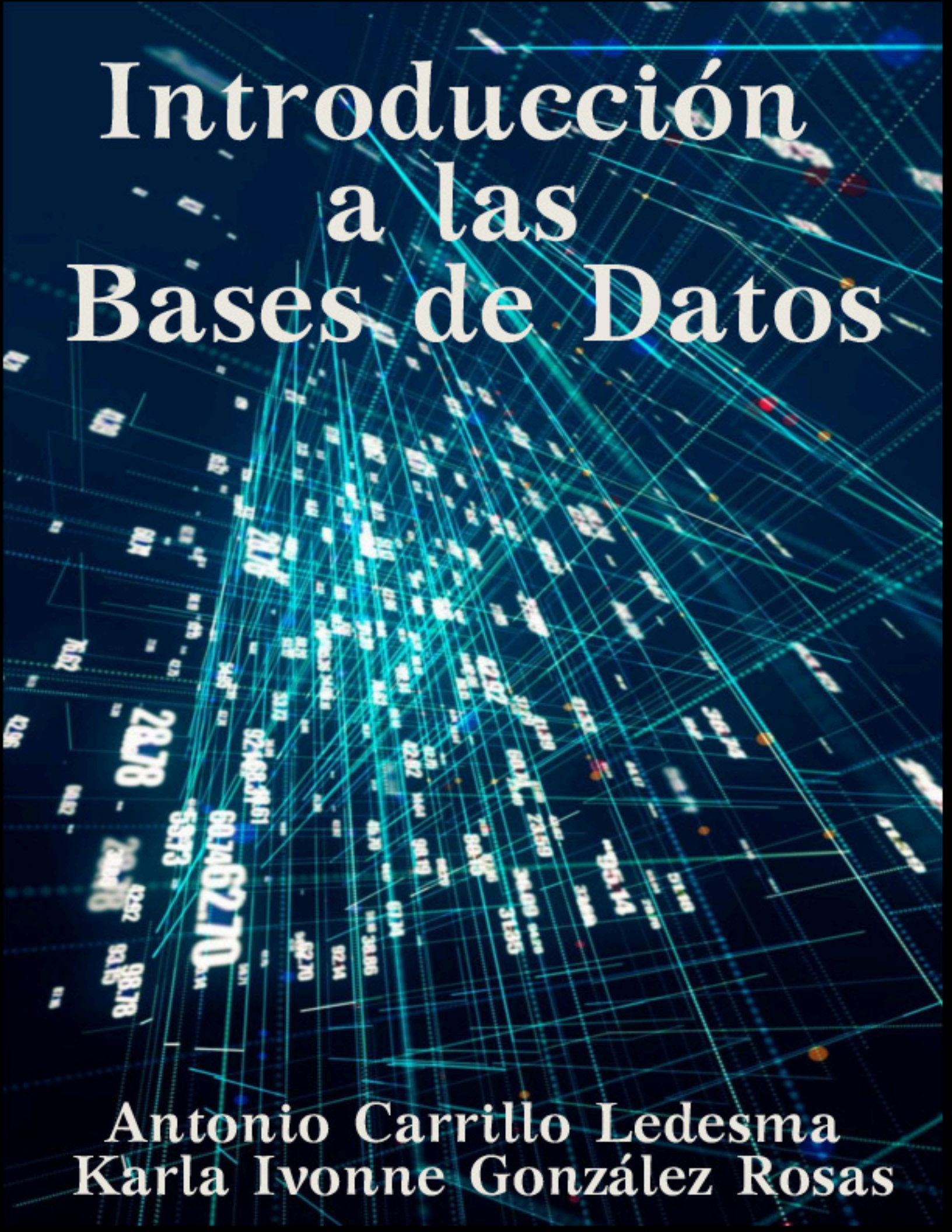

Introducción a las Bases de Datos

Antonio Carrillo Ledesma
Karla Ivonne González Rosas

Introducción a las Bases de Datos

Antonio Carrillo Ledesma y Karla Ivonne González Rosas
Facultad de Ciencias, UNAM
http://academicos.fciencias.unam.mx/antoniocarrillo

La última versión de este trabajo se puede descargar de la página:
https://sites.google.com/ciencias.unam.mx/acl/en-desarrollo
http://132.248.181.216/acl/EnDesarrollo.html

2025, Versión 1.0α[1]

Índice

1 Introducción

Con el aumento del acceso a internet por parte de personas y empresas se ha generado un auge de las redes sociales, comercio electrónico y demás actividades que implican el uso del cómputo, generando un aumento exponencial de la información y datos que se generan a los cuales se puede tener acceso. El gran reto del manejo de datos es el poder recolectar, almacenar, estructurar y analizar grandes cantidades de información en el menor tiempo posible que pueden ser usados para la toma de decisiones, adquirir conocimiento y otras tantas nuevas posibilidades que se abren con el paso de los años.

Comencemos por algunos conceptos fundamentales:

- Dato: cualquier valor aislado
- Contexto: para darle sentido al dato es necesario ponerlo en un contexto, el cual será fundamental para realizar una interpretación
- Información: la interpretación de un dato con un contexto se convierte en información
- Patrones: cuando se analiza la información se pueden descubrir patrones; es decir, que hay ciertas cosas que se repiten constantemente a lo largo del tiempo bajo determinadas condiciones
- Conocimiento: al descubrir patrones se puede generar conocimiento, por qué basándose en casos anteriores es posible predecir qué va a pasar en determinadas situaciones
- Toma de decisiones: con predicciones y conocimiento de un tema es posible tomar una decisión informado sobre escenarios futuros

El manejo de datos, es una creciente parte de la ciencia, que algunas veces es llamada ciencia de datos. Esta es usada en por ejemplo:

- Medicina: análisis de enfermedades o síntomas que se han detectado en cierta población y, con base en eso, saber si son susceptibles de adquirir alguna otra enfermedad
- Clima: a partir del análisis de ciertas condiciones climatológicas, se hacen predicciones sobre el estado del tiempo

- Mercadotecnia: a partir del análisis del patrón de comportamiento humano, se puede predecir si las personas comprarán o rechazarán un producto, en donde es preferible poner una tienda y un largo etc.

Así, mediante el análisis descriptivos de la consulta de la información almacenada localmente en bases de datos o la que se puede descargar de internet y gracias al uso de estadística y probabilidad, minería de datos y la incorporación de la inteligencia artificial y el aprendizaje de máquina se han podido descubrir patrones y modelos, y lograr análisis predictivos y prescriptivos entre muchos otros.

Para poder hacer realidad todo lo descrito anteriormente, ha sido necesario un desarrollo vertiginoso de nuevo y potente Hardware (que soporte el almacenamiento de la ingente cantidad de información y su interconexión mediante el uso de internet) como de toda una gama de Software especializado para soportar los nuevos sistemas de almacenamiento para los diferentes tipos de datos y para la recuperación de grandes volúmenes de información.

1.1 Bases de Datos de Código Abierto y sus Ventajas Frente a las Propietarias

Elegir la base de datos más adecuada para un proyecto supone enfrentarse a todo tipo de posibilidades de elección. Quizá la primera, y la más importante, es elegir una base de datos de código abierto o una propietaria. Es decir, una de código abierto, generalmente gratuita, o una propiedad de una empresa. A la vista de esto, la primera ventaja de las bases de datos de código abierto salta a la vista: evitan tener que hacer frente a un desembolso económico fuerte, puesto que no hace falta comprar su licencia a un fabricante. A no ser que tengas necesidades muy concretas y específicas, pero la proliferación de bases de datos de código abierto en los últimos años, así como de diversos complementos para ellas, hace que resulten cada vez más atractivas incluso para fines muy concretos.

Mientras tanto, la popularidad de los sistemas de bases de datos con licencia comercial, es decir, de las propietarias, ha ido cayendo a lo largo de los últimos años. Y según la lista de sistemas de gestión de bases de datos de DB-Engines, en muchos casos han sido superadas por las bases de datos de código abierto en lo que respecta a la adopción en empresas.

1.2 Ventajas de las Bases de Datos de Código Abierto

Una base de datos de código abierto es un sistema de gestión de bases de datos abierto y de uso gratuito, y quizá una de las pocas desventajas que tiene frente a las bases de datos propietarias es que no cuenta con un servicio de soporte técnico por parte de un proveedor, algo con lo que sí cuentan las bases de datos propietarias. Pero precisamente de esta desventaja nace una de sus ventajas.

Las bases de datos de código abierto cuentan con una nutrida comunidad de usuarios que, en muchos casos no solo comparten sus conocimientos en foros y grupos de usuarios especializados, sino que en muchos casos están dispuestos a ayudar a otros usuarios con problemas y necesidades concretas con ellas.

Además, la información que se guarda en una base de datos de código abierto es de quien la inserta en ella. Sin ningún tipo de restricción. También puedes modificarla para adaptarla a tus necesidades, aunque para ello necesitas contar con ciertos conocimientos de desarrollo. Puedes crear aplicaciones y complementos con base en el Software de código abierto que elijas, de una manera muy parecida a como sucede por ejemplo con el CMS WordPress. Por lo tanto, las bases de datos de código abierto te ofrecen un sinfín de posibilidades de personalización y ampliación.

1.3 Las Principales Bases de Datos de Código Abierto

El número de bases de datos de código abierto crece cada vez más a medida que pasan los años. No obstante, entre todas las posibilidades, hay varias que, según TechTarget, o bien por su sencillez y robustez, o bien por sus posibilidades y facilidades de ampliación y personalización, son las que consiguen una mayor adopción y despliegue en las empresas. Entre ellas están las seis siguientes, quizá las más populares y utilizadas.

MySQL Sin duda esta es la más extendida de las bases de datos de código abierto, y durante muchos años, la opción por defecto para muchos desarrolladores a la hora de optar por una base de datos. Oracle se hizo con ella hace ya bastantes años cuando compró Sun Microsystems, y es un sistema de gestión de bases de datos relacional. Esto quiere decir que utiliza tablas para almacenar los datos, y los tipos de datos almacenados deben estar relacionados entre sí de alguna manera. Cuenta con una edición de

código abierto y algunas versiones de pago, y debido a su popularidad hay muchas herramientas de terceros para ella, así como una cantidad importante de documentación centrada en facilitar su aprendizaje.

La base de MySQL se originó a partir de un sistema de base de datos propietario antiguo, mantener la mayor parte de su compatibilidad y convertir los resultados en código abierto. Sus creadores se centraron en la velocidad, y en la actualidad sigue estando considerada entre las opciones más rápidas de bases de datos. Además, de la velocidad, cuenta con un Script que contribuye a la mejora de la seguridad de las bases de datos de los usuarios. Esta base de datos es compatible, entre otros, con los lenguajes de programación C, C++, Java, Python y Ruby.

No obstante, se trata de una base de datos con licencia dual. Esto implica que algunas de sus funciones y Plugins solo son accesibles en ediciones propietarias de pago. En cuanto a su curva de aprendizaje, no es muy elevada. No es necesario saber mucho SQL para utilizarla, y puedes trabajar con la base de datos desde la línea de comandos. MySQL tiene una compatibilidad elevada prácticamente con cualquier sistema operativo. Se trata de una base de datos sólida, rápida y versátil, por lo que es adecuada para multitud de casos de uso.

MariaDB Cuando Oracle compró Sun, y con ella MySQL, hubo quien no se lo tomó muy bien, porque aunque Oracle ha mantenido MySQL como código abierto, la compañía no tiene mucha reputación que digamos como defensora del Software abierto. Uno de los que no se creyó a Oracle fue uno de los propios fundadores de MySQL, que dejó la compañía para crear MariaDB, un fork de MySQL con una compatibilidad casi completa con ella.

Se puede utilizar como un sustituto prácticamente para todo de MySQL, aunque cuenta con varias funciones únicas que la distinguen de ella. Para empezar, MariaDB usa el motor de almacenamiento Aria para gestionar querys de SQL complejas. Esto hace que la base de datos sea todavía más veloz que MySQL. Además, puedes utilizar filas dinámicas como columnas de tabla, lo que la hace más flexible y adaptable a diversas situaciones.

Además, cuenta con motores de almacenamiento especializados para casos de uso concretos que no tiene MySQL. Por ejemplo, permite implementar el almacenamiento distribuido o las transacciones distribuidas. Como hemos mencionado, puedes utilizar MariaDB en los mismos supuestos y casos que MySQL, pero no hay compatibilidad entre las dos, por lo que tendrás que

elegir cuál de las dos quieres utilizar.

PostgreSQL Esta base de datos relacional y de código abierto es muy utilizada en ciencia de datos y creación de gráficos, y también en el sector de la Inteligencia Artificial. Esto último se debe a que PostgreSQL está especialmente pensada para aplicaciones de Python y Ruby. No obstante, también es compatible con PHP. Se ha desarrollado pensando en la elegancia, y cuenta con ciertas funciones bastante diferenciadas. Entre ellas están la posibilidad de implementar la replicación asíncrona, y el soporte nativo para el almacenamiento de documentos de estilo JSON o XML.

Además, PostgreSQL permite hacer búsquedas completas de texto en la base de datos, y cuenta con varios tipos de datos integrados de gran valor para algunas aplicaciones, como los rangos, los arrays y la geolocalización. Eso sí, no es excesivamente adecuada para aplicaciones que requieran operaciones intensivas de lectura, y la creación de informes de datos con regularidad puede hacer que el almacenamiento de documentos no esté muy fino si tiene que almacenar un conjunto de datos muy grande.

SQLite También relacional, SQLite es una base de datos de código abierto compuesta por una librería ligera y pequeña que ofrece un motor de base de datos. A pesar de su ligereza, permite generar bases de datos de cientos de Terabytes, y un tamaño máximo de filas de un Gigabyte. Incluso con archivos de este tamaño, SQLite sigue siendo rápida. Además, es compatible con Java, Python, C o C++, entre otros lenguajes.

Cuenta con decenas de casos de uso, y por ejemplo, la encontrarán muy útil los desarrolladores de aplicaciones sencillas. Está integrada en smartphones y en muchos ordenadores. También es muy valorada en aplicaciones de Internet de las Cosas y Webs de poco tráfico. Eso sí, por su estructura, no está indicada para sitios con mucho tráfico, ya que su rendimiento se resiente. Además, cuenta con ciertas limitaciones que en algunos casos pueden ser importantes.

Estas hacen que, entre otras cosas, no esté indicada en aplicaciones a las que tienen que acceder varios usuarios con permisos de acceso especiales. SQLite lee y escribe en un archivo de disco común y corriente, por lo que los únicos permisos de acceso aplicables para esta base de datos son los incluidos habitualmente en el sistema operativo.

MongoDB Es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris y es usado en múltiples proyectos o implementaciones en empresas como MTV Network, Craigslist, BCI o Foursquare.

