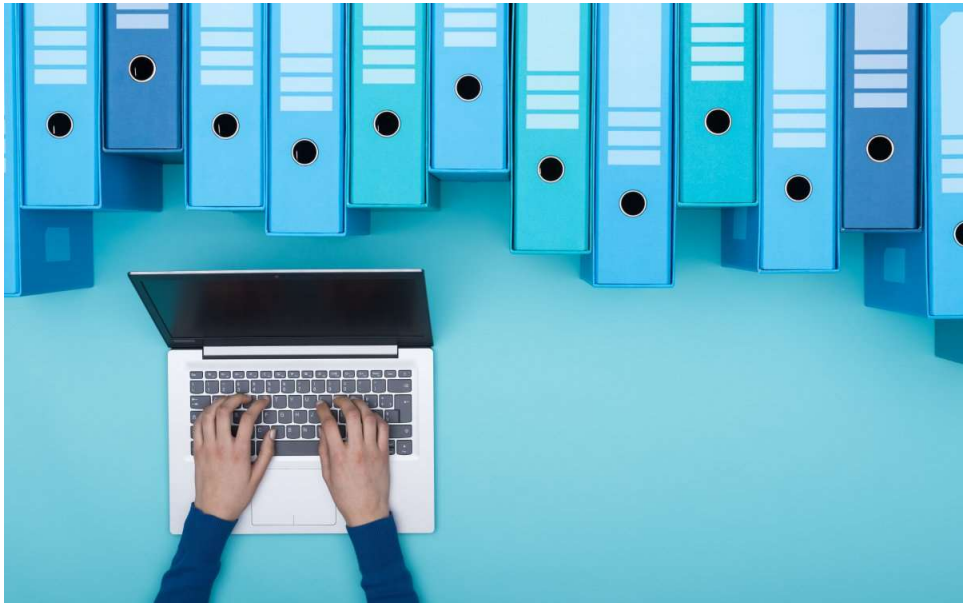
CILSA
LA INCLUSIÓN NOS UNE

INTRODUCCION A LAS BASES DE DATOS

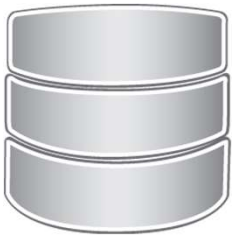


¿Que son las bases de datos?

• Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. (**Wikipedia**).

•Hay programas denominados sistemas gestores de bases de datos, abreviados DBMS por la sigla Data Base Management System, que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

•Las propiedades de estos DBMS, así como su utilización y la forma de administrarlos se estudian dentro del ámbito de la informática.



Símbolo con el que se suele representar a una base de datos.

Ok, pero. ¿Cual es su misión?

- Eliminar los formatos ad-hoc

Muy bien, pero. ¿Que sería eso?

- Si se le da a resolver a un grupo de **n** personas un problema que requiere guardar datos, y el trabajo de cada uno de ellos es totalmente independiente del trabajo de los otros, lo mas probable es que desde el punto de vista de los datos, haya **n** esquemas distintos.

- El uso de las bases de datos SQL ayuda a estructurar los datos, y accederlos en forma eficiente.

> "Si tus estructuras de datos están correctamente organizadas, los algoritmos se volverán evidentes". **Rob Pike. Ingeniero en software Canadiense, ex Laboratorios Bell.**

Funcionalidades de las bases de datos

Persistencia: Capacidad de los datos de perdurar en el tiempo, aún después de cerrar el programa, o de resetear el sistema informático.

Integridad: Es la propiedad de los DBMS de velar por la correctitud y completitud de los datos.

Ejemplos:

- Si una columna está marcada como **IS NOT NULL** verificar que no quede vacía.
- Si un campo es una **CLAVE PRIMARIA**, no permitir que se repita.

Esto de todas formas no es una tarea que sea completamente controlable por el DBMS.

Funcionalidades de las bases de datos (continuación)

Acceso: Provee una forma estandarizada de acceso a los datos.

Autenticación: Permite el uso de juegos de credenciales distintas a las del S.O. para acceder a los datos.

