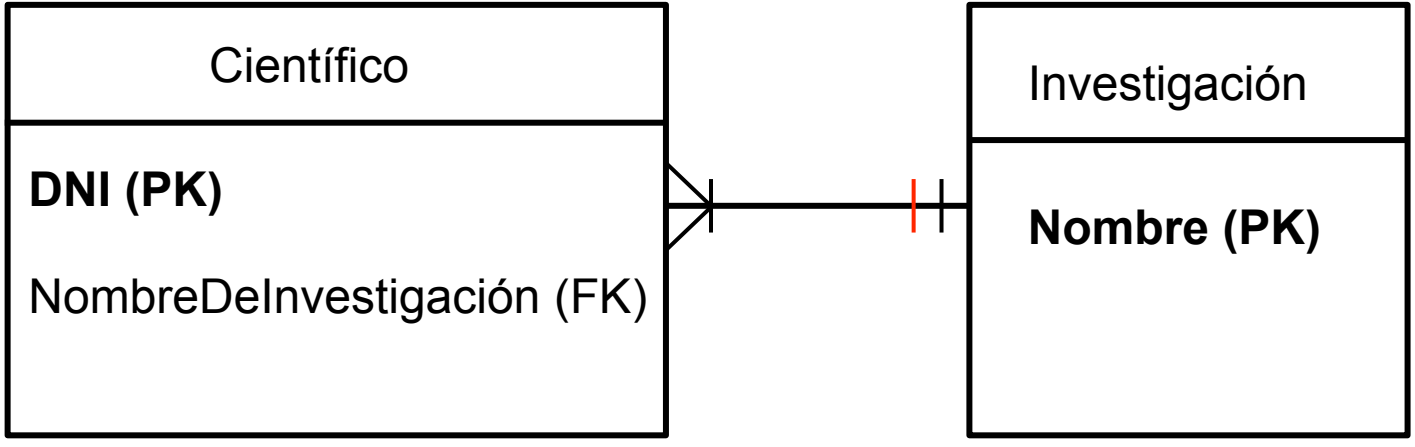


Relaciones **opcionales**



Un científico **puede tener** una investigación

Relaciones **obligatorias**



Un científico **debe tener** una investigación

Una compañía aérea necesita una base de datos para registrar la información de sus vuelos. Los vuelos tienen un identificador único. Además, cada vuelo tiene asignado un aeropuerto de origen y uno de destino (se asume que no hay escalas).

Los aeropuertos están identificados por unas siglas únicas (por ejemplo: VLC-Valencia, BCN-Barcelona, MAD-Madrid). Además, de cada aeropuerto se guarda el nombre de la ciudad en la que está situado y el país.

Cada vuelo es realizado por un avión. Los aviones tienen una matrícula que los identifica, el fabricante, un modelo e información sobre su capacidad (número máximo de pasajeros) y autonomía de vuelo (en horas). La asignación de aviones a vuelos no es única, así que es necesario saber la fecha en la que un avión realizó cada uno de los vuelos asignados.

NORMALIZACIÓN DE BASES DE DATOS

— Las formas normales de una base de datos fueron planteadas por Boyce y Codd a principios de la década del ‘70

¿Para qué normalizar un base de datos?

— Tres objetivos principales:

- **Garantizar la integridad** de la información
- **Evitar redundancia** en los datos
- **Escalabilidad**: que el modelo soporte modificaciones y extensiones con un bajo impacto

La primera forma normal exige los siguientes puntos:

- Eliminar los grupos repetidos en celdas individuales, cada celda debe contener un atributo “atómico” o indivisible
- Crear tablas separadas para cada conjunto de observaciones relacionadas
- Identificar a cada tabla con una clave primaria

PRIMERA FORMA NORMAL - EJEMPLO 1

DigitalHouse> Coding School

NO CUMPLE

Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025, 192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53; 182-929-2929
789	John	Doe	555-808-9633

Customer

Customer ID	First Name	Surname	Telephone Number1	Telephone Number2
123	Pooja	Singh	555-861-2025	192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53	182-929-2929
789	John	Doe	555-808-9633	