La razón de esto es que MongoDB, al estar escrito en C++, cuenta con una más que notoria capacidad para aprovechar los recursos de la máquina y, al estar licenciado bajo una licencia GNU AGPL 3.0, es posible adaptarlo a nuestras necesidades. Si tuviéramos que resumir a una la principal característica a destacar de MongoDB, sin duda esta sería la velocidad, que alcanza un balance perfecto entre rendimiento y funcionalidad gracias a su sistema de consulta de contenidos. Pero sus características principales no se limitan solo a esto, MongoDB cuenta, además, con otras que lo posicionan como el preferido de muchos desarrolladores.

Redis Esta base de datos de código abierto es distinta a las más utilizadas, ya que se utiliza sobre todo en situaciones completamente distintas a ellas. Se trata de una base de datos que permite estructurar datos como pares de valor clave. Por tanto, se trata de un sistema parecido a los arrays asociativos de PHP o los diccionarios de Python. En realidad, Redis es un sistema para enlazar datos para su referencia, más adelante, con rapidez. Está considerada como la mejor elección como base de datos para cacheado, y también para el trabajo con datos distribuidos.

Para empezar, es una solución de almacenamiento en memoria, enteramente en RAM. Esto quiere decir que sus velocidades de lectura y escritura son muy rápidas. Para aprender sus bases tampoco necesitas mucho tiempo, ya que puedes conocerlas en minutos y empezar con rapidez con el almacenamiento de objetos en ella. Eso sí, puede que no sea muy útil para aplicaciones complejas, pero basta con utilizarla junto con otras bases de datos, como MariaDB, para el resto de la aplicación que se esté desarrollando.

1.4 Agradecimientos

Este texto es una recopilación de múltiples fuentes, nuestra aportación -si es que podemos llamarla así- es plasmarlo en este documento, en el que tratamos de dar coherencia a nuestra visión de los temas desarrollados.

En la realización de este texto se han revisado -en la mayoría de los casos indicamos la referencia, pero pudimos omitir varias de ellas, por lo cual pedimos una disculpa- múltiples páginas Web, artículos técnicos, libros, entre otros materiales bibliográficos, los más representativos y de libre acceso los ponemos a su disposición en la siguiente liga:

Herramientas
http://132.248.181.216/Herramientas/

Además, la documentación y los diferentes ejemplos que se presentan en este trabajo, se encuentran disponibles en dicha liga, para que puedan ser copiados desde el navegador y ser usados. En aras de que el interesado pueda correr dichos ejemplos y afianzar sus conocimientos, además de que puedan ser usados en diferentes ámbitos a los presentados aquí.

Este proyecto fue posible gracias al apoyo recibido por la Facultad de Ciencias de la Universidad Nacional Autónoma de México (UNAM) y al tiempo robado a nuestras actividades académicas, principalmente durante el período de confinamiento de los años 2020 a 2022.

2 Bases de Datos

¿Qué son las bases de datos? una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su uso posterior (véase [12] y [13]). En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente la mayoría de las bases de datos están en formato digital, se han desarrollado y se ofrece un amplio rango de soluciones al problema del almacenamiento de datos.

2.1 Clasificación

Las bases de datos se pueden clasificar de diferentes formas: acuerdo a la ubicación, de acuerdo a la forma de establecer relaciones entre los datos, según la orientación, entre otros.

De acuerdo a la ubicación

- Base de datos centralizada: Se ubica, almacena y mantiene en una sola ubicación. Eso no significa que el usuario tenga que estar en el mismo lugar para acceder.
- Base de datos distribuida: Se trata en realidad de diferentes bases de datos en distintas locaciones físicas unidas por un gestor que las hace funcionar como si fueran una sola.

De acuerdo a la forma de establecer relaciones entre los datos

- Relacionales: En este tipo de base de datos estos se organizan en forma de filas y columnas.
- Orientadas a objetos: Aquí los datos se almacenan en forma de objetos. Estos objetos se guardan en la base de datos asignándoles atributos y métodos que son los que definen qué hacer con los datos.
- Orientadas a grafos: Utiliza la teoría de grafos para almacenar, mapear y consultar las relaciones entre datos.
- NoSQL: Permite almacenar datos no estructurados o semiestructurados.

- Orientada a documentos: Es un subtipo de la anterior. En lugar de almacenar los datos en filas y columnas utiliza documentos para almacenar y recuperar los datos. Estos documentos organizan los datos utilizando estándares como JSON o XML.

Según la orientación

- OLTP: Son bases de datos orientadas al procesamiento de transacciones e incluye funciones de introducción, modificación y borrado de datos.
- OLAP: Estas bases de datos están orientadas al análisis de los datos para permitir extraer conclusiones.

Otros tipos

- Autónomas: Están basadas en la nube y utilizan el aprendizaje automático para automatizar el trabajo la base de datos, la seguridad, las copias de seguridad, las actualizaciones y otras tareas de gestión rutinarias que en las bases de datos tradicionales realiza un administrador.
- Almacén de datos: Es una base de datos enfocada en el sector corporativo que integra y depura información de varias fuentes distintas para procesarla y analizarla desde diferentes puntos de vista a gran velocidad.

Hay programas denominados sistemas gestores de bases de datos SGBD (Database Management System o DBMS), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Actualmente se tienen dos importantes tipos de bases de datos:

- Structured Query Language SQL (lenguaje de consulta estructurada) es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.
- NoSQL "no solo SQL" es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas.

2.2 ¿Qué son las bases de datos SQL?

el lenguaje de consulta estructurada SQL (Structured Query Language), es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. Una de sus principales características es el manejo del álgebra y el cálculo relacional para efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como realizar cambios en ellas.

Originalmente basado en el álgebra relacional y en el cálculo relacional, SQL consiste en un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de control de datos. El alcance de SQL incluye la inserción de datos, consultas, actualizaciones y borrado, la creación y modificación de esquemas y el control de acceso a los datos. También en SQL a veces se describe como un lenguaje declarativo, también incluye elementos procesales.

Características generales de SQL SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones.

Es un lenguaje declarativo de "alto nivel" o "de no procedimiento" que, gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros –y no a registros individuales– permite una alta productividad en codificación y la orientación a objetos. De esta forma, una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros. SQL también tiene las siguientes características:

- Lenguaje de definición de datos: El LDD de SQL proporciona comandos para la definición de esquemas de relación, borrado de relaciones y modificaciones de los esquemas de relación.
- Lenguaje interactivo de manipulación de datos: El LMD de SQL incluye lenguajes de consultas basado tanto en álgebra relacional como en cálculo relacional de tuplas.
- Integridad: El LDD de SQL incluye comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos.

- Definición de vistas: El LDD incluye comandos para definir las vistas.
- Control de transacciones: SQL tiene comandos para especificar el comienzo y el final de una transacción.
- SQL incorporado y dinámico: Esto quiere decir que se pueden incorporar instrucciones de SQL en lenguajes de programación como: C++, C, Java, Python, PHP, COBOL, Pascal y Fortran.
- Autorización: El LDD incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas.

Tipos de datos Algunos de los tipos de datos básicos de SQL son:

- BINARY (1 Byte), para consultas sobre tabla adjunta de productos de bases de datos que definen un tipo de datos Binario.
- BIT (1 Byte), valores Si/No ó True/False
- BYTE (1 Byte), un valor entero entre 0 y 255.
- COUNTER (4 Bytes), un número incrementado automáticamente (de tipo Long)
- CURRENCY (8 Bytes), un entero escalable entre -922,337,203,685,477.5808 y 922,337,203,685,477.5807.
- DATETIME (8 Bytes), un valor de fecha u hora entre los años 100 y 9999.
- SINGLE (4 Bytes), un valor en punto flotante de precisión simple con un rango de $-3.402823*10^{38}$ a $-1.401298*10^{-45}$ para valores negativos, $1.401298 * 10^{-45}$ a $3.402823 * 10^{38}$ para valores positivos, y 0.
- DOUBLE (8 Bytes), un valor en punto flotante de doble precisión con un rango de $-1.79769313486232 * 10^{308}$ a $-4.94065645841247 * 10^{-324}$ para valores negativos, $4.94065645841247 * 10^{-324}$ a $1.79769313486232 * 10^{308}$ para valores positivos, y 0.
- SHORT (2 Bytes), un entero corto entre -32,768 y 32,767.
- LONG (4 Bytes), un entero largo entre -2,147,483,648 y 2,147,483,647.

- LONGTEXT (1 Byte por carácter), de cero a un máximo de 1.2 gigabytes.
- LONGBINARY (según se necesite), de cero 1 gigabyte. Utilizado para objetos OLE.
- TEXT (1 Byte por carácter), de cero a 255 caracteres.

También se tienen definidos múltiples sinónimos de los tipos de datos definidos:

- BINARY: VARBINARY
- BIT: BOOLEAN, LOGICAL, LOGICAL1, YESNO
- BYTE: INTEGER1
- COUNTER: AUTOINCREMENT
- CURRENCY: MONEY
- DATETIME: DATE, TIME, TIMESTAMP
- SINGLE: FLOAT4, IEEESINGLE, REAL
- DOUBLE: FLOAT, FLOAT8, IEEEDOUBLE, NUMBER, NUMERIC
- SHORT: INTEGER2, SMALLINT
- LONG: INT, INTEGER, INTEGER4
- LONGBINARY: GENERAL, OLEOBJECT
- LONGTEXT: LONGCHAR, MEMO, NOTE
- TEXT: ALPHANUMERIC, CHAR - CHARACTER, STRING - VARCHAR

SQL no requiere que escribamos todas las palabras clave en mayúscula, pero por convención ayuda a las personas a distinguir las palabras clave de SQL de los nombres de las columnas y las tablas.

Optimización Como ya se dijo antes, y suele ser común en los lenguajes de acceso a bases de datos de alto nivel, SQL es un lenguaje declarativo. O sea, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución.

El orden de ejecución interno de una sentencia puede afectar seriamente a la eficiencia del SGBD, por lo que se hace necesario que este lleve a cabo una optimización antes de su ejecución. Muchas veces, el uso de índices acelera una instrucción de consulta, pero ralentiza la actualización de los datos. Dependiendo del uso de la aplicación, se priorizará el acceso indexado o una rápida actualización de la información. La optimización difiere sensiblemente en cada motor de base de datos y depende de muchos factores.

Los sistemas de bases de datos modernos poseen un componente llamado optimizador de consultas. Este realiza un análisis detallado de los posibles planes de ejecución de una consulta SQL y elige aquel que sea más eficiente para llevar adelante la misma.

Existe una ampliación de SQL conocida como FSQL (Fuzzy SQL, SQL difuso) que permite el acceso a bases de datos difusas, usando la lógica difusa.

2.2.1 Lenguaje de definición de datos (DDL)

El lenguaje de definición de datos (en inglés Data Definition Language, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

CREATE (Crear) este comando permite crear objetos de datos, como nuevas bases de datos, tablas, vistas y procedimientos almacenados.

Ejemplos (crear una tabla):

```
CREATE TABLE clientes;
CREATE TABLE personas(
    IDPersonas INT NOT NULL,
    Nombre VARCHAR(50) NOT NULL,
    Apellido VARCHAR(60) NOT NULL,
UNIQUE(IDPersonas)
);
CREATE TABLE pedidos(
```

```
    OrdenID INT NOT NULL,
    NumeroOrden INT NOT NULL,
    IDPersona INT,
    PRIMARY KEY(OrdenID),
    FOREING KEY(IDPersona) REFERENCES personas(IDPersonas)
);
```

ALTER (Alterar) este comando permite modificar la estructura de una tabla u objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un *trigger*, etc.

Ejemplo (agregar columna a una tabla):

```
ALTER TABLE personas ADD edad INT UNSIGNED;
```

DROP (Eliminar) este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

Ejemplo (elimina la tabla personas):

```
DROP TABLE personas;
```

TRUNCATE (Truncar) este comando solo aplica a tablas y su función es borrar el contenido completo de la tabla especificada. La ventaja sobre el comando DELETE, es que si se quiere borrar todo el contenido de la tabla, es mucho más rápido, especialmente si la tabla es muy grande. La desventaja es que TRUNCATE solo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula WHERE. Si bien, en un principio, esta sentencia parecería ser DML (Lenguaje de Mani-pulación de Datos), es en realidad una DDL, ya que internamente, el comando TRUNCATE borra la tabla y la vuelve a crear y no ejecuta ninguna transacción.

Ejemplo (borra el contenido de la tabla personas):

```
TRUNCATE TABLE personas;
```

CREATE INDEX permite crear índices (ya sea únicos o permite valores duplicados) en una o más columnas.

Ejemplo (crea un índice admitiendo valores duplicados en la columna)

```
CREATE INDEX indicePersonas ON personas(personas);
```

Ejemplo (crear un índice sin que puedan existir valores duplicados)

```
CREATE UNIQUE INDEX indicePersonas ON personas(personas);
```

2.2.2 Lenguaje de manipulación de datos DML

Un lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

El lenguaje de manipulación de datos más popular hoy en día es SQL, es usado para recuperar y manipular datos en una base de datos relacional.

SELECT (Seleccionar) La sentencia SELECT nos permite consultar los datos almacenados en una tabla de la base de datos.

Forma básica:

```
SELECT [{ALL|DISTINCT}]
<nombre_campo>[, <nombre_campo>...]
FROM {<nombre_tabla>|<nombre_vista>}[,
{<nombre_tabla>|<nombre_vista>}...]
[WHERE <condición> [{AND|OR} <condición>...]]
[GROUP BY <nombre_campo>[, <nombre_campo>...]]
[HAVING <condición> [{AND|OR} <condición>...]]
[ORDER BY {<nombre_campo>|<indice_campo>}
[{ASC|DESC}][,{<nombre_campo>|<indice_campo>}
[{ASC|DESC}]]];
```

SELECT palabra clave que indica que la sentencia de SQL que queremos ejecutar es de selección.

ALL indica que queremos seleccionar todos los valores. Es el valor por defecto y no suele especificarse casi nunca.

DISTINCT indica que queremos seleccionar solo los valores distintos.

FROM indica la tabla (o tablas) desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se denomina a la consulta "consulta combinada" o "join". En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula WHERE.

WHERE especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admite los operadores lógicos AND y OR.

GROUP BY especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas.

HAVING especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Su funcionamiento es similar al de WHERE pero aplicado al conjunto de resultados devueltos por la consulta. Debe aplicarse siempre junto a GROUP BY y la condición debe estar referida a los campos contenidos en ella.

ORDER BY presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC (orden ascendente) y DESC (orden descendente). El valor predeterminado es ASC.

Ejemplo:

Para formular una consulta a la tabla coches, usamos la palabra clave FROM y con SELECT le indicamos que deseamos recuperar los campos *matrícula, marca, modelo, color, número_kilómetros, num_plazas*, para ello debemos ejecutar la siguiente consulta:

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
ORDER BY
    marca, modelo;
```

los datos serán devueltos ordenados por marca y por modelo en orden ascendente, de menor a mayor.

Ejemplo de consulta simplificada a través de un comodín de campos (*): El uso del asterisco indica que queremos que la consulta devuelva todos los campos que existen en la tabla y los datos serán devueltos ordenados por marca y por modelo.