Particionamiento: Es la capacidad de separar la información contenida en la base de datos en diferentes partes, y cada una de ellas almacenarla en un dispositivo físico o lógico distinto, de forma tal de poder aprovechar al máximo el paralelismo de los motores de bases de datos.

Indexación: Propiedad clave de los DBMS para poder buscar un dato en forma rápida, sin tener que recorrer toda la tabla.

Normalmente el tamaño requerido para almacenar los índices en la base de datos es pequeño respecto al volumen de datos en sí.

Funcionalidades de las bases de datos (continuación 2)

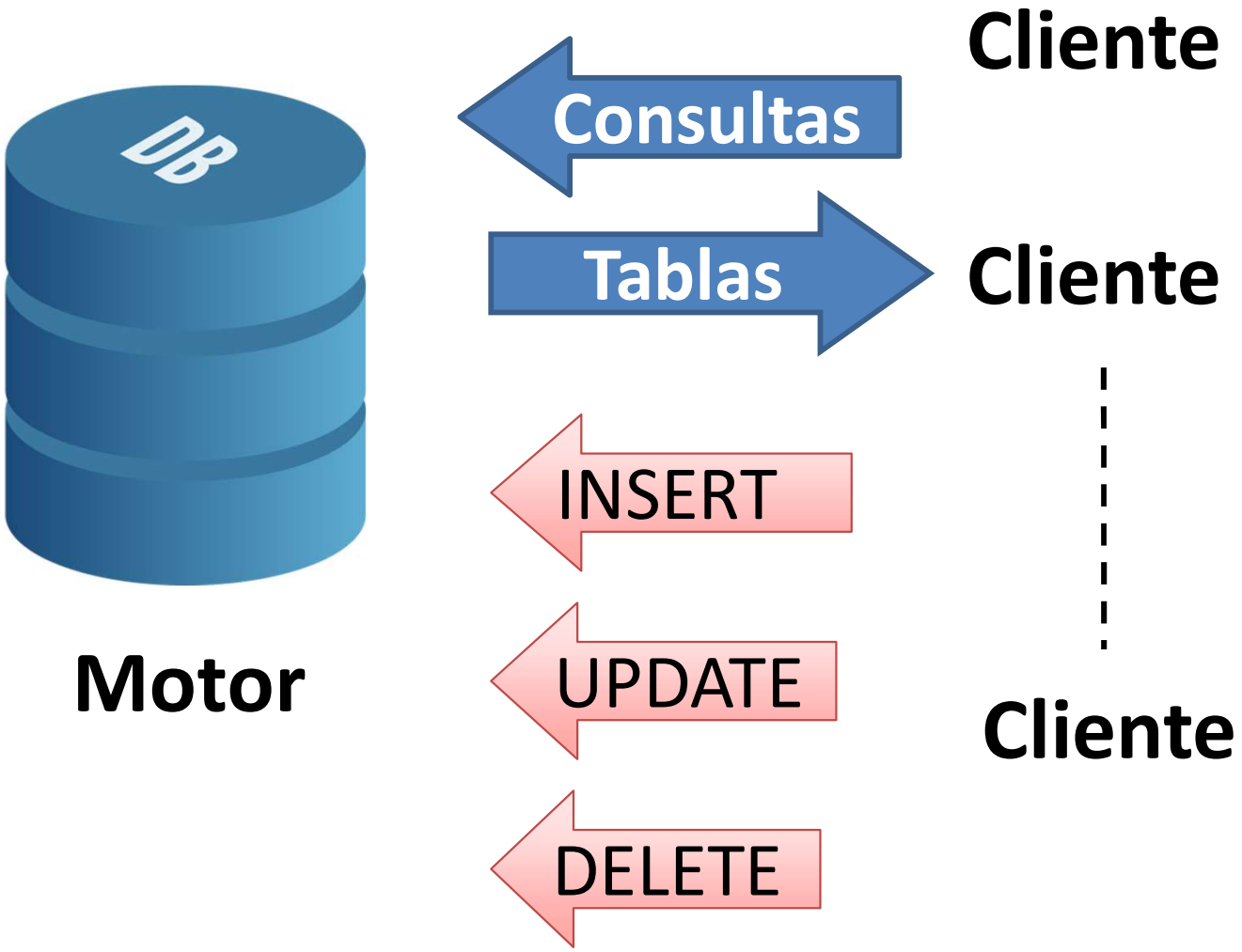
Transaccionado: Es la propiedad de una base de datos de poder efectuar un conjunto de instrucciones en forma atómica. Aplica principalmente en bases SQL.

