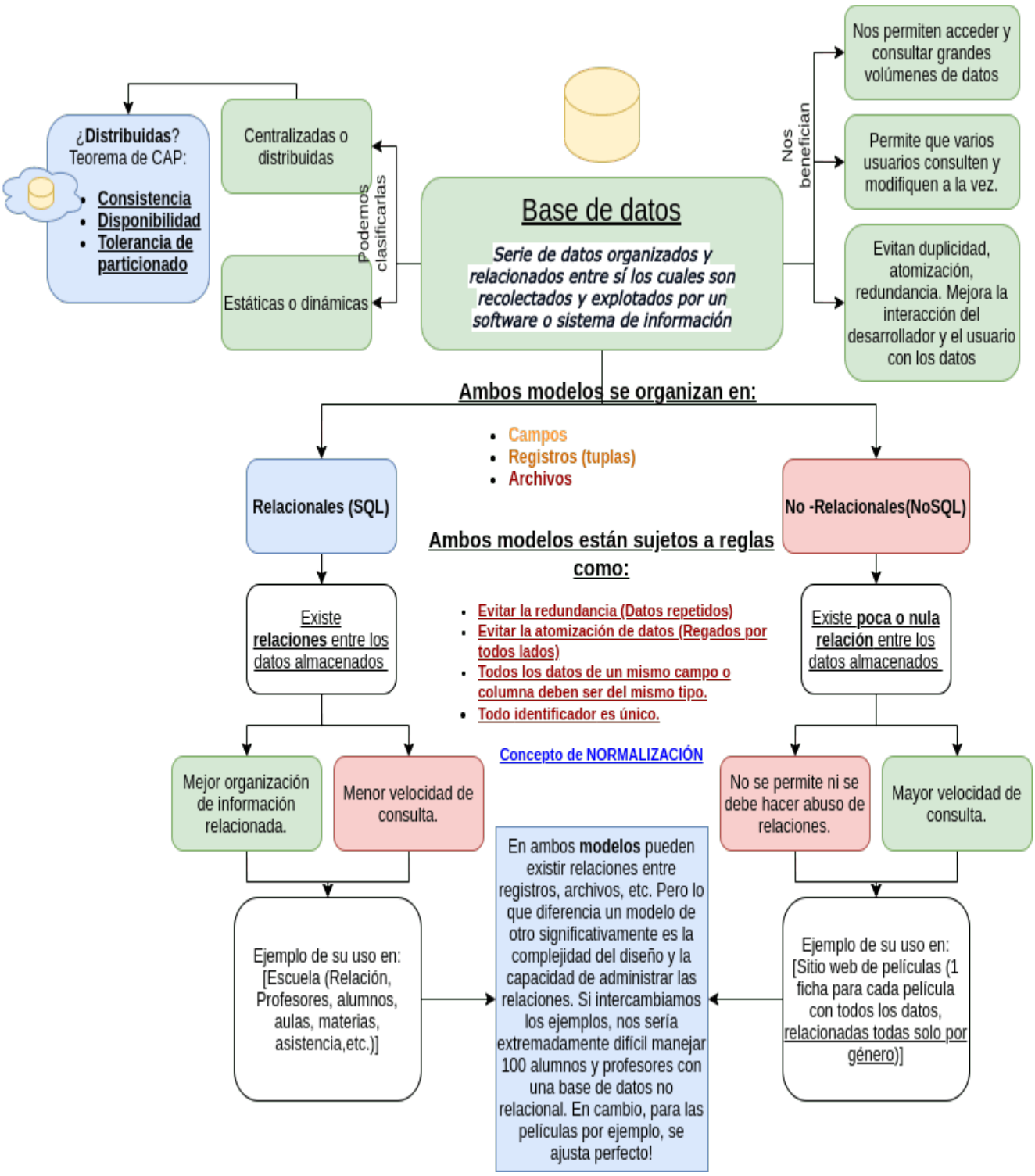
El modelo relacional de bases de datos.

Indice del contenido

Vimos anteriormente que existen dos modelos de bases de datos, el **modelo relacional** conocido popularmente como (SQL) y el modelo no-relacional o llamado (NoSQL).