```
SELECT *
FROM
    coches
ORDER BY
    marca, modelo;
```

Ejemplo de consulta simplificada a través de un comodín de campos (*): El uso del asterisco indica que queremos que la consulta devuelva todos los campos que existen en la tabla y los datos serán devueltos ordenados por marca en forma descendente y por modelo en forma ascendente.

```
SELECT *
FROM
    coches
ORDER BY
    marca DESC, modelo ASC;
```

Cláusula WHERE (Donde) la cláusula WHERE es la instrucción que nos permite filtrar el resultado de una sentencia SELECT. Habitualmente no deseamos obtener toda la información existente en la tabla, sino que queremos obtener sólo la información que nos resulte útil en ese momento. La cláusula WHERE filtra los datos antes de ser devueltos por la consulta. Cuando en la cláusula WHERE queremos incluir un tipo texto, debemos incluir el valor entre comillas simples.

Ejemplos:

En nuestro ejemplo, se desea consultar un coche en concreto, para esto se agregó una cláusula WHERE. Esta cláusula especifica una o varias condiciones que deben cumplirse para que la sentencia SELECT devuelva los datos. En este caso la consulta devolverá sólo los datos del coche con matrícula para que la consulta devuelva sólo los datos del coche con matrícula MF-234-ZD o bien la matrícula FK-938-ZL . Se puede utilizar la cláusula WHERE solamente, ó en combinación con tantas condiciones como queramos.

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
WHERE
matricula = 'MF-234-ZD' OR matricula = 'FK-938-ZL';
```

Una condición WHERE puede ser negada a través del Operador Lógico NOT. La siguiente consulta devolverá todos los datos de la tabla coches, menos el que tenga la matrícula MF-234-ZD.

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
WHERE
    NOT matricula = 'MF-234-ZD';
```

La siguiente consulta utiliza la condicional DISTINCT, la cual nos devolverá todos los valores distintos formados por los campos *marca* y *modelo* de la tabla coches.

```
SELECT DISTINCT marca, modelo FROM coches;
```

Cláusula ORDER BY (Ordernar Por) la cláusula ORDER BY es la instrucción que nos permite especificar el orden en el que serán devueltos los datos. Podemos especificar el orden de forma ascendente o descendente a través de las palabras clave ASC y DESC. El orden depende del tipo de datos que estén definidos en la columna, de forma que un campo numérico será ordenado como tal, y un alfanumérico se ordenará de la A a la Z, aunque su contenido sea numérico. El valor predeterminado es ASC si no se especifica al hacer la consulta.

Ejemplos:

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
ORDER BY
    marca ASC, modelo DESC;
```

Este ejemplo, selecciona todos los campos *matricula, marca, modelo, color, numero_kilometros y num_plazas* de la tabla coches, ordenándolos por los campos *marca* y *modelo*, *marca* en forma ascendente y *modelo* en forma descendente.

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
ORDER BY 2;
```

Este ejemplo, selecciona todos los campos *matrícula, marca, modelo, color, numero_kilometros y num_plazas* de la tabla coches, ordenándolos por el campo *marca*, ya que aparece en segundo lugar dentro de la lista de campos que componen la cláusula SELECT.

Subconsultas una subconsulta es una sentencia SELECT que está embebida en una cláusula de otra sentencia SQL. También pueden utilizarse subconsultas en los comandos INSERT, UPDATE, DELETE y en la cláusula FROM.

Las subconsultas pueden resultar útiles si necesitas seleccionar filas de una tabla con una condición que depende de los datos de la propia tabla o de otra tabla.

La subconsulta (consulta interna), se ejecuta antes de la consulta principal; el resultado de la subconsulta es utilizado por la consulta principal (consulta externa).

```
SELECT
    c.matricula, c.modelo
FROM
    coches AS c
WHERE c.matricula IN
(
    SELECT m.matricula
    FROM multas AS m
    WHERE m.importe > 100
);
```

En este ejemplo, se seleccionan las matrículas y los modelos de los coches cuyas multas superan los USD 100.

INSERT (Insertar) una sentencia INSERT de SQL agrega uno o más registros a una (y solo una) tabla en una base de datos relacional.

Forma básica:

```
INSERT INTO
tablatura(columnaA, [columnaB, ...])
VALUES
('valor1', ['valor2', ...]);
```

o también se puede utilizar como:

```
INSERT INTO tablatura VALUES ('valor1', 'valor2');
```

Las cantidades de columnas y valores deben ser iguales. Si una columna no se especifica, le será asignado el valor por omisión. Los valores especificados (o implícitos) por la sentencia INSERT deberán satisfacer todas las restricciones aplicables. Si ocurre un error de sintaxis o si alguna de las restricciones es violada, no se agrega la fila y se devuelve un error.

Ejemplo:

```
INSERT INTO agenda_telefonica (nombre, numero)
VALUES ('Roberto Jeldrez', 4886850);
```

Cuando se especifican todos los valores de una tabla, se puede utilizar la sentencia acortada:

```
INSERT INTO nombre_tabla VALUES ('valor1', ['valor2',
...]);
```

Ejemplo (asumiendo que 'nombre' y 'número' son las únicas columnas de la tabla 'agenda_telefonica'):

```
INSERT INTO agenda_telefonica
VALUES ('Jhonny Aguilar', 080473968);
```

Formas avanzadas una característica de SQL (desde SQL-92) es el uso de constructores de filas para insertar múltiples filas a la vez, con una sola sentencia SQL:

```
INSERT INTO
tabla(columna1[, columna2, ...])
VALUES
('valor1A', ['valor1B', ...]),
('value2A', ['value2B', ...]), ...;
```

Esta característica es soportada por DB2, PostgreSQL (desde la versión 8.2), MySQL, y H2.

Ejemplo (asumiendo que nombre y número son las únicas columnas en la tabla agenda_telefonica):

```
INSERT INTO
    agenda_telefonica
VALUES
    ('Roberto Fernández', '4886850'),
    ('Alejandro Sosa', '4556550');
```

Que podía haber sido realizado por las sentencias:

```
INSERT INTO agenda_telefonica VALUES ('Roberto Fernández', '4886850');
INSERT INTO agenda_telefonica VALUES ('Alejandro Sosa', '4556550');
```

Notar que las sentencias separadas pueden tener semántica diferente (especialmente con respecto a los triggers), y puede tener diferente rendimiento que la sentencia de inserción múltiple.

Para insertar varias filas en MS SQL puede utilizar esta construcción:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212'
UNION ALL
SELECT 'Peter Doe', '555-2323';
```

Tenga en cuenta que no se trata de una sentencia SQL válida de acuerdo con el estándar SQL (SQL: 2003), debido a la cláusula subselect incompleta.

Para hacer lo mismo en Oracle se usa la Tabla DUAL, siempre que se trate de solo una fila simple:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM DUAL
UNION ALL
SELECT 'Peter Doe','555-2323' FROM DUAL
```

Una implementación conforme al estándar de esta lógica se muestra en el siguiente ejemplo, o como se muestra arriba (no aplica en Oracle):

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM LATERAL ( VALUES (1) ) AS t(c)
UNION ALL
SELECT 'Peter Doe','555-2323' FROM LATERAL ( VALUES (1) ) AS t(c)
```

Copia de filas de otras tablas un INSERT también puede utilizarse para recuperar datos de otros, modificarla si es necesario e insertarla directamente en la tabla. Todo esto se hace en una sola sentencia SQL que no implica ningún procesamiento intermedio en la aplicación cliente. Un SUBSELECT se utiliza en lugar de la cláusula VALUES. El SUBSELECT puede contener la sentencia JOIN, llamadas a funciones, y puede incluso consultar en la misma TABLA los datos que se insertan. Lógicamente, el SELECT se evalúa antes de que la operación INSERT esté iniciada. Un ejemplo se da a continuación:

```
INSERT INTO phone_book2
SELECT *
FROM phone_book
WHERE name IN ('John Doe', 'Peter Doe');
```

Una variación es necesaria cuando algunos de los datos de la tabla fuente se está insertando en la nueva tabla, pero no todo el registro. (O cuando los esquemas de las tablas no son iguales.)

```
INSERT INTO phone_book2 ([name], [phoneNumber])
SELECT [name], [phoneNumber]
FROM phone_book
WHERE name IN ('John Doe', 'Peter Doe');
```

El SELECT produce una tabla (temporal), y el esquema de la tabla temporal debe coincidir con el esquema de la tabla donde los datos son insertados.

UPDATE (Actualizar) una sentencia UPDATE de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla.
Ejemplo:

```
UPDATE My_table SET field1 = 'updated value' WHERE
field2 = 'N';
```

DELETE (Borrar) una sentencia DELETE de SQL borra uno o más registros existentes en una tabla.
Forma básica:
DELETE FROM tabla WHERE columna1 = 'valor1';
Ejemplo:

```
DELETE FROM mi_tabla WHERE columna2 = 'N';
```

Funciones Disponemos de varias funciones que podemos usar, por ejemplo la función MIN(), devuelve el valor más pequeño de la columna seleccionada:

```
SELECT MIN(columna)
FROM tabla
WHERE condición;
```

La función MAX() devuelve el valor más grande de la columna seleccionada:

```
SELECT MAX(columna)
FROM tabla
WHERE condición;
```

La función COUNT() devuelve el número de filas de la consulta, es decir, el número de registros que cumplen una determinada condición:

```
SELECT COUNT(columna)
FROM tabla
WHERE condición;
```

La función AVG() devuelve el valor promedio de una columna especifica (numérica):

```
SELECT AVG(columna)
FROM tabla
WHERE condición;
```

La función SUM() devuelve la suma total de una columna especifica (numérica):

```
SELECT SUM(columna)
FROM tabla
WHERE condición;
```

Caracteres Comodín Los comodines son caracteres especiales que se utilizan con las palabras clave LIKE y NOT LIKE. Esto permite buscar datos con patrones sofisticados de manera bastante eficiente.

- % Equivale a cero o más caracteres

```
SELECT * FROM clientes
WHERE nombre LIKE '%ory';
```

- _ Equivale a cualquier carácter individual

```
SELECT * FROM clientes
WHERE ciudad LIKE '___vale';
```

- [listacaracteres] Equivale a cualquier carácter en la lista

```
SELECT * clientes
WHERE nombre LIKE '[JKT]%';
```

Claves En las bases de datos relacionales, existe un concepto de claves primarias y externas . En las tablas SQL, estos se incluyen como restricciones, donde una tabla puede tener una clave principal, una clave externa o ambas.

Claves Primarias una clave primaria permite que cada registro de una tabla se identifique de forma única. Solo puede tener una clave principal por tabla y puede asignar esta restricción a cualquier columna única o combinación. Sin embargo, esto significa que cada valor dentro de esta (s) columna (s) debe ser único.

Por lo general, en una tabla, la columna de ID es una clave principal y, por lo general, está emparejada con la palabra clave AUTO_INCREMENT. Esto significa que el valor aumenta automáticamente a medida que se crean nuevos registros.

Ejemplo: Cree una nueva tabla y establezca la clave principal en la columna ID.

```
CREATE TABLE clientes (
    id int NOT NULL AUTO_INCREMENT,
    Nombre varchar(255),
    Apellido varchar(255) NOT NULL,
    Direccion varchar(255),
    Email varchar(255),
    PRIMARY KEY (id)
);
```

Claves Externas se puede aplicar una clave externa a una columna o a muchas. Se usa para vincular 2 tablas juntas en una base de datos relacional.

- La tabla que contiene la clave externa se llama clave secundaria ,
- La tabla que contiene la clave a la que se hace referencia (o candidata) se denomina tabla principal .

Básicamente, esto significa que los datos de la columna se comparten entre 2 tablas, porque una clave externa también evita que se inserten datos no válidos que no están presentes en la tabla principal.

Ejemplo: Cree una nueva tabla y convierta cualquier columna que haga referencia a ID en otras tablas en claves externas.

```
CREATE TABLE ordenes (
id int NOT NULL,
usuario_id int,
```

```
producto_id int,
PRIMARY KEY (id),
FOREIGN KEY (usuario_id) REFERENCES users(id),
FOREIGN KEY (producto_id) REFERENCES products(id)
);
```

Índices Los índices son atributos que se pueden asignar a columnas en las que se buscan con frecuencia para que la recuperación de datos sea un proceso más rápido y eficiente.

```
CREATE INDEX idx_test
ON users (first_name, surname);
CREATE UNIQUE INDEX idx_test
ON users (first_name, surname);
DROP INDEX
ALTER TABLE users
DROP INDEX idx_test;
```

Uniones En SQL, JOINse utiliza una cláusula para devolver un resultado que combina datos de varias tablas, basándose en una columna común que aparece en ambas.

Hay varias combinaciones diferentes disponibles para su uso:

- Inner Join (predeterminado): Devuelve cualquier registro que tenga valores coincidentes en ambas tablas.
- Left Join: Devuelve todos los registros de la primera tabla, junto con los registros coincidentes de la segunda tabla.
- Right Join: Devuelve todos los registros de la segunda tabla, junto con los registros coincidentes de la primera.
- Full Join: Devuelve todos los registros de ambas tablas cuando hay una coincidencia.

```
SELECT orders.id, users.FirstName, users.Surname, products.name
as 'product name'
FROM orders
INNER JOIN users on orders.user_id = users.id
INNER JOIN products on orders.product_id = products.id;
```

Vistas en SQL Una vista es esencialmente un conjunto de resultados de SQL que se almacena en la base de datos bajo una etiqueta, por lo que puede volver a él más tarde sin tener que volver a ejecutar la consulta. Estos son especialmente útiles cuando tiene una consulta SQL costosa que puede necesitar varias veces. Entonces, en lugar de ejecutarlo una y otra vez para generar el mismo conjunto de resultados, puede hacerlo una vez y guardarlo como una vista.

Cómo crear vistas para crear una vista, puede hacerlo así:

```
CREATE VIEW priority_users AS
SELECT * FROM users
WHERE country = 'United Kingdom';
```

Luego, en el futuro, si necesita acceder al conjunto de resultados almacenado, puede hacerlo así:

```
SELECT * FROM [priority_users];
```

Cómo reemplazar vistas con el CREATE OR REPLACE comando, podemos actualizar una vista como esta:

```
CREATE OR REPLACE VIEW [priority_users] AS
SELECT * FROM users
WHERE country = 'United Kingdom' OR country='USA';
```

Cómo eliminar vistas para eliminar una vista, simplemente usamos DROP VIEW comando.

```
DROP VIEW priority_users;
```

Recuperación de Clave Los diseñadores de base de datos que usan una clave suplente como la clave principal para cada tabla, se ejecutará en el escenario ocasional en el que es necesario recuperar automáticamente la base de datos, generando una clave primaria de una sentencia SQL INSERT para su uso en otras sentencias SQL. La mayoría de los sistemas no permiten sentencias SQL INSERT para retornar la fila de datos. Por lo tanto, se hace necesario aplicar una solución en tales escenarios.