CUMPLE

Customer Name

Customer ID	First Name	Surname
123	Pooja	Singh
456	San	Zhang
789	John	Doe

Customer Telephone Number

Customer ID	Telephone Number
123	555-861-2025
123	192-122-1111
456	(555) 403-1659 Ext. 53
456	182-929-2929
789	555-808-9633

NO CUMPLE

DATA	
Curso	Contenido
Programación	Java, C++
Web	HTML, CSS, Php

CUMPLE

CURSO	
id_curso	descripcion
1	Programación
2	Web

CONTENIDO	
id_contenido	descripcion
1	Java
2	C++
3	HTML
4	CSS
5	php

CURSO_CONTENIDO	
id_curso	**id_contenido**
1	1
1	2
2	3
2	4
2	5

Además de cumplir con la primera forma normal la segunda forma normal exige:

- Que todos los atributos **que no forman parte** de la clave primaria, **dependan de todos** los componentes de la clave primaria.

- Si uno de los atributos depende únicamente de una parte de la clave primaria, entonces no se cumple la segunda forma normal.

Electric Toothbrush Models

Manufacturer	Model	Model Full Name	Manufacturer Country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

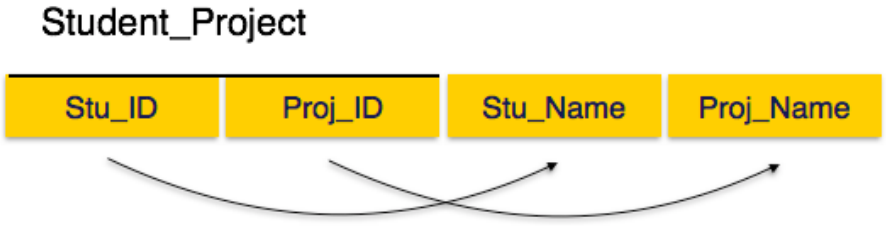
Electric Toothbrush Manufacturers

Manufacturer	Manufacturer Country
Forte	Italy
Dent-o-Fresh	USA
Kobayashi	Japan
Hoch	Germany

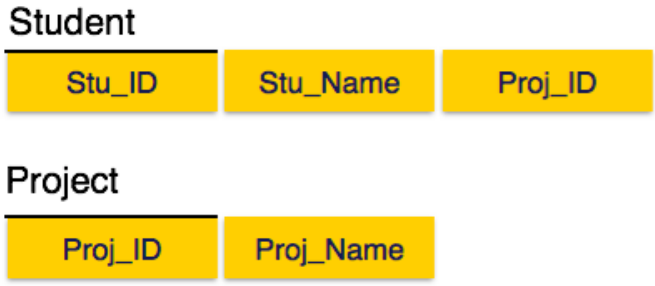
Electric Toothbrush Models

Manufacturer	Model	Model Full Name
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime

NO CUMPLE



CUMPLE



Además de cumplir con la segunda forma normal la tercera forma normal exige:

- Que ninguno de los atributos que no forman parte de la clave primaria dependan transitivamente de alguno de los otros atributos

La tercera forma normal se puede parafrasear de la siguiente manera:

> *"Every non-keyattribute must provide a fact about the key, the whole key, and nothing but the key."*

DigitalHouse> Coding School

NO CUMPLE

Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

CUMPLE

Tournament Winners

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

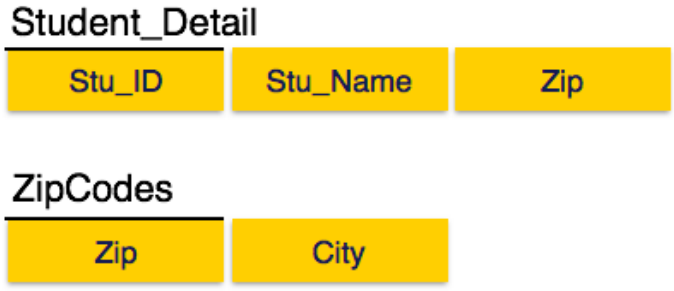
Winner Dates of Birth

Winner	Date of Birth
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

TERCERA FORMA NORMAL - EJEMPLO 2

NO CUMPLE

CUMPLE



- En esta clase hemos comenzado a descubrir todo el potencial de las bases de datos relacionales mediante JOINs y subquerys. Estos nos permiten mezclar y combinar datos de varias tablas, con el fin de extraer resultados útiles.