Concurrencia: El DBMS coordina el acceso simultaneo a los datos por parte de varios programas.

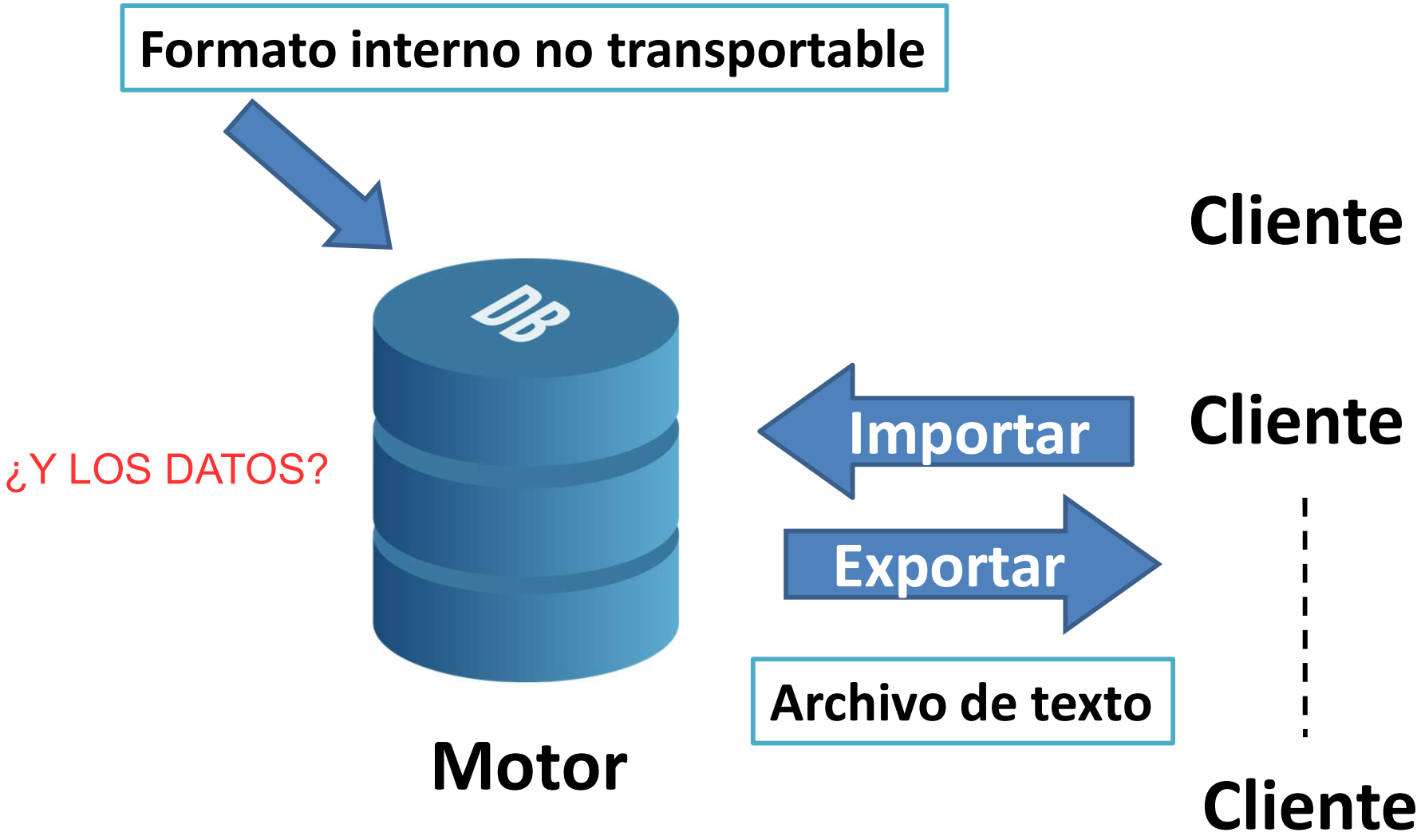
Escalabilidad: Es la capacidad de poder incrementar la cantidad de datos que maneja la base, sin que se vea degradada su performance. Como regla general, las bases noSQL son más fáciles de escalar.