Implementaciones comunes incluyen:

- Utilizando un procedimiento almacenado específico de base de datos que genera la clave suplente, realice la operación INSERT y finalmente devuelva la clave generada.
- Utilizando una sentencia SELECT específica de base de datos, sobre una tabla temporal que contiene la última fila insertada. DB2 implementa esta característica de la siguiente manera:

```
SELECT *
FROM NEW TABLE (
INSERT INTO phone_book
VALUES ('Cristobal Jeldrez','0426.817.10.30')
) AS t
```

- Utilizando una sentencia SELECT después de la sentencia INSERT con función específica de base de datos, que devuelve la clave primaria generada por el registro insertado más recientemente.
- Utilizando una combinación única de elementos del original SQL INSERT en una sentencia posterior SELECT.
- Utilizando un GUID en la sentencia SQL INSERT y la recupera en una sentencia SELECT.
- Utilizando la función de PHP mysql_insert_id() de MySQL después de la sentencia INSERT.
- Utilizando un INSERT con la cláusula RETURNING para Oracle, que solo se puede utilizar dentro de un bloque PL/SQL, en el caso de PostgreSQL se puede usar tanto con SQL como con PL/SQL.

```
INSERT INTO phone_book VALUES ('Cristobal Jeldrez',
'0426.817.10.30')
RETURNING phone_book_id INTO v_pb_id
```

- En el caso de MS SQL se puede utilizar la siguiente instrucción:

```
Set NoCount On;
INSERT INTO phone_book VALUES ('Cristobal Jeldrez',
'0426.817.10.30');
Select @@Identity as id
```

2.3 ¿Qué son las bases de datos NoSQL?

A veces llamado "no solo SQL" es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad) y habitualmente escalan bien horizontalmente. Los sistemas NoSQL se denominan a veces "no solo SQL" para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

Por lo general, los investigadores académicos se refieren a este tipo de bases de datos como almacenamiento estructurado, término que abarca también las bases de datos relacionales clásicas. A menudo, las bases de datos NoSQL se clasifican según su forma de almacenar los datos, y comprenden categorías como clave-valor, las implementaciones de BigTable, bases de datos documentales, y bases de datos orientadas a grafos.

Las bases de datos NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala. Bastantes sistemas NoSQL emplean una arquitectura distribuida, manteniendo los datos de forma redundante en varios servidores, usando frecuentemente una tabla Hash distribuida. De esta forma, el sistema puede realmente escalar añadiendo más servidores, y el fallo en un servidor puede ser tolerado.

¿Cómo funciona una base de datos NoSQL (no relacionales)? Las bases de datos NoSQL utilizan una variedad de modelos de datos para acceder y administrar datos. Estos tipos de bases de datos están optimizados específicamente para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles, lo que se logra mediante la flexibilización de algunas de las restricciones de coherencia de datos en otras bases de datos.

Considere el ejemplo de modelado del esquema para una base de datos simple de libros:

- En una base de datos relacional, un registro de libros a menudo se enmascara (o "normaliza") y se almacena en tablas separadas, y las relaciones se definen mediante restricciones de claves primarias y externas. En este ejemplo, la tabla *Libros* tiene las columnas *ISBN, Título*

del libro y Número de edición, la tabla *Autores* tiene las columnas *IDAutor y Nombre de autor* y, finalmente, la tabla *Autor-ISBN* tiene las columnas *IDAutor e ISBN*. El modelo relacional está diseñado para permitir que la base de datos aplique la integridad referencial entre tablas en la base de datos, normalizada para reducir la redundancia y, generalmente, está optimizada para el almacenamiento.

- En una base de datos NoSQL, el registro de un libro generalmente se almacena como un documento JSON. Para cada libro, el elemento *ISBN, Título del libro, Número de edición, Nombre autor y IDAutor* se almacenan como atributos en un solo documento. En este modelo, los datos están optimizados para un desarrollo intuitivo y escalabilidad horizontal.

¿Por qué debería usar una base de datos NoSQL? Las bases de datos NoSQL se adaptan perfectamente a muchas aplicaciones modernas, como dispositivos móviles, Web y juegos, que requieren bases de datos flexibles, escalables, de alto rendimiento y altamente funcionales para proporcionar excelentes experiencias de usuario.

- Flexibilidad: las bases de datos NoSQL generalmente ofrecen esquemas flexibles que permiten un desarrollo más rápido y más iterativo. El modelo de datos flexible hace que las bases de datos NoSQL sean ideales para datos semiestructurados y no estructurados.

- Escalabilidad: las bases de datos NoSQL generalmente están diseñadas para escalar usando clústeres distribuidos de Hardware en lugar de escalar añadiendo servidores caros y sólidos. Algunos proveedores de la nube manejan estas operaciones en segundo plano, como un servicio completamente administrado.

- Alto rendimiento: la base de datos NoSQL está optimizada para modelos de datos específicos y patrones de acceso que permiten un mayor rendimiento que el intento de lograr una funcionalidad similar con bases de datos relacionales.

- Altamente funcional: las bases de datos NoSQL proporcionan APIs altamente funcionales y tipos de datos que están diseñados específicamente para cada uno de sus respectivos modelos de datos.

Algunos ejemplos de bases de datos NoSQL son: DynamoDB, Cassandra, Redis, CouchDB, RethinkDB, RavenDB, Elasticsearch y MongoDB.

Tipos de bases de datos NoSQL

- Clave-valor: las bases de datos clave-valor son altamente divisibles y permiten escalado horizontal a escalas que otros tipos de bases de datos no pueden alcanzar. Los casos de uso como juegos, tecnología publicitaria e IoT se prestan particularmente bien con el modelo de datos clave-valor. Amazon DynamoDB está diseñado para proporcionar una latencia de milisegundos constante de un solo dígito para cualquier escala de cargas de trabajo. Este rendimiento sistemático es uno de los principales elementos que explican por qué la característica de historias de Snapchat, que incluye la carga de trabajo de escritura de almacenamiento más grande de Snapchat, se trasladó a DynamoDB.

- Documentos: en el código de aplicación, los datos se representan a menudo como un objeto o un documento de tipo JSON porque es un modelo de datos eficiente e intuitivo para los desarrolladores. Las bases de datos de documentos facilitan a los desarrolladores el almacenamiento y la consulta de datos en una base de datos mediante el uso del mismo formato de modelo de documento que emplean en el código de aplicación. La naturaleza flexible, semiestructurada y jerárquica de los documentos y las bases de datos de documentos permiten que evolucionen según las necesidades de las aplicaciones. El modelo de documentos funciona bien con catálogos, perfiles de usuario y sistemas de administración de contenido en los que cada documento es único y evoluciona con el tiempo. Amazon DocumentDB (con compatibilidad para MongoDB) y MongoDB son bases de datos de documentos conocidas que proporcionan APIs poderosas e intuitivas para un desarrollo flexible e iterativo.

- Gráficos: el propósito de una base de datos de gráficos es facilitar la creación y la ejecución de aplicaciones que funcionan con conjuntos de datos altamente conectados. Los casos de uso típicos para una base de datos de gráficos incluyen redes sociales, motores de recomendaciones, detección de fraude y gráficos de conocimiento. Amazon Neptune es un servicio de base de datos de gráficos completamente administrado. Neptune admite tanto el modelo de Property Graph como el Resource

Description Framework (RDF), que ofrece la opción de dos API de gráficos: TinkerPop y RDF/SPARQL. Las bases de datos de gráficos populares incluyen Neo4j y Giraph.

- En memoria: las aplicaciones de juegos y tecnología publicitaria tienen casos de uso como tablas de clasificación, tiendas de sesión y análisis en tiempo real que requieren tiempos de respuesta de microsegundos y pueden tener grandes picos de tráfico en cualquier momento. Amazon ElastiCache ofrece MemCachéd y Redis, para servir cargas de trabajo de baja latencia y alto rendimiento, como McDonald's, en las que no se pueden servirse con almacenes de datos basados en disco. Amazon DynamoDB Accelerator (DAX) es otro ejemplo de un almacén de datos especialmente diseñado. DAX hace que DynamoDB lea una orden de magnitud más rápida.

- Buscar: muchas aplicaciones generan registros para ayudar a los desarrolladores a solucionar problemas. Amazon Elasticsearch Service (Amazon ES) está diseñado para proporcionar visualizaciones en tiempo real y análisis de datos generados por máquinas al indexar, agregar y buscar registros y métricas semiestructuradas. Amazon ES también es un poderoso motor de búsqueda de alto rendimiento para casos de uso de búsqueda de texto completo. Expedia está utilizando más de 150 dominios de Amazon ES, 30 TB de datos y 30 mil millones de documentos para una variedad de casos de uso críticos, que van desde el monitoreo operativo y la resolución de problemas, hasta el seguimiento de la pila de aplicaciones distribuidas y la optimización de precios.

2.3.1 SQL en comparación con NoSQL

Durante décadas, el modelo de datos predominante utilizado para el desarrollo de aplicaciones era el modelo de datos relacional empleado por bases de datos relacionales como Oracle, DB2, SQL Server, MySQL y PostgreSQL. No fue sino hasta mediados y finales de la década del 2000 que otros modelos de datos comenzaron a adoptarse y aumentó su uso significativamente. Para diferenciar y categorizar estas nuevas clases de bases de datos y modelos de datos, se acuñó el término "NoSQL". Con frecuencia, los términos "NoSQL" y "no relacional" se usan indistintamente.

Cargas de trabajo óptimas las bases de datos relacionales están diseñadas para aplicaciones de procesamiento de transacciones online (OLTP) altamente coherentes y transaccionales, y son buenas para el procesamiento analítico online (OLAP), en cambio las bases de datos NoSQL están diseñadas para varios patrones de acceso a datos que incluyen aplicaciones de baja latencia. Las bases de datos de búsqueda NoSQL están diseñadas para hacer análisis sobre datos semiestructurados.

Modelo de datos el modelo relacional normaliza los datos en tablas conformadas por filas y columnas. Un esquema define estrictamente las tablas, las filas, las columnas, los índices, las relaciones entre las tablas y otros elementos de las bases de datos. La base de datos impone la integridad referencial en las relaciones entre tablas, en cambio las bases de datos NoSQL proporcionan una variedad de modelos de datos, como clave-valor, documentos y gráficos, que están optimizados para el rendimiento y la escala.

Propiedades ACID las bases de datos relacionales ofrecen propiedades de atomicidad, coherencia, aislamiento y durabilidad (ACID):

- La atomicidad requiere que una transacción se ejecute por completo o no se ejecute en absoluto.
- La coherencia requiere que una vez confirmada una transacción, los datos deben acoplarse al esquema de la base de datos.
- El aislamiento requiere que las transacciones simultáneas se ejecuten por separado.
- La durabilidad requiere la capacidad de recuperarse de un error inesperado del sistema o de un corte de energía y volver al último estado conocido.

Las bases de datos NoSQL a menudo hacen concesiones al flexibilizar algunas de las propiedades ACID de las bases de datos relacionales para un modelo de datos más flexible que puede escalar horizontalmente. Esto hace que las bases de datos NoSQL sean una excelente opción para casos de uso de baja latencia y alto rendimiento que necesitan escalar horizontalmente más allá de las limitaciones de una sola instancia.

Rendimiento en las bases de datos relacionales normalmente, el rendimiento depende del subsistema de disco. Se necesita la optimización de consultas, índices y estructura de tabla para lograr el máximo rendimiento, en cambio en las bases de datos NoSQL el rendimiento depende del tamaño del clúster de Hardware subyacente, la latencia de red y la aplicación que efectúa la llamada.

Escalado las bases de datos relacionales generalmente escalan en forma ascendente las capacidades de computación del Hardware o la ampliación mediante la adición de réplicas para cargas de trabajo de solo lectura, en cambio en las bases de datos NoSQL normalmente se pueden particionar porque los patrones de acceso son escalables mediante el uso de arquitectura distribuida para aumentar el rendimiento que proporciona un rendimiento constante a una escala casi ilimitada.

API en las bases de datos relacionales se solicita almacenar y recuperar datos que están comunicados mediante consultas que se ajustan a un lenguaje de consulta estructurado (SQL). Estas consultas son analizadas y ejecutadas por la base de datos relacional, en cambio en las bases de datos NoSQL las APIs basadas en objetos permiten a los desarrolladores almacenar y recuperar fácilmente estructuras de datos. Las claves de partición permiten que las aplicaciones busquen pares de clave-valor, conjuntos de columnas o documentos semiestructurados que contengan atributos y objetos de aplicación serializados.

En los últimos años, las bases de datos SQL y NoSQL incluso han comenzado a fusionarse. Por ejemplo, los sistemas de bases de datos, como PostgreSQL, MySQL y Microsoft SQL Server ahora admiten el almacenamiento y la consulta de datos JSON, al igual que las bases de datos NoSQL. Con esto, ahora puede lograr muchos de los mismos resultados con ambas tecnologías. Pero aún no obtiene muchas de las funciones NoSQL, como el escalado horizontal y la interfaz fácil de usar.

2.3.2 SQL en comparación con Terminología NoSQL

La siguiente tabla compara la terminología utilizada por las bases de datos NoSQL seleccionadas con la terminología utilizada por las bases de datos SQL.

SQL	MongoDB	DynamoDB	Cassandra	Couchbase
Tabla	Conjunto	Tabla	Tabla	Bucketdedatos
Fila	Documento	Elemento	Fila	Documento
Columna	Campo	Atributo	Columna	Campo
Claveprincipal	ObjectId	Clave principal	Clave principal	ID del documento
Índice	Índice	Índice secundario	Índice	Índice
Ver	Ver	Índice secundario global	Vista materializada	Ver
Tabla u objeto anidado	Documento incrustado	Mapa	Mapa	Mapa
Matriz	Matriz	Lista	Lista	Lista

Un conjunto de *Campos* formarán un *Documento*, que en caso de asociarse con otros formará una *Colección*. Las *Bases de Datos* estarán formadas por *Colecciones*, y a su vez, cada *Servidor* puede tener tantas bases de datos como el equipo lo permita.

Los tipos de datos soportados son: Numéricos:

- Double: punto flotante de 64 bits.
- Decimal: punto flotante de 128 bits.
- Int: enteros de hasta 32 bits.
- Long: enteros de hasta 64 bits.

Texto:

- String: Cadenas UTF-8.
- Regex: Almacena expresiones regulares.

Fecha:

- Date: entero de 64 bits representa en número de milisegundos desde el primero de enero de 1970.
- Timestamp: entero de 64 bits, en el que los primeros 32 bits representan los segundos pasados desde el primero de enero de 1970 y los otros 32 bits son ordinales incrementales.

Especiales:

- Array: almacena un conjunto de elementos de cualquier tipo.
- Objectld: tipo de dato único, principalmente utilizados para dar valor al campo _id de los documentos.
- Javascript: código de Javascript
- Null: valor nulo
- Boolean: valor booleano.