SQL VS NoSQL

- Las bases de datos SQL se llaman principalmente bases de datos relacionales (RDBMS); mientras que la base de datos NoSQL se llama principalmente como base de datos no relacional o distribuida.

- Las bases de datos SQL son bases de datos basadas en tablas, mientras que las bases de datos NoSQL son document based, key-value pairs, graph databases or wide-column stores.

- Esto significa que las bases de datos SQL representan datos en forma de tablas que consisten en n filas de datos, mientras que las bases de datos NoSQL son la colección de pares clave-valor, documentos, bases de datos de grafoss o wide column stores que no tienen definiciones de esquema estándar al que debe adherirse.

- Las bases de datos SQL tienen un esquema predefinido, mientras que las bases de datos NoSQL tienen un esquema dinámico para los datos no estructurados.

- Las bases de datos SQL son escalables verticalmente mientras que las bases de datos NoSQL son escalables horizontalmente. Las bases de datos SQL se escalan aumentando la potencia del hardware. Las bases de datos NoSQL se escalan aumentando los servidores de bases de datos en el grupo de recursos para reducir la carga.

- Las bases de datos SQL usan SQL (lenguaje de consulta estructurado) para definir y manipular los datos, lo cual es muy poderoso. En la base de datos NoSQL, las consultas se centran en las colecciones de documentos. A veces también se llama como UnQL (lenguaje de consulta no estructurada). La sintaxis del uso de UnQL varía de una base de datos a otra.

- Ejemplos de bases de datos SQL: MySql, Oracle, Sqlite, Postgres y MS-SQL. Ejemplos de bases de datos NoSQL: MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j y CouchDb

- Las bases de datos SQL son adecuadas para el entorno complejo de consultas complejas, mientras que las bases de datos NoSQL no son adecuadas para consultas complejas. A alto nivel NoSQL no tiene interfaces estándar para realizar consultas complejas, y las consultas en sí mismas en NoSQL no son tan potentes como el lenguaje de consulta SQL.

- Las bases de datos SQL no son las más adecuadas para el almacenamiento jerárquico de datos. Sin embargo, la base de datos NoSQL se ajusta mejor al almacenamiento de datos jerárquico ya que sigue la forma de pares clave-valor de almacenar datos similares a los datos JSON.

- La base de datos NoSQL es muy recomendable para grandes conjuntos de datos (es decir, para Big Data). Hbase es un ejemplo para este propósito.

- En la mayoría de las situaciones típicas, las bases de datos SQL son escalables verticalmente. Puede gestionar el aumento de la carga aumentando la CPU, la RAM, la SSD, etc. en un solo servidor.

- Por otro lado, las bases de datos NoSQL son escalables horizontalmente. Simplemente puede agregar algunos servidores más fácilmente en su infraestructura de base de datos NoSQL para manejar el gran tráfico.

- Las bases de datos SQL son las más adecuadas para aplicaciones de tipo transaccional de servicio pesado, ya que es más estable y promete la atomicidad así como la integridad de los datos. Si bien puede utilizar NoSQL para fines de transacciones, todavía no es comparable y lo suficientemente estable en alta carga y para aplicaciones transaccionales complejas.

- La mayoría de las bases de datos NoSQL están diseñadas para lograr dos de estas propiedades a costa de otra propiedad.

Consistencia

- Esto demuestra la garantía en la ejecución de las actualizaciones y la disponibilidad de las actualizaciones tan pronto como se confirman en el actualizador.

Disponibilidad

- La mayoría de las bases de datos SQL eliminan las consultas si los tiempos de carga / ejecución son mayores. Se espera que la disponibilidad sea muy alta y se espera que los tiempos de respuesta sean muy bajos en las bases de datos NoSQL mediante la eliminación de las propiedades transaccionales que están presentes en las bases de datos SQL.