Replicación: Es la propiedad de poder llevarse una copia exacta de una base de datos de un sistema a otro, y que este funcione cómo si los datos siempre le hubieran pertenecido.

Funcionamiento



Funcionamiento



Tipos de bases de datos

- **SQL (Structured Query Language):** Lo que significa lenguaje de consultas estructuradas.

Diseñado para manejar el acceso a los datos en las llamadas bases relacionales.

Principales bases SQL

Tipos de bases de datos

• **NoSQL (Not Only SQL):** Amplia clase de bases de datos que no utilizan el lenguaje de consultas estructurado para acceder a los datos (o por lo menos no es obligatorio que lo usen).

- No permiten operaciones del tipo JOIN
- No garantizan atomicidad, y por lo tanto, les resulta mas dificil garantizar la integridad de los datos.

Principales bases SQL

¿Cuando uso una y cuando uso otra?

No hay una regla definitiva para definir cual es el tipo de base de datos a utilizar, pero si hay perfiles que nos ayudan a tomar la decisión.

SQL	NOSQL
• Si se necesita confianza absoluta en la integridad de los datos. • Consistencia. • Soporte. • Sistema bien maduro.	• Poco presupuesto para hardware. • Tiendas Online • Acceso a grandes volúmenes de datos. • Cloud

SQLite

- Es una librería que implementa el standard SQL, y además.
    - Es autocontenido, lo que significa que no depende de nada externo.
    - Es transaccional.
    - Es embebible, o sea que se puede meter por completo dentro de nuestro sistema, aunque este tenga pocas capacidades de HW (lo cual es muy común para el HW que se suele manejar en electrónica).
    - Todo el motor está contenido en una librería en C de unos 350kB.
    - Puede residir en memoria.
    - Carece de servidor.
    - No se configura.
    - La base de datos es solamente un archivo. Nada mas.
    - Portable 100% Incluso entre sistemas con distintos File Systems.

Para que se usa

- Persistencia “Stand-Alone” o a nivel local en aplicaciones de escritorio.
- Websites de bajo tráfico.
- Reemplazo de estructuras de datos ad-hoc
- Prototipos, testeos.
- Bases de datos internas o temporales.
- Sistemas embebidos.
- Aplicaciones para celulares (perfectamente integrado en Android Studio por ejemplo).

Para que NO se usa

- Aplicaciones cliente-servidor.
- Websites de alto tráfico.
- Volúmenes de datos grandes.
- Aplicaciones concurrentes

Componentes de una base de datos

• **TABLA:** Es el elemento principal de una base de datos, ya que allí se almacenan los datos que se quiere gestionar.

Está compuesta por filas y columnas, como si se tratase de una hoja de cálculos.

Cada archivo de la base de datos puede contener tantas tablas como sean necesarias.

• **ENTIDAD:** Es la representación de un objeto o concepto del mundo real que se describe en una base de datos. Las entidades se describen en el estructura de la base de datos, utilizando un modelo de datos.

Cada entidad está constituida por uno o mas ATRIBUTOS.

Por ejemplo una entidad **Alumno** podría tner los atributos **nombre, apellido, fecha de nacimiento, legajo, etc.**

En las bases relacionales las entidades se relacionan entre si de distintas formas (1:1, 1:n, n:n)

Componentes de una base de datos

• **Primary Key (PK):** La clave primaria identifica de manera única a cada registro de una tabla. La columna que está definida como clave primaria debe cumplir con los siguientes puntos.