Abajo te voy a dejar un diagrama explicativo de lo que vimos en la entrada anterior para que te resulte más fácil la comprensión de la introducción a modelos de bases de datos brindada anteriormente y porque es bueno para mí seguir un hilo mientras me explayo en los conceptos:



Comprender la importancia de la elección del modelo y su diseño.

Como ya sabes **predominan dos modelos** a la hora de diseñar una base de datos y es muy importante que comprendas estos conceptos, porque si te interesa aprender a programar de manera profesional tienes que tener

muy claro que ***la administración de la información es muy pero que muuy importante***.

Si programas un sistema de gestión de productos, clientes, o bien un simple sistema de registro y login de usuarios estarás trabajando con información sensible y personal, datos privados que las personas te han confiado!. Y quizás sin saberlo **tú eres responsable de su seguridad, gestión, organización, consulta o permanencia, entre otras**.

Y la elección del modelo y el diseño de la base de datos tienen muchísimo que ver!.

Cuando hablamos de **modelo relacional de bases de datos** nos referimos normalmente al modelo aplicado para trabajar con datos que tienen numerosas relaciones entre sí y que las seguirán teniendo a medida que se ingresen nuevos datos.

Al añadir nuevos datos o eliminar otros en la base de datos se crearán y eliminarán relaciones, por lo que si eliges mal el modelo o no la diseñas correctamente puede traducirse en desorden, desorganización, redundancia, errores, problemas y más problemas. Y créeme, los problemas con enormes cantidades de información que se relaciona entre sí se traducen en dolores de cabeza muy fuertes!.

La teoría del modelo relacional de bases de datos.

El **modelo relacional** se basa en el principio de que los datos son almacenados en campos según sus **Atributos (Columnas),** en **registros (tuplas, osea filas)** conformando así **archivos (Tablas)**. Y estas tablas o datos se pueden relacionar entre sí. Como vimos anteriormente (por arribita) se basa en:

Columna

Nombre de la tabla: **Trabajo**

Código	*Nombre*	*Posición*	*Salario*
1	Edgardo Trujillo	Gerente	19000
2	Lidimarie Fonsi	Empleada	12000
3	Jean Piaget	Empleado	13500
4	Jerome Bruner	Empleado	14000

Fila

Campo:

Un campo puede verse como una celda, un espacio, una variable (si quieres). La cual admite un solo **tipo de dato**(la mayoría de las veces), pero también puede admitir **diferentes tipos de datos** dentro del mismo campo (podría ser el ejemplo de un nick de usuario que contiene letras y números, o una dirección, etc.). Estos campos están organizados dentro de **registros**, y cada campo dentro del registro (podrías ver un registro como una tupla) tiene a su vez una organización por **columnas**, donde cada **columna** determina el **tipo de dato del campo**!. Así que si vamos a almacenar datos de una persona, un campo será su nombre, otro su número de teléfono por ejemplo.

Dónde el primero admite datos de tipo "String" (texto) y el segundo de tipo "int" (enteros). Y será así consecuentemente para todas las personas que se añadan, y cada persona será un registro, visto en forma de tupla:

("Marcelo", 34351532123).

("Carlos", 34351522113).

Nunca se admitirá en este caso añadir por ejemplo (342134561, "Carla"). O si se admitiera causaría enormes problemas a la hora de buscar a Carla por su nombre, puesto que en su nombre estará su número.. ¿Me entiendes?

Registro:

Un registro es un **conjunto de datos de campos** y puede verse como una tupla de datos o también como una **fila dentro de una tabla (gráficamente hablando)**. Así tendremos una tabla compuesta de **Columnas (Atributos, los cuales determinan el tipo de dato** y tienen un identificador, por ejemplo "Nombre" (tipo String)) y **Filas (Registros)** que determinan una Entidad, un registro puede ser una entidad. Por ejemplo el registro de una tabla alumnos puede contener el conjunto de campos:

("Josele", "Perez", 18, "Av Arias 101")

Archivo o tabla:

Podemos llamar **archivo** al conjunto de registros, pero normalmente se le llama así en el modelo no relacional. En el modelo relacional de bases de datos le llamamos **Tabla**. Y una base de datos puede estar compuesta de varias tablas, que a su vez contienen registros, que a su vez contienen campos. Por ejemplo la tabla **Alumnos**, y por otro lado la tabla **Profesores**.