Consulta y modificación de datos Los datos pueden ser accedidos mediante operaciones CRUD (Create, Read, Update, Delete), la ejecución de cualquiera de estas instrucciones sigue el siguiente esquema:

db.<nombre_colección>.<nombre_Operación>(<condiciones_de_la_operación>)

Operaciones:

- Find: lectura (Read) de datos
- Insert: escritura (Create) de datos
- Update: actualización de valores de datos
- Remove: borrado (Delete) de datos

A modo de ejemplo utilizaremos un JSON como el que se usa en mongoDB. Supongamos que vamos a registrar diferentes personas en una colección perteneciente a una BBDD NoSQL con algunos campos especiales. Estos no necesariamente tienen que seguir un patrón específico como veremos a continuación:

```
item1 = {
    Nombre: "Luís",
    Apellidos: "Martinez",
    Edad: 18,
    Aficiones: ["fútbol","senderismo","tenis"],
    Amigos: [
        {
```

```
            Nombre:"Monica",
            Edad:20
        },
        {
            Nombre:"Andrés",
            Edad:24
        }
    ]
}
db.hi.insert(item1)
db.amigos.find().pretty()
```

Ahora bien, si queremos añadir otros datos con algunas características diferentes la podemos hacer sin mayor problema introduciendo lo siguiente:

```
item2 = {
    Nombre: "Luís",
    Apellidos: "Martinez",
    Edad: 18,
}
db.hi.insert(item2)
db.amigos.find().pretty()
```

En un modelo relacional o SQL clásico esto sería imposible de hacer. Esta es una de las tantas ventajas que hemos comentado

Operación Find ejemplo del comando Find, para encontrar a alguien de nombre Luis:

```
db.amigos.find({Nombre:"Luís"})
```

ejemplo del comando Find, para encontrar a alguien de nombre Luis y apellidos Martinez:

```
db.amigos.find({$and:[{Nombre:"Luís"},{Apellidos:"Martinez"}]})
```

ejemplo del comando Find, para encontrar a alguien de nombre Luis y apellidos Martinez o edad mayor a 18:

```
db.amigos.find({
    $or:[
        {$and:[{Nombre:"Luís"},{Apellidos:"Martinez"}]},
        {Edad:{gte:18}}
    ]
})
```

ejemplo en el que el campo Amigos exista:

```
db.amigos.find({Amigos:{$exists:true}})
```

Lista de comando ejecutables contra la base de datos:

db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]
db.auth(username, password)
db.cloneDatabase(fromhost)
db.commandHelp(name) returns the help for the command
db.copyDatabase(fromdb, todb, fromhost)
db.createCollection(name, { size : ..., capped : ..., max : ... })
db.createUser(userDocument)
db.currentOp() displays currently executing operations in the db
db.dropDatabase()
db.fsyncLock() flush data to disk and lock server for backups
db.fsyncUnlock() unlocks server following a db.fsyncLock()
db.getCollection(cname) same as db['cname'] or db.cname
db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
db.getCollectionNames()
db.getLastError() - just returns the err msg string
db.getLastErrorObj() - return full status object
db.getLogComponents()
db.getMongo() get the server connection object
db.getMongo().setSlaveOk() allow queries on a replication slave server
db.getName()

db.getPrevError()
db.getProfilingStatus() - returns if profiling is on and slow threshold
db.getReplicationInfo()
db.getSiblingDB(name) get the db at the same server as this one
db.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server object if set
db.hostInfo() get details about the server's host
db.isMaster() check replica primary status
db.killOp(opid) kills the current operation in the db
db.listCommands() lists all the db commands
db.loadServerScripts() loads all the scripts in db.system.js
db.logout()
db.printCollectionStats()
db.printReplicationInfo()
db.printShardingStatus()
db.printSlaveReplicationInfo()
db.dropUser(username)
db.repairDatabase()
db.resetError()
db.runCommand(cmdObj) run a database command. if cmdObj is a string, turns it into { cmdObj : 1 }
db.serverStatus()
db.setLogLevel(level,<component>)
db.setProfilingLevel(level,<slowms>) 0=off 1=slow 2=all
db.setWriteConcern(<write concern doc>) - sets the write concern for writes to the db
db.unsetWriteConcern(<write concern doc>) - unsets the write concern for writes to the db
db.setVerboseShell(flag) display extra information in shell output
db.shutdownServer()
db.stats()
db.version() current version of the server

2.4 ¿Qué son las bases de datos de Series de Tiempo?

Una base de datos de series de tiempo[1] (TSDB) es una base de datos que está optimizada para manejar datos con marcas de tiempo o series temporales, que son datos indexados por tiempo. Los TSDB están diseñados para manejar grandes volúmenes de datos de series temporales de manera eficiente y rápida, con funciones como compresión, indexación y agregación. Por lo general, utilizan un formato de almacenamiento en columnas que permite consultas y análisis rápidos de datos de series temporales.

Las bases de datos de series temporales están completamente personalizadas con datos con marca de tiempo, que se indexan y se escriben de manera eficiente de tal manera que puede insertar datos de series temporales. Puede consultar esos datos de series temporales mucho más rápido de lo que lo haría en un entorno relacional o Base de datos NoSQL.

Los ejemplos de bases de datos de series temporales incluyen InfluxDB, Prometheus, TimescaleDB y OpenTSDB. Estas bases de datos se utilizan en una amplia gama de aplicaciones, que incluyen Internet de las cosas, monitoreo de sistemas y análisis de datos financieros.

Influjo DB es una de las bases de datos de series temporales más populares entre DevOps, que está escrita en Go. InfluxDB fue diseñado desde cero para proporcionar un motor de almacenamiento e ingesta de datos altamente escalable. Es muy eficiente para recopilar, almacenar, consultar, visualizar y tomar medidas en flujos de datos. datos de series de tiempo, eventos y métricas en tiempo real.

Proporciona políticas de reducción de resolución y retención de datos para respaldar el mantenimiento de datos de alto valor y alta precisión en la memoria y datos de menor valor en el disco. Está construido de manera nativa de la nube para proporcionar escalabilidad en múltiples topologías de implementación, incluidos los entornos de nube locales e híbridos.

InfluxDB es una solución de código abierto, utiliza InfluxQL, que es muy similar a un lenguaje de consulta de estructura, para interactuar con los datos. Ofrece agentes, paneles, consultas y tareas en un conjunto de he-

[1] Por serie de tiempo nos referimos a datos estadísticos que se recopilan, observan o registran en intervalos de tiempo regulares (diario, semanal, semestral, anual, entre otros). El término serie de tiempo se aplica por ejemplo a datos registrados en forma periódica que muestran, por ejemplo, las ventas anuales totales de almacenes, el valor trimestral total de contratos de construcción otorgados, el valor trimestral del PIB

rramientas. Es una herramienta todo en uno para generar paneles, visualizar y alertar.

Características:

- Alto rendimiento para datos de series de tiempo con alto nivel de ingesta y consultas en tiempo real
- InfluxQL para interactuar con datos que es un SQL como un lenguaje de consulta
- Componente principal de la pila TICK (Telegraf, InfluxDB, Chronograf y Kapacitor).
- Soporte de complementos para protocolos como collectd, Graphite, OpenTSDB para la ingestión de datos
- Puede manejar millones de puntos de datos en solo 1 segundo
- Políticas de retención para automáticos.ally eliminando los datos obsoletos

Como es de código abierto, puedes descargarlo e iniciarlo en tu servidor. Sin embargo, ofrecen InfluxDB Cloud en AWS, Azurey GCP.

Prometeo es un código abierto Monitor-Solución innovadora utilizada para comprender información a partir de datos de métricas y enviar las alertas necesarias. Tiene una base de datos local de series temporales en disco que almacena datos en un formato personalizado en el disco.

El modelo de datos de Prometheus es multidimensional y se basa en series de tiempo; almacena todos los datos como secuencias de valores con marca de tiempo. Es muy útil cuando se trabaja con series de tiempo completamente numéricas. Recopilar datos de microservicios y consultarlos es una de las fortalezas de Prometheus. Se integra estrechamente con Grafana para visualización.

Características

- Tiene un modelo multidimensional que utiliza el nombre de las métricas y los pares clave-valor (etiquetas)
- PromQL para consultar datos de series temporales para generarate tablas, alertas y gráficos Adhoc

- Utiliza el modo de extracción HTTP para recopilar datos de series de tiempo
- Utiliza intermediario gateforma de impulsar series de tiempo

Prometeo tiene cientos de exportadores para exportar los datos de Windows, Linux, Java, bases de datos, API, sitios web, hardware de servidor, PHP, mensajería y más. A monitor Linux, mira esto Configuración de Prometheus + Grafana.

Escala de tiempoDB es una base de datos relacional de código abierto que hace que SQL sea escalable para datos de series temporales. Esta base de datos se basa en PostgreSQL. Ofrece dos productos: en la primera opción es una edición comunitaria, de uso gratuito, que puede instalar en su servidor. La opción es TimescaleDB Cloud, donde obtiene una infraestructura completamente alojada y administrada en la nube para sus necesidades de implementación.

Se puede utilizar para DevOps. monitoring, entender métricas de aplicaciones, seguimiento de datos de dispositivos IoT, comprensión landing datos financieros, etc. Puede medir registros, eventos de Kubernetes, métricas de Prometheus e incluso métricas personalizadas.

Características

- Ejecute consultas de 10 a 100 veces más rápido que PostgreSQL, MongoDB
- Puede escalar a petabytes y escribe millones de puntos de datos por segundo
- Muy similar a PostgreSQL, muy fácil de operar para desarrolladores y administradores.ate
- Combina funcionalidades de bases de datos relacionales y de series de tiempo para crear aplicaciones potentes.
- Algoritmos incorporados y función de rendimiento para ahorrar muchos costos.

Grafito es una solución todo en uno para almacenar y visualizar de manera eficiente datos de series de tiempo en tiempo real. Graphite puede hacer dos cosas, almacenar datos de series de tiempo y representar gráficos bajo demanda. Pero no recopila datos por usted; para eso, puede utilizar herramientas como collectd, Ganglia, Sensu, telegraf, etc.

Tiene tres componentes: Carbono, Susurro, y Web de grafito. Carbon recibe los datos de la serie temporal, agregados y lo persiste en el disco. Whisper es un almacenamiento de base de datos de series temporales que almacena los datos. Graphite-Web es la interfaz para crear paneles y visualizar los datos.

Características del grafito:

- El formato de métricas en el que se envían los datos es sencillo.
- API integral para renderizar los datos y crear cuadros, tableros de control, gráficos
- Proporciona un amplio conjunto de funciones de representación transformadora y biblioteca estadística
- Encadena múltiples funciones de renderizado para construir una consulta de destino.

QuestDB es una base de datos relacional orientada a columnas que puede realizar análisis en tiempo real de datos de series de tiempo. Funciona con SQL y algunas extensiones para crear un modelo relacional para datos de series de tiempo. QuestDB ha sido codificado desde cero y no tiene dependencias que mejoren su rendimiento.

QuestDB admite uniones relacionales y de series de tiempo, lo que ayuda a correlacionar los datos. La forma más sencilla de comenzar con QuestDB es implementarlo dentro de un contenedor Docker.

Características

- Consola interactiva para importar datos usando arrastrar y soltar y consultarlos
- Compatible con la nube nativa (AWS, Azure, GCP), local o integrada
- Proporciona integración empresarial con funciones como directorio activo, alta disponibilidad, enterprise seguridad, agrupación

- Proporciona información en tiempo real mediante análisis operativos y predictivos.

Flujo de tiempo de AWS es un servicio de base de datos de series temporales sin servidor que es rápido y escalable. Se utiliza principalmente para que las aplicaciones de IoT almacenen billones de eventos en un día y 1000 veces más rápido con una décima parte del costo de las bases de datos relacionales.

Utilizando su motor de consultas especialmente diseñado, puede consultar simultáneamente datos recientes y datos históricos almacenados.neously. Proporciona múltiples funciones integradas para analizar datos de series temporales y encontrar información útil.

Características

- No hay servidores que administrar ni instancias que aprovisionar; todo se maneja automaticoally.
- Rentable, pague solo por lo que ingiera, almacene y consulte.
- Capaz de ingerir billones de eventos al día sin que disminuya el rendimiento.
- Capacidad analítica incorporada con funciones estándar de SQL, interpolación y suavizado para identificar tendencias, patrones y anomalías
- Todos los datos se cifran mediante el sistema de administración de claves (KMS) de AWS con claves administradas por el cliente (CMK)

OpenTSDB es una base de datos de series temporales escalable que se ha escrito sobre HBase. Es capaz de almacenar billones de puntos de datos a millones de escrituras por segundo. Puede mantener los datos en OpenTSDB para ver con su marca de tiempo original y valor preciso, para que no pierda ningún dato.

Tiene un demonio de serie temporal (TSD) y utilidades de línea de comandos. El demonio de series de tiempo es responsable de almacenar datos en HBase o recuperarlos de él. Puede hablar con TSD mediante HTTP API, telnet o una sencilla GUI incorporada. Necesita herramientas como flume, collectd, vacuumetrix, etc., para recopilar datos de varias fuentes en OpenTSDB.

Características

- Puede agregar, filtrar y reducir la resolución de métricas a una velocidad vertiginosa
- Almacena y escribe datos con precisión de milisegundos
- Se ejecuta en Hadoop y HBase y se escala fácilmente agregando nodos al clúster.
- Utiliza GUI para generar gráficos

En resumen: dado que en la actualidad se utilizan cada vez más dispositivos IoT/inteligentes, se está generando un enorme tráfico en tiempo en sitios Web con millones de eventos en un día, las operaciones en el mercado están aumentando y ¡ha llegado la base de datos de series temporales!.

2.5 Bases de Datos de Código Abierto y sus Ventajas Frente a las Propietarias

Elegir la base de datos más adecuada para un proyecto supone enfrentarse a todo tipo de posibilidades de elección. Quizá la primera, y la más importante, es elegir una base de datos de código abierto o una propietaria. Es decir, una de código abierto, generalmente gratuita, o una propiedad de una empresa. A la vista de esto, la primera ventaja de las bases de datos de código abierto salta a la vista: evitan tener que hacer frente a un desembolso económico fuerte, puesto que no hace falta comprar su licencia a un fabricante. A no ser que tengas necesidades muy concretas y específicas, pero la proliferación de bases de datos de código abierto en los últimos años, así como de diversos complementos para ellas, hace que resulten cada vez más atractivas incluso para fines muy concretos.

Mientras tanto, la popularidad de los sistemas de bases de datos con licencia comercial, es decir, de las propietarias, ha ido cayendo a lo largo de los últimos años. Y según la lista de sistemas de gestión de bases de datos de DB-Engines, en muchos casos han sido superadas por las bases de datos de código abierto en lo que respecta a la adopción en empresas.