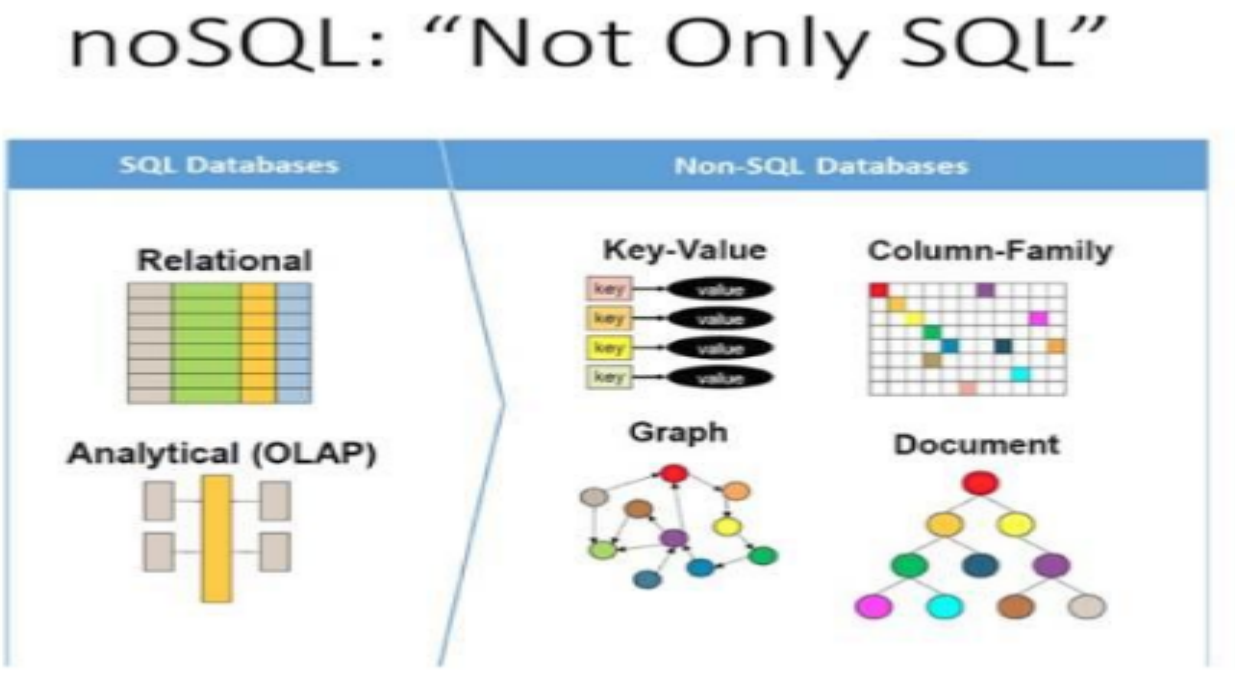
Tolerancia de partición

- La propiedad de las bases de datos que pueden funcionar con fallas entre los nodos debido a problemas de red.

- Se garantiza que las bases de datos NoSQL se adhieren a dos de las propiedades de CAP.

Tales bases de datos son de varios tipos.

- Key-Value Store: almacena en forma de tabla hash {Ejemplo- Riak, Amazon S3 (Dynamo), Redis}
- Document- Based Store : almacena objetos, principalmente JSON, que es web friendly o admite ODM (Object Document Mappings). {Ejemplo- CouchDB, MongoDB}
- Column-based store : cada bloque de almacenamiento contiene datos de una sola columna {Ejemplo- HBase, Cassandra}
- Graph-based: Representación con grafos de las relaciones, principalmente utilizada por las redes sociales. {Ejemplo- Neo4J}


noSQL: “Not Only SQL”
SQL Databases
Non-SQL Databases
Relational
Analytical (OLAP)
Key-Value
key
value
Column-Family
Graph
Document

DATABASES MARKET SHARE TABLE

Ranking	Technology	Domains	Market Share
1	MySQL	18,768	19.69%
2	Microsoft SQL Server	18,759	19.68%
3	NoSQL	11,156	11.70%