Así, entre ambas tablas existe una relación, vista desde ambos puntos ¿Quién es el profesor de Josele? y ¿A quien da clases el profesor Oscar?. La tabla alumnos se relacionarán con la tabla profesores utilizando una **Clave o Llave**.

Clave o llave:

Una **clave o llave** es un **identificador único**. Es un dato que bien ***puede diferenciar una entidad de otra (un registro de otro) o bien relacionar una entidad con otra***. Pero lo cierto es que todo registro debe tener un identificador único para poder referirse a él (clave PRIMARIA) y normalmente se especifica en la columna que se trata del tipo "clave".

¡¡¡MUCHO OJO !!!

Un identificador único de la entidad **Persona** no puede ser nunca por ejemplo su "**Nombre**". ¿Por qué?. Pues que no sabes si **puede haber dos personas con el mismo nombre!!**. Y luego si quieres acceder únicamente a una de ellas ¿Cómo lo harías?. Ya tendrías un problema!.

Los Atributos a ser definidos como claves primarias han de ser elegidos cuidadosamente!.

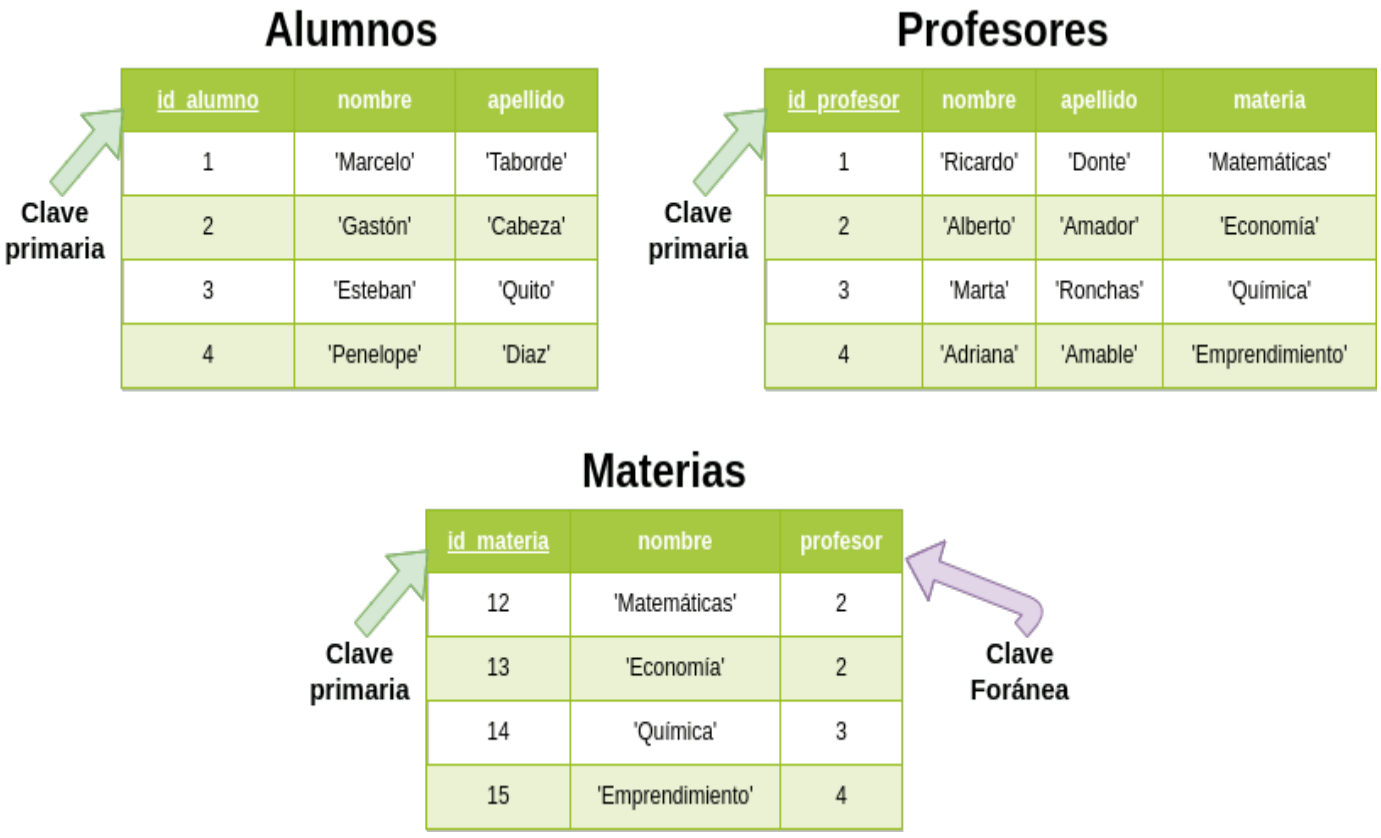
Al hablar de **IDENTIFICADOR ÚNICO** hablamos de **CLAVE / LLAVE PRIMARIA**.