Ventajas de las Bases de Datos de Código Abierto Una base de datos de código abierto es un sistema de gestión de bases de datos abierto y de uso gratuito, y quizá una de las pocas desventajas que tiene frente a

las bases de datos propietarias es que no cuenta con un servicio de soporte técnico por parte de un proveedor, algo con lo que sí cuentan las bases de datos propietarias. Pero precisamente de esta desventaja nace una de sus ventajas.

Las bases de datos de código abierto cuentan con una nutrida comunidad de usuarios que, en muchos casos no solo comparten sus conocimientos en foros y grupos de usuarios especializados, sino que en muchos casos están dispuestos a ayudar a otros usuarios con problemas y necesidades concretas con ellas.

Además, la información que se guarda en una base de datos de código abierto es de quien la inserta en ella. Sin ningún tipo de restricción. También puedes modificarla para adaptarla a tus necesidades, aunque para ello necesitas contar con ciertos conocimientos de desarrollo. Puedes crear aplicaciones y complementos con base en el Software de código abierto que elijas, de una manera muy parecida a como sucede por ejemplo con el CMS WordPress. Por lo tanto, las bases de datos de código abierto te ofrecen un sinfín de posibilidades de personalización y ampliación.

Las Principales Bases de Datos de Código Abierto El número de bases de datos de código abierto crece cada vez más a medida que pasan los años. No obstante, entre todas las posibilidades, hay varias que, según TechTarget, o bien por su sencillez y robustez, o bien por sus posibilidades y facilidades de ampliación y personalización, son las que consiguen una mayor adopción y despliegue en las empresas. Entre ellas están las seis siguientes, quizá las más populares y utilizadas.

MySQL Sin duda esta es la más extendida de las bases de datos de código abierto, y durante muchos años, la opción por defecto para muchos desarrolladores a la hora de optar por una base de datos. Oracle se hizo con ella hace ya bastantes años cuando compró Sun Microsystems, y es un sistema de gestión de bases de datos relacional. Esto quiere decir que utiliza tablas para almacenar los datos, y los tipos de datos almacenados deben estar relacionados entre sí de alguna manera. Cuenta con una edición de código abierto y algunas versiones de pago, y debido a su popularidad hay muchas herramientas de terceros para ella, así como una cantidad importante de documentación centrada en facilitar su aprendizaje.

La base de MySQL se originó a partir de un sistema de base de datos

propietario antiguo, mantener la mayor parte de su compatibilidad y convertir los resultados en código abierto. Sus creadores se centraron en la velocidad, y en la actualidad sigue estando considerada entre las opciones más rápidas de bases de datos. Además, de la velocidad, cuenta con un Script que contribuye a la mejora de la seguridad de las bases de datos de los usuarios. Esta base de datos es compatible, entre otros, con los lenguajes de programación C, C++, Java, Python y Ruby.

No obstante, se trata de una base de datos con licencia dual. Esto implica que algunas de sus funciones y Plugins solo son accesibles en ediciones propietarias de pago. En cuanto a su curva de aprendizaje, no es muy elevada. No es necesario saber mucho SQL para utilizarla, y puedes trabajar con la base de datos desde la línea de comandos. MySQL tiene una compatibilidad elevada prácticamente con cualquier sistema operativo. Se trata de una base de datos sólida, rápida y versátil, por lo que es adecuada para multitud de casos de uso.

MariaDB Cuando Oracle compró Sun, y con ella MySQL, hubo quien no se lo tomó muy bien, porque aunque Oracle ha mantenido MySQL como código abierto, la compañía no tiene mucha reputación que digamos como defensora del Software abierto. Uno de los que no se creyó a Oracle fue uno de los propios fundadores de MySQL, que dejó la compañía para crear MariaDB, un fork de MySQL con una compatibilidad casi completa con ella.

Se puede utilizar como un sustituto prácticamente para todo de MySQL, aunque cuenta con varias funciones únicas que la distinguen de ella. Para empezar, MariaDB usa el motor de almacenamiento Aria para gestionar querys de SQL complejas. Esto hace que la base de datos sea todavía más veloz que MySQL. Además, puedes utilizar filas dinámicas como columnas de tabla, lo que la hace más flexible y adaptable a diversas situaciones.

Además, cuenta con motores de almacenamiento especializados para casos de uso concretos que no tiene MySQL. Por ejemplo, permite implementar el almacenamiento distribuido o las transacciones distribuidas. Como hemos mencionado, puedes utilizar MariaDB en los mismos supuestos y casos que MySQL, pero no hay compatibilidad entre las dos, por lo que tendrás que elegir cuál de las dos quieres utilizar.

PostgreSQL Esta base de datos relacional y de código abierto es muy utilizada en ciencia de datos y creación de gráficos, y también en el sector

de la Inteligencia Artificial. Esto último se debe a que PostgreSQL está especialmente pensada para aplicaciones de Python y Ruby. No obstante, también es compatible con PHP. Se ha desarrollado pensando en la elegancia, y cuenta con ciertas funciones bastante diferenciadas. Entre ellas están la posibilidad de implementar la replicación asíncrona, y el soporte nativo para el almacenamiento de documentos de estilo JSON o XML.

Además, PostgreSQL permite hacer búsquedas completas de texto en la base de datos, y cuenta con varios tipos de datos integrados de gran valor para algunas aplicaciones, como los rangos, los arrays y la geolocalización. Eso sí, no es excesivamente adecuada para aplicaciones que requieran operaciones intensivas de lectura, y la creación de informes de datos con regularidad puede hacer que el almacenamiento de documentos no esté muy fino si tiene que almacenar un conjunto de datos muy grande.

SQLite También relacional, SQLite es una base de datos de código abierto compuesta por una librería ligera y pequeña que ofrece un motor de base de datos. A pesar de su ligereza, permite generar bases de datos de cientos de Terabytes, y un tamaño máximo de filas de un Gigabyte. Incluso con archivos de este tamaño, SQLite sigue siendo rápida. Además, es compatible con Java, Python, C o C++, entre otros lenguajes.

Cuenta con decenas de casos de uso, y por ejemplo, la encontrarán muy útil los desarrolladores de aplicaciones sencillas. Está integrada en smartphones y en muchos ordenadores. También es muy valorada en aplicaciones de Internet de las Cosas y Webs de poco tráfico. Eso sí, por su estructura, no está indicada para sitios con mucho tráfico, ya que su rendimiento se resiente. Además, cuenta con ciertas limitaciones que en algunos casos pueden ser importantes.

Estas hacen que, entre otras cosas, no esté indicada en aplicaciones a las que tienen que acceder varios usuarios con permisos de acceso especiales. SQLite lee y escribe en un archivo de disco común y corriente, por lo que los únicos permisos de acceso aplicables para esta base de datos son los incluidos habitualmente en el sistema operativo.

MongoDB Es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios

están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris y es usado en múltiples proyectos o implementaciones en empresas como MTV Network, Craigslist, BCI o Foursquare.

La razón de esto es que MongoDB, al estar escrito en C++, cuenta con una más que notoria capacidad para aprovechar los recursos de la máquina y, al estar licenciado bajo una licencia GNU AGPL 3.0, es posible adaptarlo a nuestras necesidades. Si tuviéramos que resumir a una la principal característica a destacar de MongoDB, sin duda esta sería la velocidad, que alcanza un balance perfecto entre rendimiento y funcionalidad gracias a su sistema de consulta de contenidos. Pero sus características principales no se limitan solo a esto, MongoDB cuenta, además, con otras que lo posicionan como el preferido de muchos desarrolladores.

Redis Esta base de datos de código abierto es distinta a las más utilizadas, ya que se utiliza sobre todo en situaciones completamente distintas a ellas. Se trata de una base de datos que permite estructurar datos como pares de valor clave. Por tanto, se trata de un sistema parecido a los arrays asociativos de PHP o los diccionarios de Python. En realidad, Redis es un sistema para enlazar datos para su referencia, más adelante, con rapidez. Está considerada como la mejor elección como base de datos para cacheado, y también para el trabajo con datos distribuidos.

Para empezar, es una solución de almacenamiento en memoria, enteramente en RAM. Esto quiere decir que sus velocidades de lectura y escritura son muy rápidas. Para aprender sus bases tampoco necesitas mucho tiempo, ya que puedes conocerlas en minutos y empezar con rapidez con el almacenamiento de objetos en ella. Eso sí, puede que no sea muy útil para aplicaciones complejas, pero basta con utilizarlas junto con otras bases de datos, como MariaDB, para el resto de la aplicación que se esté desarrollando.

2.6 Programando en Diferentes Manejadores de Bases de Datos

Existen diferentes paquetes para que permiten trabajar con bases de datos, en esta sección mostraremos cómo trabajar con algunos de ellos:

- SQLite
- MySQL/MariaDB/Microsoft SQL Server

- PostgreSQL
- MongoDB
- Redis

A continuación, mostraremos el mismo ejemplo en cada uno de estos manejadores:

2.6.1 SQLite

es una herramienta de Software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser una PDA o un teléfono celular. Está presente en todos los dispositivos Android, iPhone, Mac y Windows 10, además en los navegadores Firefox, Chromium, Chrome y Safari entre otros.

SQLite implementa el estándar SQL92 y también agrega extensiones que facilitan su uso en cualquier ambiente de desarrollo. Esto permite que SQLite soporte desde las consultas más básicas hasta las más complejas del lenguaje SQL, y lo más importante es que se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, ya que existe compatibilidad al 100% entre las diversas plataformas disponibles, haciendo que la portabilidad entre dispositivos y plataformas sea transparente.

Lenguajes de Programación de SQLite gracias a que SQLite[2] es Software libre, es posible encontrar una gran cantidad de componentes, librerías y Drivers para interactuar con SQLite desde una gran diversidad de lenguajes y plataformas de programación. Ya sea que estemos utilizando lenguajes modernos como Java, Perl, Python, PHP, Ruby, C#, lenguajes más antiguos como Pascal, SmallTalk, Clipper, o lenguajes poco conocidos como Suneido, REXX, S-Lang, para todos podemos encontrar librerías y ejemplos de código para SQLite.

[2] También existe Rqlite que es una base de datos relacional distribuida y liviana de código abierto para configuraciones de cluster que está construido sobr SQLite y tiene como objetivo ser fácil de usar y tolerante a fallas.

http://www.sqlite.org/cvstrac/wiki?p=SqliteWrappers ofrece más información sobre "wrappers" para SQLite sobre diferentes plataformas y lenguajes.

Instalación en Debian GNU/Linux Para instalar el paquete usamos:

```
# apt install sqlite3 sqlite3-doc sqlitebrowser
```

y ya podemos iniciar a trabajar, usando

```
# sqlite3
```

SQLite y Python SQLite es probablemente la base de datos más sencilla para conectarse con una aplicación Python, ya que no necesita instalar ningún módulo SQL externo de Python para hacerlo. De forma predeterminada, su instalación de Python contiene una biblioteca SQL de Python llamada *sqlite3* que puede usar para interactuar con una base de datos SQLite.

Además, las bases de datos SQLite no tienen servidor y son autónomas, ya que leen y escriben datos en un archivo. Esto significa que, a diferencia de MySQL y PostgreSQL, ni siquiera necesita instalar y ejecutar un servidor SQLite para realizar operaciones de base de datos.

Así es como se usa *sqlite3* para conectarse a una base de datos SQLite en Python:

```
import sqlite3
from sqlite3 import Error

def create_connection(path):
    connection = None
    try:
        connection = sqlite3.connect(path)
        print("Connection to SQLite DB successful")
    except Error as e:
        print(f"The error '{e}' occurred")
    return connection

connection = create_connection("/trayectoria/basedatos")
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema que cree o abra una base de datos:

```
sqlite3 prueba.db
```

y estamos en posibilidad de crear nuestras tablas, por ejemplo:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
```

e ingresamos valores:

```
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
```

y los visualizamos:

```
select * from Productos;
```

ahora procedemos hacer lo pertinente para nuestra otra tabla, usando:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
  Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

2.6.2 MySQL/MariaDB

MySQL/MariaDB es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, estos manejadores de bases de datos, no son más que una aplicación que permite gestionar archivos llamados de bases de datos.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL/MariaDB, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL/MariaDB fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL/MariaDB, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

MariaDB este sistema de gestión de base de datos es una bifurcación de MySQL, aunque parecidos, estos dos sistemas de gestión de bases de datos siguen siendo bastante diferentes:

- MariaDB tiene licencia GPL mientras que MySQL tiene un enfoque de doble licencia
- Cada mango se acumula de una manera diferente
- MariaDB soporta muchos motores de almacenamiento diferente
- En muchos escenarios, MariaDB ofrece un mejor rendimiento

Instalación en Debian GNU/Linux En este caso tenemos dos opciones, instalar MariaDB que ya es parte integral de múltiples aplicaciones dentro de GNU/Linux o usar MySQL, en cada caso el uso es prácticamente el mismo.

Para instalar el paquete MariaDB usamos:

```
# apt install mariadb-server mariadb-client
```

y para configurarlo, usamos:

```
# mysql
> use mysql
> update user set plugin=” where user='root';
> alter user root@localhost identified by 'XXXXXXX';
> flush privilegies;
> exit
```

y para usarlo

```
$ mysql -u root -p
```

Para instalar el paquete MySQL usamos:

```
# apt install mysql-common mysql-client mysql-server mytop
mysql-admin
```

y para configurarlo (las bases de datos se guardan en /var/lib/mysql), usamos:

```
# mysql_secure_installation
```

o

```
# mysql
```

en caso de no crear contraseña, podemos ingresar así:

```
$ mysql -u root -p
```

una vez en la consola del cliente de MySQL creamos un nuevo usuario

```
> create user nombre identified by 'XXXXX';
```

y a continuación le damos privilegios totales

```
> gran all privileges on *.* to nombre with grant option;
> flush privileges;
> exit;
```

otras opciones a instalar son:

```
# apt install ferret mysql-workbench mysql-workbench-data
mycli
# snap install squirrelsql mysql-workbench-community data-
grip dbeaver-ce
```

Crear un Nuevo Usuario Para esto usamos la declaración CREATE USER. La sintaxis básica de la declaración es:

> CREATE USER 'username'@'hostname' IDENTIFIED BY 'password';

En la sintaxis anterior, claramente debes de reemplazar el nombre de usuario y la contraseña con el nombre de usuario y contraseña deseados.

Establezca el hostname en localhost si desea que el usuario pueda conectarse a MySQL Server solo desde localhost. Si desea que el usuario pueda conectarse desde cualquier host, use el comodín % como hostname.

Por ejemplo, crearemos un usuario con el nombre Javier y la contraseña 1234 usando el siguiente comando:

> CREATE USER 'Javier'@'localhost' IDENTIFIED BY '1234';

Sin embargo, este usuario no podrá trabajar con ninguna de las bases de datos MySQL hasta que se le otorguen privilegios adicionales. Ahí el detalle.