- No puede ser **NULL**.
- Debe ser **UNIQUE** o sea que no puede repetirse.
- Cada tabla además solo puede tener una clave primaria.

• **Foreign Key (FK):** La clave externa es una columna que identifica cual es la clave primaria de otra tabla que tiene una relación con esta.

- Puede tranquilamente haber mas de una FK por tabla.

Tipos de datos

Cuando se crea una tabla se debe especificar el tipo de dato que va a contener cada uno de sus atributos (o columnas)

Se muestra una lista de los tipos de datos (no exhaustiva)

Tipo de dato	Almacenamiento requerido
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT, INTEGER	4 bytes
BIGINT	8 bytes
FLOAT	4 bytes
DOUBLE	8 bytes
BIT (M)	Approx. (M+7)/8 bytes
DATE	3 bytes
TIME	3 bytes
DATETIME	8 bytes

Tipos de datos para strings

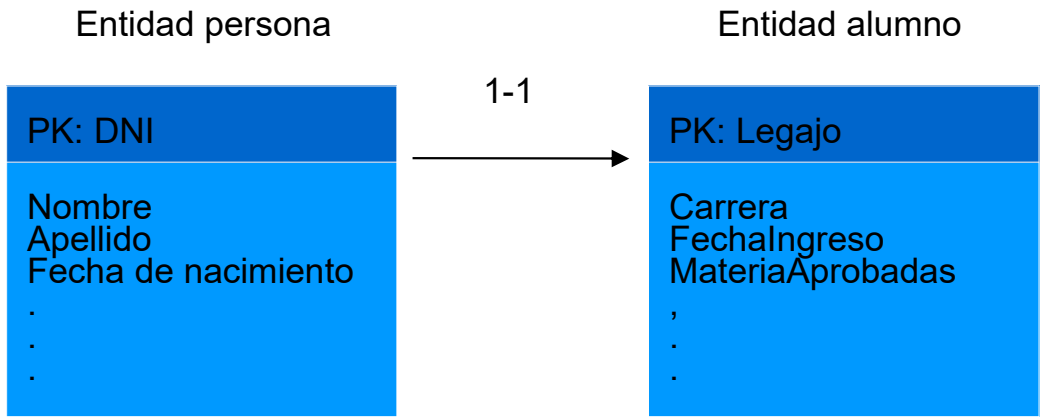
Tipo de dato	Almacenamiento requerido
CHAR(L)	L x w bytes (w cantidad de bytes del dato mas grande)
BINARY(L)	L bytes, 0 <= L <= 255
VARCHAR(L)	L+1 bytes sila columna requiere menos de 255 bytes, si no L+2 bytes
VARBINARY(M)	L+1 bytes sila columna requiere menos de 255 bytes, si no L+2 bytes
BLOB, TEXT	L+2 bytes, con L < 65536
LONGBLOB, LONGTEXT	L+2 bytes, con L < 2 exp 32
ENUM	1 o 2 bytes, depende de cuantos elementos tenga nuestra enumeración
SET	1, 2, 4 u 8 bytes, dependiendo del número de sets (máximo 64)

Tipos de relaciones

• **1-1:** Poco frecuentes. Por lo general si entre dos entidades existe una relación 1 a 1, estamos hablando de la misma entidad, y, por lo tanto, los atributos de una de ellas se pueden agregar a los atributos de la otra. Por ejemplo