Una **llave o clave primaria** dentro del modelo relacional es un campo o grupo de campos (llave primaria compuesta) que identifica de manera inequívoca un registro. **Ningún otro registro puede tener en el campo designado como clave o llave el mismo valor**. Por ello decimos que es **único e irrepetible y normalmente tiende a ser auto-incrementable**, por ejemplo al añadir dos registros a una base de datos, uno podría tener como clave primaria el "id = 1" y el otro el "id = 2" si eliminamos el primero y añadimos otro, automáticamente será asignado un "id = 3", **jamás volverá a ser ingresado un registro con la clave primaria eliminada** "1" ya que al haber existido anteriormente el volver a crearla nos enfrenta a lo que llamamos **inconsistencia de datos.** Y no solo podemos encontrar como clave primaria un **id**, a veces puede ser por ejemplo un **DNI**, un **CUIT**, etc.

Las claves primarias nos permiten también mediante la identificación inequívoca de un elemento (registro) dentro de una tabla, **relacionarlo con otro de otra tabla**. Como el ejemplo de **Alumnos** y **Profesores**, podemos colocar un "**id**" a cada **Alumno** (registro) y a cada **Profesor** (registro), y luego, en una tercera tabla llamada **Materias**, podemos usar la clave primaria de **Profesores**, para indicar que profesor dicta cada Materia, entonces, en la

tabla **Materias** nos encontraremos una clave primaria también que identifica la materia, pero además una **clave FORÁNEA** que identifica al profesor que la imparte.

Es importante diferenciar los dos tipos de claves o llaves:



Así las claves FORÁNEAS que identifican y determinan una *relación* pueden ser **internas** (estar definidas dentro de la misma tabla, EJEMPLO: dentro de profesores) o **externas** (estar definidas en una tabla aparte EJEMPLO: como en este caso en **MATERIAS**).

Ejemplo:

En base al ejemplo anterior podemos decir que el profesor "**Alberto**" cuya clave primaria "**id_profesor**" es "2". Dicta dos materias cuya clave primaria **id_materia** es 12 y 13. Y corresponden a Matemáticas y Economía. 😉

Así que en resumen:

Una clave puede identificar una entidad (registro) de las demás o una relación (que también es una entidad) entre ellas. ¿Capisci?

Y de ello dependerá que sea primaria o foránea. Que a su vez las foráneas pueden ser internas (en la misma tabla) o externas (en una tabla aparte).

Clave o llave foránea

Vamos a ver otro ejemplo que no me quedo contento!

Ejemplo:

Este que ves es un diagrama conocido como **Entidad-Relación** que más adelante vamos a aprender a interpretar en el momento de diseñar bases de datos relacionales. Pero fijate que en cada tabla, excepto en VENTAS podemos encontrar claves primarias que **identificarán de manera inequívoca los registros almacenados en ella**. Por ejemplo: id_Cliente, id_Producto e id_Fecha.