Como otorgar privilegios adicionales al usuario inmediatamente después de crear con éxito el nuevo usuario, podemos otorgar privilegios a este nuevo usuario. En la mayoría de los casos, otorgará privilegios a los usuarios de MySQL en función de la base de datos particular a la que la cuenta debería tener acceso. Hay varios tipos de privilegios que se pueden otorgar a una cuenta de usuario, como:

- ALL PRIVILEGES: otorga todos los privilegios a una cuenta de usuario.
- ALTER: el usuario puede cambiar la estructura de una tabla o base de datos.
- CREATE: la cuenta de usuario puede crear bases de datos y tablas.
- DROP: la cuenta de usuario puede eliminar bases de datos y tablas.
- DELETE: la cuenta de usuario puede eliminar filas de una tabla específica.
- INSERT: la cuenta de usuario puede insertar filas en una tabla específica.

- SELECT: la cuenta de usuario puede leer una base de datos.
- UPDATE: la cuenta de usuario puede actualizar las filas de la tabla.

Para proporcionar acceso y otorgar permisos a un $user, generalmente se debe usar la siguiente declaración GRANT:

> GRANT permission_type ON privilege_level TO 'username'@'hostname';

Sí retomamos el ejemplo de arriba, al usuario Javier y su base de datos Javierdb, se le puede otorgar permisos, así:

> GRANT ALL PRIVILEGES ON Javierdb.* TO 'Javier'@'localhost';

Oh claro otorgarle permisos específicos, solo cambiando el ALL PRIVILEGES, por SELECT, CREATE, INSERT.... todos separados por una coma (,).

En algunos casos, es posible que desee crear otro "superusuario". Para otorgar a un usuario los mismos privilegios que el usuario raíz de MySQL, usamos el siguiente comando, que otorga privilegios globales al usuario james que se conecta a través de localhost:

> GRANT ALL ON *.* TO 'Javier'@'localhost' WITH GRANT OPTION;

Como revocar privilegios y eliminar un usuario si necesitas revocar privilegios del usuario Javier en una base de datos Javierdb, sería suficiente con la sintaxis siguiente, que es muy similar

> REVOKE ALL PRIVILEGES ON Javierdb.* TO 'Javier'@'localhost';

Mientras que para eliminar a ese usuario. Basta con:

> DROP USER 'Javier'@'localhost';

Para que cualquier cambio surta efecto, es necesario usar el comando:

> FLUSH PRIVILEGES;

Mantenimiento En caso de requerir mantenimiento a las tablas (analizar, verificar, optimizar y reparar las tablas de una base de datos) podemos usar *mysqlcheck*, por ejemplo para verificar y reparar todas las bases de datos:

```
$ mysqlcheck -A –auto-repair -u root -p
```

para forzar la optimización y autoreparación de todas las tablas:

```
$ mysqlcheck -A –auto-repair –f -o -u root -p
```

para verificar todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -c
```

para verificar las tablas de todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -a
```

para reparar las tablas de todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -r
```

para optimizar las tablas de todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -o
```

Exportación e Importación Para exportar una base de datos, debemos utilizar una utilidad de cliente *mysqldump* que crea la copia de seguridad lógica de las bases de datos en archivos de texto SQL, lo que facilita la transferencia de archivos de un servidor a otro, para ello hacemos:

```
$ mysqldump -u usuario -p baseDatos > archivo.sql
```

para restaurarla usamos:

```
$ mysqldump -u usuario -p baseDatos < archivo.sql
```

Exportar como CSV Podemos exportar el contenido de una o más tablas al formato CSV, por ejemplo:

```
> SELECT * FROM test_proyect
INTO OUTFILE '/var/tmp/get_all_queries.csv'
FIELDS ENCLOSED BY '"'
TERMINATED BY ';'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n';
```

MySQL y Python A diferencia de SQLite, no existe un módulo SQL de Python predeterminado que pueda usar para conectarse a una base de datos MySQL. En su lugar, deberá instalar un controlador Python SQL para MySQL a fin de interactuar con una base de datos MySQL desde una aplicación Python. Uno de estos controladores es *mysql-connector-python*. Puede descargar este módulo SQL de Python con pip:

```
$ pip install mysql-connector-python
```

Tenga en cuenta que MySQL es un sistema de gestión de bases de datos basado en servidor. Un servidor MySQL puede tener varias bases de datos. A diferencia de SQLite, donde crear una conexión equivale a crear una base de datos, una base de datos MySQL tiene un proceso de dos pasos para la creación de la base de datos:

- Establezca una conexión a un servidor MySQL.
- Ejecute una consulta separada para crear la base de datos.

Defina una función que se conecte al servidor de la base de datos MySQL y devuelva el objeto de conexión:

```
import mysql.connector
from mysql.connector import Error

def create_connection(host_name, user_name, user_password):
    connection = None
    try:
        connection = mysql.connector.connect(
```

```
            host=host_name,
            user=user_name,
            passwd=user_password
        )
        print("Connection to MySQL DB successful")
    except Error as e:
        print(f"The error '{e}' occurred")
    return connection

connection = create_connection("localhost", "root", "")
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema que nos muestre las bases de datos existentes:

```
show databases
```

si necesitamos crear una, por ejemplo prueba, usamos:

```
create databases prueba
```

ahora, le indicamos que usaremos dicha base de datos, mediante:

```
use prueba
```

y ahora podremos, por ejemplo ver las tablas creadas usando:

```
show tables
```

y estamos en posibilidad de crear nuestras tablas, por ejemplo:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
```

e ingresamos valores:

```
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
```

y los visualizamos:

```
select * from Productos;
```

ahora procedemos hacer lo pertinente para nuestra otra tabla, usando:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    Select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
  Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    From Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

2.6.3 PostgreSQL

Las bases de datos relacionales son uno de los tipos más populares que existen, y su uso nos permite crear sitios web de toda clase. Entre los sistemas de bases de datos relacionales más utilizados se encuentra uno que es conocido como PostgreSQL, que dicho sea de paso está entre los más usados a nivel mundial.

PostgreSQL, o simplemente Postgres para darle un nombre más pintoresco, es un sistema de código abierto de administración de bases de datos del tipo relacional, aunque también es posible ejecutar consultas que sean no relaciones. En este sistema, las consultas relacionales se basan en SQL, mientras que las no relacionales hacen uso de JSON.

Como decíamos, se trata de un sistema de código abierto y además gratuito, y su desarrollo es llevado adelante por una gran comunidad de colaboradores de todo el mundo que día a día ponen su granito de arena para hacer de este sistema una de las opciones más sólidas a nivel de bases de datos.

Dos detalles a destacar de PostgreSQL es que posee data types (tipos de datos) avanzados y permite ejecutar optimizaciones de rendimiento avanzadas, que son características que por lo general solo se ven en sistemas de bases de datos comerciales, como por ejemplo SQL Server de Microsoft u Oracle de la compañía homónima.

Instalación en Debian GNU/Linux Para instalar el paquete usamos:

```
# apt install postgresql postgresql-client postgresql-doc postgresql-contrib
# snap install postgresql10
```

y ya podemos conectarnos al servidor mediante:

```
# postgres psql
```

PostgreSQL y Python Al igual que MySQL, no existe una biblioteca Python SQL predeterminada que pueda usar para interactuar con una base de datos PostgreSQL. En su lugar, debe instalar un controlador Python SQL de terceros para interactuar con PostgreSQL. Uno de esos controladores Python SQL para PostgreSQL es *psycopg2*. Ejecute el siguiente comando en su terminal para instalar el módulo Python SQL psycopg2:

```
$ pip install psycopg2
```

Al igual que con las bases de datos SQLite y MySQL, definirás create_connection () para hacer una conexión con tu base de datos PostgreSQL:

```
import psycopg2
from psycopg2 import OperationalError

def create_connection(db_name, db_user, db_password, db_host, db_port):
    connection = None
    try:
```

```
            connection = psycopg2.connect(
                database=db_name,
                user=db_user,
                password=db_password,
                host=db_host,
                port=db_port,
            )
            print("Connection to PostgreSQL DB successful")
        except OperationalError as e:
            print(f"The error '{e}' occurred")
        return connection

connection = create_connection("postgres", "postgres", "abc123",
"127.0.0.1", "5432")
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema que nos muestre las bases de datos existentes:

```
\l
```

si necesitamos crear una, por ejemplo prueba, usamos:

```
create databases prueba
```

ahora, le indicamos que usaremos dicha base de datos, mediante:

```
\c prueba
```

y ahors podremos, por ejemplo ver las tablas creadas usando:

```
\d
```

y estamos en posibilidad de crear nuestras tablas, por ejemplo:

```
create table Productos(
    id_prod int,
    descripcion text not null,
    precio money,
    precioVenta money,
    primary key (id_prod)
);
```

e ingresamos valores:

```
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
```

y los visualizamos:

```
select * from Productos;
```

ahora procedemos hacer lo pertinente para nuestra otra tabla, usando:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '12/01/2020 8:01:00.00'), (2, 5, '12/01/2020 10:15:00.00'),
(2, 4, '12/01/2020 13:34:00.00'), (1, 3, '12/01/2020 21:56:00.00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

2.6.4 MongoDB

MongoDB (del inglés humongous, "enorme") es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris y es usado en múltiples proyectos o implementaciones en empresas como MTV Network, Craigslist, BCI o Foursquare.

La razón de esto es que MongoDB, al estar escrito en C++, cuenta con una más que notoria capacidad para aprovechar los recursos de la máquina

y, al estar licenciado bajo una licencia GNU AGPL 3.0, es posible adaptarlo a nuestras necesidades.

Si no conocemos a MongoDB, al principio podemos sentirnos un poco perdidos. Al no tener tablas ni nada que se parezca a SQL como referencia, tendremos que estudiar un poco su filosofía y características para entender cómo manejar los datos. Aun así, MongoDB es una seria candidata para almacenar los datos de nuestras aplicaciones.

Características de MongoDB si tuviéramos que resumir a una la principal característica a destacar de MongoDB, sin duda esta sería la velocidad, que alcanza un balance perfecto entre rendimiento y funcionalidad gracias a su sistema de consulta de contenidos. Pero sus características principales no se limitan solo a esto, MongoDB cuenta, además, con otras que lo posicionan como el preferido de muchos desarrolladores.

Características principales:

- Consultas ad hoc. Con MongoDb podemos realizar todo tipo de consultas. Podemos hacer búsqueda por campos, consultas de rangos y expresiones regulares. Además, estas consultas pueden devolver un campo específico del documento, pero también puede ser una función JavaScript definida por el usuario.

- Indexación. El concepto de índices en MongoDB es similar al empleado en bases de datos relacionales, con la diferencia de que cualquier campo documentado puede ser indexado y añadir múltiples índices secundarios.

- Replicación. Del mismo modo, la replicación es un proceso básico en la gestión de bases de datos. MongoDB soporta el tipo de replicación primario-secundario. De este modo, mientras podemos realizar consultas con el primario, el secundario actúa como réplica de datos en solo lectura a modo copia de seguridad con la particularidad de que los nodos secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

- Balanceo de carga. Resulta muy interesante cómo MongoDB puede escalar la carga de trabajo. MongoDB tiene la capacidad de ejecutarse de manera simultánea en múltiples servidores, ofreciendo un balanceo

de carga o servicio de replicación de datos, de modo que podemos mantener el sistema funcionando en caso de un fallo del hardware.

- Almacenamiento de archivos. Aprovechando la capacidad de MongoDB para el balanceo de carga y la replicación de datos, Mongo puede ser utilizado también como un sistema de archivos. Esta funcionalidad, llamada GridFS e incluida en la distribución oficial, permite manipular archivos y contenido.
- Ejecución de JavaScript del lado del servidor. MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

Instalación en Debian GNU/Linux Para instalar el paquete usamos:

```
# apt install mongodb mongodb-clients mongodb-server
# snap install mongo44-configurable
```

y ya podemos iniciar su uso mediante:

```
$ mongo
```

MongoDB y Python A diferencia de SQLite, no existe un módulo noSQL de Python predeterminado que pueda usar para conectarse a una base de datos MongoDB. En su lugar, deberá instalar un controlador Python noSQL para MongoDB a fin de interactuar con una base de datos MongoDB desde una aplicación Python. Uno de esos controladores es *pymongo*. Puede descargar este módulo noSQL de Python con pip:

```
$ pip install pymongo
```

Defina una función que se conecte al servidor de la base de datos MongoDB y devuelva el objeto de conexión:

```
from pymongo import MongoClient
import pymongo

def get_database():
    # Provide the mongodb atlas url to connect python to
mongodb using pymongo
```

```
        CONNECTION_STRING = "mongodb+srv:
    //<username>:<password>@<cluster-name>.mongodb.net/myFirstDatabase"
        # Create a connection using MongoClient. You can im-
  port MongoClient or use pymongo.MongoClient
        client = MongoClient(CONNECTION_STRING)
        # Create the database for our example
        return client['user_shopping_list']

    dbname = get_database()
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema nos muestre las colecciones existentes (bases de datos), mediante:

```
db
```

podemos crear una usando:

```
db.createCollection(prueba)
```

podemos ver las existentes mediante:

```
show collections
```

y le pedimos usar alguna, mediante:

```
use prueba
```

y estamos en posibilidad de crear nuestras documentos, por ejemplo:

```
var miProductos =
[
    {"_id" : 1, "descripcion" : "Cigarros", "precio" : 25.5,
  "precioVenta" : 30.0},
    {"_id" : 2, "descripcion" : "Refresco", "precio" : 4.5,
  "precioVenta" : 6.0},
];
db.Productos.insert(miProductos);
```

y los visualizamos:

```
db.Productos.find()
```

ahora procedemos hacer lo pertinente para nuestra otro documento, usando:

```
var miVentas =
[
    {"_id" : 1, "id_prod" : 1, "Cantidad" : 2, "Fecha" :
"2020/12/01 8:01:00"},
    {"_id" : 2, "id_prod" : 2, "Cantidad" : 5, "Fecha" :
"2020/12/01 10:15:00"},
    {"_id" : 3, "id_prod" : 2, "Cantidad" : 4, "Fecha" :
"2020/12/01 13:34:00"},
    {"_id" : 4, "id_prod" : 1, "Cantidad" : 3, "Fecha" :
"2020/12/01 21:56:00"},
];
db.Ventas.insert(miVentas);
db.Ventas.find()
```

de esta forma, ahora podemos usar Find para hacer el equivalente a un JOIN como en los ejemplos anteriores y mostrar el resultado de la solicitud, mediante:

```
db.Ventas.find().forEach(
    function (object) {
        var commonInBoth=db.Productos.findOne({ "_id":
object.id_prod} );
        if (commonInBoth != null) {
            print(object.Fecha, object.Cantidad, common-
InBoth.descripcion, commonInBoth.precioVenta, (commonInBoth.precioVenta
* object.Cantidad));
        }
    }
);
```

Exportar Base de Datos en MongoDB En MongoDB, cuando se exporta información, se puede obtener un archivo de texto legible por el ser humano con sus datos. Por defecto, la información se exporta en formato *json*, pero también se puede exportar en formato *csv*.

