Bases de

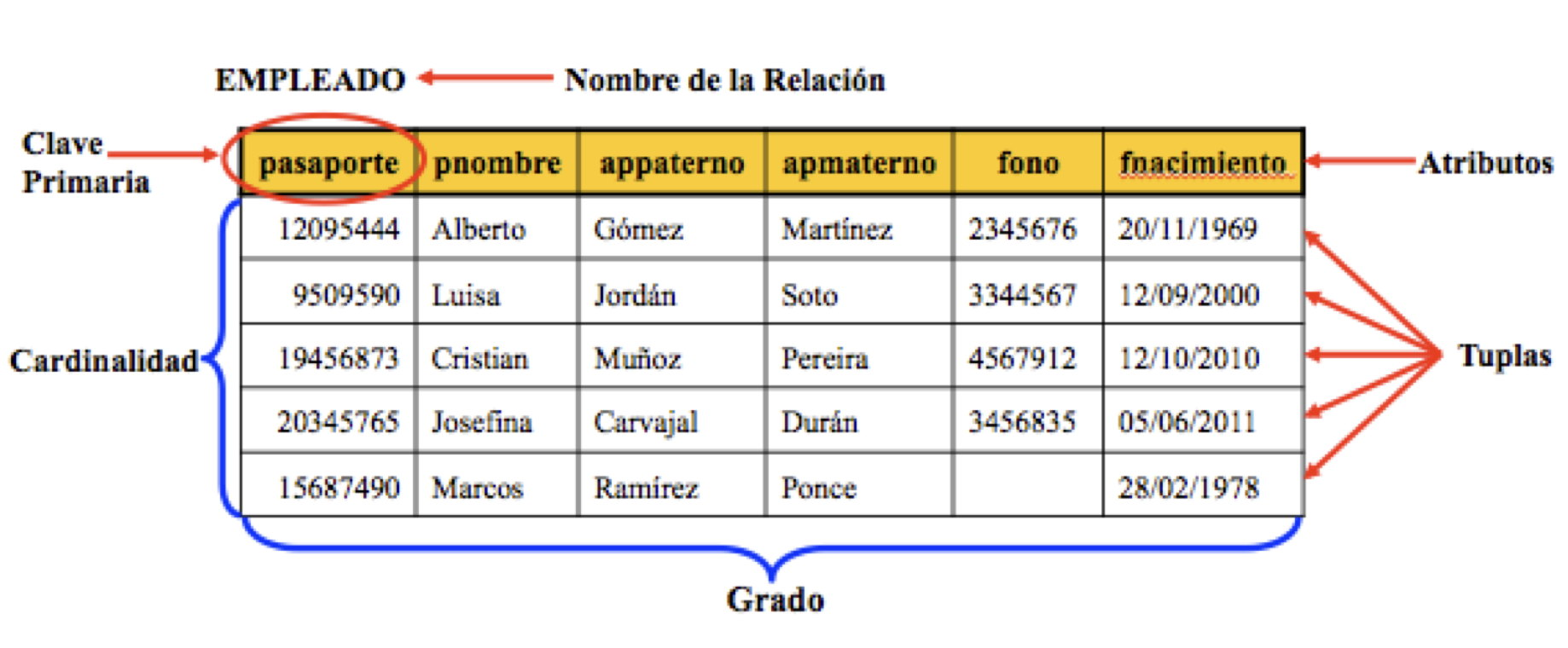
datos - EIN009B

**Modelo de base de datos:**

* Entidad.
* Ocurrencias: lista en entidad.
* Atributos.
  + Normal: cualificativo.
  + Identificador: permite individualizar a cada una de las ocurrencias de la entidad.
* Relaciones: se denominan por grados, según la cantidad de entidades relacionadas.
* Cardinalidad: cantidad en que se relacionan las entidades.
  + (0,1) = PUEDE tener 1 relación.
  + (0,N) = PUEDE tener relación con muchos.
  + (1,1) = DEBE tener relación uno a uno.
  + (1,N) = DEBE haber relación una a muchos.
  + (N,N) = DEBE haber relación muchos a muchos -> podría ser una ENTIDAD.

**Diseño Lógico.**

Proceso que forma parte diseño de Bases de Datos, el cual parte del esquema conceptual, resultando en un esquema lógico. Se selecciona el modelo de base de datos por adoptar (Modelo Jerárquico, Modelo Reticular, Modelo Relacional, Modelo Orientado a Objetos), generalmente se emplea el Modelo Relacional.



• Se emplean los términos Clave Principal (PK) para referirse al Atributo Identificador (o atributos identificadores), Clave Foránea (FK) a aquellos atributos que permiten la relación con una entidad externa, Clave Compuesta a aquellas claves conformadas por más de un atributo y Tuplas a las ocurrencias.

• Se emplea el Método de Normalización para organizar adecuadamente los datos, mediante el análisis de dependencias.

**Normalización.**

Proceso por medio del cual se pretende organizar los datos dentro de una base de datos relacional, en donde se minimice las redundancias, se proteja la integridad y se eliminen las inconsistencias, de tal forma que todas las tablas creadas estén relacionadas. Se intenta obtener **valores atómicos** (indivisibles) para evitar la redundancia de los datos.