Como ves las llamadas **llaves foráneas** son una limitación referencial entre dos tablas, esta identifica una columna o grupo de columna (atributos) en una tabla (hija o referendo) que se refiere en otra tabla (maestra o referenciada).**Las columnas en la tabla hija o referendo deben ser la clave primaria u otra candidata en la tabla referenciada**.

Y este es el caso de la tabla VENTAS. Las claves que encontramos dentro de esta tabla son **claves foráneas** que permiten relacionar los registros de las demás tablas dentro de VENTAS mediante sus claves primarias. Para al momento de hacer una venta registrar el id del cliente, del producto y de la fecha en que fue realizada la venta. De esa manera se crea una "**Relación**" entre las tablas.

Relaciones:

Considero la parte más importante del modelo relacional, puesto que claro. Se llama relacional carajos!, eso significa que su nombre deviene de las relaciones entre los datos y la facilidad con las que nos permitirá establecerlas.

Las relaciones son en las que tenemos que poner el ojo y de ellas dependerá la elección del modelo también!. Una base de datos relacional tendrá **relaciones definidas por el diseñador de la misma y estas relaciones tienen ciertos criterios que se deben respetar y existen también diferentes tipos de relaciones**. Pero todas están definidas por una clave o llave foránea interna o externa que determina la relación, como viste en el ejemplo anterior de la tablas CLIENTES, PRODUCTOS, FECHAS con VENTAS. Y el ejemplo anterior Profesores con Materia.

Dónde en este caso en las líneas que las unen podemos distinguir un número 1 y "n" lo que simboliza una relación en un sentido "**uno a varios**". De forma que un un cliente puede estar involucrado en varias ventas o varias ventas a un mismo cliente.

El tipo de dato nulo.

"*El que en una clave primaria te puede patear el culo..*" – *Edgar codd.*

El **valor nulo** es aquel que se establece en un campo que aún no ha sido definido o se desconoce su valor, pero no su tipo. Y la posibilidad de añadirlo depende de que el diseñador del modelo de bases de datos **lo permita**. Por ejemplo en la tabla **Alumnos** podemos permitir dejar vacío (**null**) el número de teléfono, porque puede haber algún alumno que no tenga smartphone.

Entonces, al quedar vacío simplemente se especifica una **valor NULO o null**.

¡¡¡CUIDADO!!!

Jamás, jamás debemos permitir valores nulos en campos que son importantes para realizar consultas u obtener información relevante, y ni hablemos de las claves. *JAMÁS UNA CLAVE NI PRIMARIA, NI FORÁNEA DEBE PERMITIR VALORES NULOS.*

Permitir valores nulos se configura al momento de crear la tabla y las relaciones, así podemos evitar que el usuario deje en blanco atributos como por ejemplo "su nombre, apellido, edad, etc.". También claramente un valor nulo luego puede ser reemplazado por un valor real, pero sólo del tipo que se

especificó en el atributo al momento de la creación de la tabla. Por ejemplo, un usuario puede dejar en blanco su edad, por lo que obtendremos un "**null**" en este **atributo**, y luego en un futuro puede completarlo. Pero, claro **jamás podrá colocar otro valor que no sea un número entero**, porque el diseñador de la base de datos relacional ya ha especificado el **tipo de dato** aceptado. ¿Me explico?
Resumen gráfico para los que gustan ver imágenes!

Pues bien, hasta aquí ya tienes una base del **modelo relacional** bastante amplia para comenzar a trabajar y diseñar una base de datos. Que es justamente lo que vamos a hacer en la siguiente entrada..
Donde aprenderemos el **Diagrama Entidad- Relación, las buenas prácticas y las reglas de normalización de las bases de datos, entre otras cosillas**.
La idea es que al finalizar logres diseñar tus propias bases de datos relacionales y puedas trabajar con ellas mediante **MySQL** e integrarlas a tus aplicaciones en Python. Para lo cual utilizaremos diversas herramientas!.