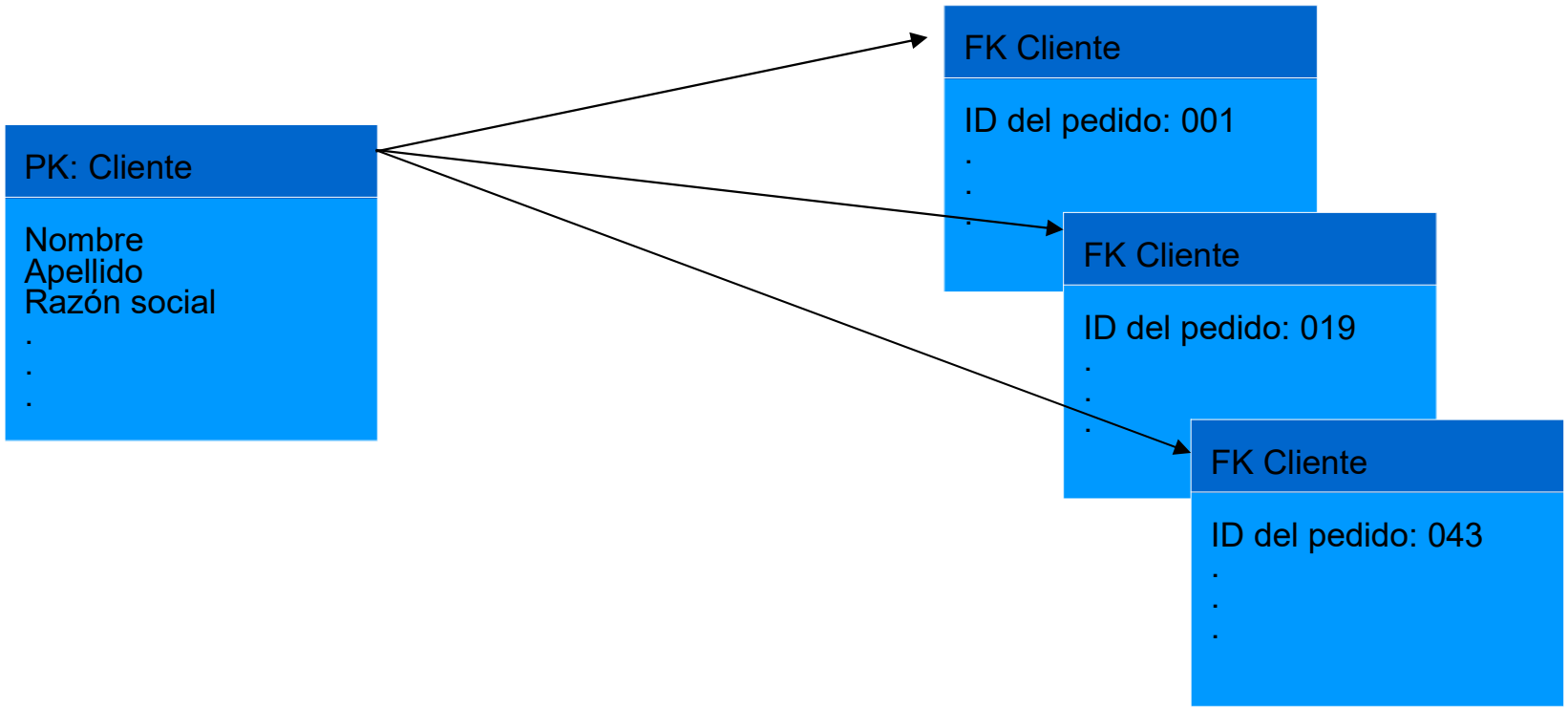
•Entidad persona y Entidad alumno.

- Cada persona tiene un único DNI
- Cada alumno tiene un único legajo



Tipos de relaciones

• **1-n:** Son las mas comunes. Por ejemplo, un cliente de una empresa, puede tener varios pedidos de mercadería.



Tipos de relaciones

• **n-n:** Son menos comunes que las otras, se trata de evitarlas durante el modelado pero no siempre se puede, y esto normalmente desemboca en tner que incorporar una Junction Entity que traduzca esa relación n-n en 2 relaciones 1-n.

•Un ejemplo que describe perfectamente una relación n-n son los productos y clientes de una empresa.

•Una empresa normalmente tiene n clientes y m productos. Cada uno de los n clientes puede adquirir cualquiera de los m productos, pero estos m productos a su vez pueden ser vendidos a cualquiera de los n clientes.