• Primera Forma Normal (1FN):

• En todos los atributos los valores almacenados deben ser indivisibles.

• No deben existir grupos de valores repetidos.

• Segunda Forma Normal (2FN):

• No deben existir dependencias parciales (solamente sucede con claves compuestas).

• Tercera Forma Normal (3FN):

• No deben existir dependencias transitivas (un atributo depende de otro que no es parte de la clave).



Se selecciona una entidad y se evalúan los **atributos**:

**1.a.** Se eliminan los atributos **multivaluados**, es decir, aquellos que tienen varias respuestas distintas.

**1.b.** Se separan los atributos **compuestos**, en caso de que sea óptimo para nuestro caso.

**2.a.** Se da cuando la Clave Principal es una Clave Compuesta que genera **Dependencia Parcial**, que consiste en que un atributo depende (atributo dependiente) de otro (atributo dominante) que es sólo 1 parte de la Clave Principal.

**3.a.** Se revisa la existencia de **Dependencia Transitiva**, que se da cuando un atributo no clave depende de otro, aunque este NO sea Clave Principal o parte de ella.

* Cada vez que se elimina/separa un atributo multivaluado, compuesto o dependiente, se debe generar una nueva entidad que represente la relación entre ellas, donde el atributo principal de la entidad original se replica, convirtiéndose en una Clave Foránea (y Clave Principal de la nueva entidad), y el atributo principal se elimina de la entidad original y queda sólo en la nueva entidad.

**SQL: Structured Query Language**

¿Es un lenguaje de programación? Algunos dicen que no, debido a que no incluye “estructuras de control”. Aunque quienes dicen que sí, argumentan que “es un mecanismo de comunicación efectiva”.

Cuenta con 3 categorías de órdenes principales:

* DDL: Lenguaje de Definición de Datos.
* DML: Lenguaje de Manipulación de Datos.
* DCL: Lenguaje de Control de Datos (algunos la dividen en 2, “transitivas” y “x”).

DDL:

CREATE: crear tabla, índice, una vista

DROP: en general se usa para eliminar una tabla , índice o vista

ALTER: modificar estructura de una tabla.

DML:

SELECT: Selección (para consultas) de una o más tablas.

INSERT: insertar datos en una tabla.

UPDATE: modificar datos de una tabla.

DELETE: borrar registros de una tabla.

\*\*Método Pascal Case y Common Case.

**Tipos de notación**

Cuando programas debes trasladar palabras del mundo real o nombres de diferentes componentes al lenguaje que estás utilizando, representando dichos términos como variables, funciones o clases. Para ello debes eliminar los espacios de dichos nombres para que el compilador no falle y procurar que sean lo más legibles posibles.

El tipo de notación a usar depende de varios factores, como el lenguaje de programación que utilizas, el estándar que usas o tu preferencia personal, siendo lo más importante [que seas consistente con tu elección](https://www.neoguias.com/estilo-programacion-javascript/). Por ejemplo, en JavaScript es muy habitual utilizar el Camel Case para variables, nombres de funciones y métodos, y Pascal Case para los nombres de las clases.

**#1 Camel Case (contarElementos)**

La notación Camel Case combina las palabras directamente, sin usar ningún símbolo, estableciendo que la primera letra de cada palabra esté en mayúscula a excepción de la primera palabra, estando el resto de letras en minúsculas. Este tipo de notación está muy extendida, siendo su uso muy común tanto en la declaración de variables como en el nombre de funciones y métodos.

Aquí tienes varios ejemplos:

* La sintaxis de contar palabras en notación Camel Case sería contarPalabras.
* La sintaxis de aumentar nivel dificultad en notación Camel Case sería aumentarNivelDificultad.

**#2 Pascal Case (ContarElementos)**

La notación Pascal Case combina las palabras directamente, sin usar ningún símbolo, estableciendo que la primera letra de cada palabra esté en mayúscula sin excepciones, estando el resto de letras en minúsculas. Su uso es muy habitual en la definición de los nombres de las clases de múltiples lenguajes, como JavaScript o PHP.

Aquí tienes varios ejemplos:

* La sintaxis de contar palabras en notación Pascal Case sería ContarPalabras.
* La sintaxis de aumentar nivel dificultad en notación Pascal Case sería AumentarNivelDificultad.

| CREATE TABLE VENDEDOR (  ID\_VENDEDOR Number(3),  NOMBRE VarChar2(20),  CONSTRAINT PK\_VENDEDOR PRIMARY KEY (ID\_VENDEDOR)  );  --COMENTARIOOO  CREATE TABLE VENTA (  ID\_VENTA Number(5),  FECHA\_VENTA DATE,  ID\_VENDEDOR Number(3),  NUM\_BOLETA Number(6),  CONSTRAINT PK\_VENTA PRIMARY KEY (ID\_VENTA),  CONSTRAINT FK\_ID\_VENDEDOR FOREIGN KEY (ID\_VENDEDOR) REFERENCES VENDEDOR(ID\_VENDEDOR)  ); |
| --- |

EFICIENCIA DE MEMORIA:

* El manejo de memoria es más óptimo con VarChar2 que con Char, porque el primero hace un uso *variable* de memoria dependiendo de la cantidad de caracteres ingresados.
* Integer es más eficiente que Number.