Para exportar información desde MongoDB, utilizaremos el comando mongoexport. Este comando nos permite exportar de forma detallada, por lo que podemos especificar una base de datos, una colección, un campo e incluso utilizar una consulta para la exportación.

Si queremos exportar debemos de ejecutar lo siguiente:

```
# mongoexport –db mibasededatos -c restaurants –out Restaurants.json
```

En el comando anterior, se utilizó –db para especificar la base de datos, -c para la colección y –out, para el archivo en el que se guardarán los datos.

Importar Base de Datos en MongoDB Importaremos los datos de Restaurante.json a una nueva base de datos llamada mibasededatos y a una colección llamada restaurantes. Utilizaremos el comando mongoimport de la siguiente forma:

```
# mongoimport –db mibasededatos –collection restaurantes –file Restaurants.json
```

Como podemos observar, la base de datos no la hemos tenido que crear, si no que al importar los datos se ha creado de forma automática.

Comprobar los datos importados Nos conectaremos a la base de datos:

```
$ mongo mibasededatos
```

Se nos habrá cambiado el prompt, por lo que estaremos conectados a la base de datos, contaremos los documentos de la colección importada:

```
> db.restaurantes.count()
```

y debe coincidir con los datos exportados. Y con esto veremos que hemos importado los mismos registros que hemos exportado.

2.6.5 Redis

El servidor de diccionarios remotos o Redis (Remote Dictionary Server) es un almacén de datos de valores clave que destaca por su gran velocidad y su facilidad de uso. Un almacén de valores clave es una base de datos NoSQL que utiliza un sistema de clave/valor para guardar la información. El proyecto se inició cuando Salvatore Sanfilippo, el desarrollador original de Redis, trataba de mejorar la escalabilidad de su empresa emergente italiana. A partir de ahí, desarrolló Redis, que ahora se utiliza como base de datos, caché, agente de mensajes y cola.

En Redis se almacenan los datos como un grupo de clave-valor, donde la clave es un identificador único y el valor puede contener cualquier tipo de dato como un texto, un archivo, un documento, o una imagen, entre otros. Redis utiliza la memoria para almacenar de forma rápida datos que pueden rescatarse de manera inmediata cuando son necesarios.

Redis ofrece tiempos de respuesta inferiores al milisegundo, lo que permite que se realicen millones de solicitudes por segundo para aplicaciones en tiempo real de la industria, como videojuegos, tecnología publicitaria, servicios financieros, sanidad e IoT. Hoy en día, Redis es uno de los motores de código abierto más populares en la actualidad, denominado la base de datos "preferida" por Stack Overflow durante cinco años consecutivos. Por su rápido rendimiento, Redis es una opción muy habitual en aplicaciones de almacenamiento en caché, administración de sesiones, videojuegos, tablas de clasificación, análisis en tiempo real, datos geoespaciales, servicios de vehículos compartidos, Chat/mensajería, Streaming de contenido multimedia y publicación/suscripción.

Para qué se Usa y sus Funcionalidades Redis tiene muchos usos debido a sus características especiales. Se trata de una gran alternativa para implementar una caché y reducir la latencia a la hora de acceder a los datos de un servidor, en salas de chat para ofrecer un alto rendimiento, en el desarrollo de videojuegos para ofrecer tablas de clasificación en tiempo real, como almacenamiento de sesiones de usuarios, como complemento en Streamings multimedia para guardar metadatos, o en Machine Learning para procesar grandes cantidades de información y ofrecerla en tiempo real.

Los principales beneficios que aporta Redis son:

- Guarda la información en memoria en lugar de en discos HDD o SSD como las bases de datos SQL.

- Cuenta con una amplia variedad de estructuras de datos que permite adaptarse a cualquier aplicación.
- El código para utilizar Redis es sencillo y simple, haciendo que sean necesarias menos líneas de código.
- Permite replicar los datos de forma asíncrona en distintos servidores, favoreciendo un mejor rendimiento, acelerando los tiempos de recuperación e incrementando el nivel de seguridad.
- Aporta un alto nivel de disponibilidad y escalabilidad.
- Es un proyecto de código abierto que se encuentra en constante evolución.

Instalación en Debian GNU/Linux de Redis Para instalar el paquete usamos:

```
# apt install redis-server
```

Una vez instalado el servicio de Redis es necesario acceder al mismo, iniciamos Redis usando:

```
$ redis-server
```

Para comprobar si Redis funciona correctamente, iniciamos la interfaz de comunicación con la base de datos:

```
$ redis-cli
```

La interfaz debería mostrar entonces la dirección IP y el puerto a través del cual se ejecuta Redis, a los que se puede enviar un ping de comprobación.

```
127.0.0.1:6397> ping
PONG
```

Si Redis responde, queda demostrado que el sistema de bases de datos se ha instalado correctamente. Ahora también se puede comprobar si se puede escribir texto.

```
127.0.0.1:6397> set test "OK!"
127.0.0.1:6397> get test
"OK!"
```

Configurar Redis Redis se instala, en un principio, con la configuración estándar, que luego puede modificarse con los comandos correspondientes.

```
127.0.0.1:6397> config get *
```

En la lista de ajustes para la configuración, las parejas de elementos se descomponen en dos posiciones, una debajo de la otra: al elemento *dbfilename* le corresponde, entonces, el valor *dump.rdb*. El asterisco que hemos usado para abrir la lista actúa de marcador de posición para un ajuste determinado de la lista. Cuando solo se quiere examinar un ajuste, el asterisco se reemplaza por el nombre del elemento, utilizando siempre para ello el que se encuentra en primera posición, que es la llave para el valor de configuración correspondiente.

```
127.0.0.1:6397> config get dbfilename
1) "dbfilename"
2) "dump.rdb"
```

Para cambiar una entrada en el archivo de configuración[3], se usa el comando *set*. Puede usarse, por ejemplo, para definir una contraseña:

```
127.0.0.1:6397> config set requirepass "password"
OK
```

Tras hacerlo, si solicitamos la contraseña con el comando get, se nos pedirá que la introduzcamos nosotros primero, al fin y al cabo, por algo le hemos puesto una. Para introducirla, usamos el comando authy, a continuación, consultamos la entrada en el archivo de configuración como acabamos de hacer.

```
127.0.0.1:6397> auth "password"
127.0.0.1:6397> config get requirepass
1) "requirepass"
2) "password"
```

[3]Existen más formas de hacer más segura tu base de datos. En la página web oficial de Redis, los desarrolladores resumen diversos puntos al respecto.

En realidad, Redis guarda todos los datos en la memoria principal. Teniendo esto en cuenta, para lograr la persistencia de los datos, se puede almacenar una copia (*snapchot*) de la base de datos en el disco duro, que se ubicará en el archivo *dump.rdb*.

```
127.0.0.1:6397> save
```

Con el comando *save*, se crea manualmente una copia, pero también pueden programarse para que se realicen de forma automática.

```
127.0.0.1:6397> save 60 10
```

En este ejemplo, hemos asignado dos parámetros al comando: ahora, se creará una copia cada 60 segundos si ya se han producido 10 cambios en dicho intervalo de tiempo.

Sin embargo, el comando *save* no se recomienda mientras el sistema esté en funcionamiento, ya que impide que los clientes accedan a la base de datos. En estos casos es más conveniente *bgsave*, que realiza el proceso en un segundo plano.

Además de la posibilidad de hacer una copia de la base de datos, también existe el modo *append only file (AOF)*, en el que Redis guarda en un archivo cada acción realizada. Gracias a este archivo, si el servidor se averiará inesperadamente, se podría averiguar qué fue lo último que se hizo. Para activar el modo AOF[4], debe modificarse el archivo de configuración.

```
127.0.0.1:6397> config set appendonly yes
```

Crear Entradas Una vez hayas configurado Redis, ya puedes trabajar con la base de datos. Dispones para ello de varios tipos distintos de datos y de comandos.

[4] Si tus datos requieren la máxima seguridad, deberías activar el modo AOF y, además, hacer copias regulares de la base de datos: así será prácticamente imposible que pierdas tus datos. Estos procesos, no obstante, ralentizan en cierta medida el funcionamiento de la base de datos.

Strings Lo más fácil es crear una String[5], es decir, una cadena o secuencia de elementos. Para ello, utiliza el comando *set*.

```
127.0.0.1:6397> set foo "bar"
127.0.0.1:6397> set value 1
```

Si se solicitan ahora las entradas foo y value mediante el comando *get*, se mostrarán los valores correspondientes.

```
127.0.0.1:6397> get foo
"bar"
127.0.0.1:6397> get value
"1"
```

El comando para borrar una entrada es *del*.

```
127.0.0.1:6397> del foo
(integer) 1
127.0.0.1:6397> get foo
(nil)
```

Si no quieres crear muchas entradas usando una fila nueva cada vez, puedes usar la función avanzada *mset*. Para solicitar los valores de varios campos a la vez, también existe el comando *mget*.

```
127.0.0.1:6397> mset foo1 "bar1" foo2 "bar2" foo3 "bar3"
OK
127.0.0.1:6397> mget foo1 foo2 foo3
1) "bar1"
2) "bar2"
3) "bar3"
```

[5]No importa si se introducen los valores entre comillas o no. Para hacer el código más legible, puede ponerse el texto entre comillas y los valores numéricos sin ellas.

Listas Con Redis se pueden usar, además, otros tipos de datos. Algunos de los más populares para trabajar con la base de datos son, por ejemplo, las listas y los sets. Ambos son conjuntos de valores, pero, mientras que los sets no tienen un orden concreto, los valores de las listas están numerados. En una lista se pueden añadir, solicitar o borrar entradas.

```
127.0.0.1:6397> lpush mylist foo
(integer) 1
127.0.0.1:6397> lpush mylist bar
(integer) 2
127.0.0.1:6397> lrange mylist 0 10
1) "foo"
2) "bar"
127.0.0.1:6397> linsert mylist before "bar" "test"
(integer) 3
127.0.0.1:6397> lrange mylist 0 10
1) "foo"
2) "test"
3) "bar"
127.0.0.1:6397> lrem mylist 0 foo
(integer) 1
127.0.0.1:6397> lrange mylist 0 10
1) "test"
2) "bar"
```

En este ejemplo hemos añadido en primer lugar dos elementos a una lista (*lpush*) y luego hemos solicitado que se muestren. Con el comando *lrange* se indica qué segmento debe mostrarse (aquí del 0 al 10, pero pueden usarse también números negativos). A continuación, mediante el comando *linsert* hemos añadido un valor nuevo delante de uno que ya existía (también podría usarse *after*), con lo cual hemos cambiado la numeración. El comando *lrem* permite borrar de la lista entradas con un valor específico.

Sets Para los sets, Redis utiliza otros comandos, pero con resultados muy similares:

```
127.0.0.1:6397> sadd myset "foo"
(integer) 1
```

```
127.0.0.1:6397> sadd myset "bar"
(integer) 1
127.0.0.1:6397> smembers myset
1) "bar"
2) "foo"
127.0.0.1:6397> sismember myset "bar"
(integer) 1
127.0.0.1:6397> srem myset "bar"
(integer) 1
127.0.0.1:6397> smembers myset
1) "foo"
```

Con el comando *sadd* también se pueden integrar varios elementos en el set si se introducen en el comando uno detrás de otro. Para visualizar el set, basta con usar el comando *smembers* y el nombre del set en cuestión. El comando *sismember* permite, además, buscar una entrada concreta. De manera análoga a la lista, con *srem* se pueden borrar entradas sueltas.

Sin embargo, Redis también ofrece a los usuarios la posibilidad de utilizar sets en un formato ordenado.

```
127.0.0.1:6397> zadd mysortedset 1 "foo"
(integer) 1
127.0.0.1:6397> zadd mysortedset 2 "bar"
(integer) 1
127.0.0.1:6397> zadd mysortedset 2 "foobar"
(integer) 1
127.0.0.1:6397> zrange mysortedset 0 10
1) "foo"
2) "bar"
3) "foobar"
```

Para añadir elementos se utiliza, en este caso, el comando *zadd* y un *score* o puntaje. Mientras que los valores propiamente dichos no pueden aparecer más de una vez, con un *score* se puede indicar un mismo valor varias veces. El score no es, por lo tanto, una numeración directa dentro del set, sino una ponderación, de manera que todas las entradas con el puntaje o score2 aparecerán tras los que tengan el score1. Con el comando *zrange* se pueden visualizar todos los elementos o los que se seleccionen.

Hashes Un tipo especial de datos son los hashes: entradas individuales compuestas de varios valores, de manera similar a los sets y las listas, pero en los que cada valor va acompañado de una clave, formando así los llamados pares clave-valor o key-value.

```
127.0.0.1:6397> hset user1 name "bob" email "bob@example.com"
password "rK87_x"
OK
127.0.0.1:6397> hget user1 name
1) "bob"
127.0.0.1:6397> hgetall user1
1) "name"
2) "bob"
3) "email"
4) "bob@example.com"
5) "password"
6) "rK87_x"
127.0.0.1:6397> hvals user1
1) "bob"
2) "bob@example.com"
3) "rK87_x"
127.0.0.1:6397> hkeys user1
1) "name"
2) "email"
3) "password"
> hdel user1 password
(integer) 1
127.0.0.1:6397> hgetall user1
1) "name"
2) "bob"
3) "email"
4) "bob@example.com"
127.0.0.1:6397> del user1
(integer) 1
127.0.0.1:6397> hgetall user1
(empty list or set)
```

En este ejemplo, hemos usado *hset* para crear un hash con el nombre user1 y tres campos. Mediante el comando *hget* podemos solicitar el valor de cada campo. Para que se muestren todos, se puede usar *hgetall*. Otras opciones para visualizar valores son *hvals* (muestra todos los valores guardados en el hash) y *hkeys* (muestra todas las claves guardadas en el hash). Con *hdel* se pueden borrar valores sueltos, mientras que con *del*, como ya hemos visto, se borra el hash entero[6].

Otras opciones Naturalmente, con Redis no solo se pueden crear entradas en una base de datos, sino que también pueden asignarse propiedades concretas a los datos. En este sentido, pueden resultar muy útiles, por ejemplo, los comandos de incremento y decremento.

```
127.0.0.1:6397> set foo 1
OK
127.0.0.1:6397> get foo
"1"
127.0.0.1:6397> incr foo
(integer) 2
127.0.0.1:6397> incr foo
(integer) 3
127.0.0.1:6397> get foo
"3"
127.0.0.1:6397> decr foo
(integer) 2
127.0.0.1:6397> get foo
"2"
```

Con la ayuda de estas funciones, se pueden incrementar o reducir los valores en una unidad. A veces, en cambio, se quieren introducir valores que solo permanezcan en la base de datos durante cierto tiempo: para ello existe la función expire.

```
127.0.0.1:6397> set foo "bar"
OK
127.0.0.1:6397> expire foo 100
(integer) 1
```

[6] El comando flushall sirve para borrar todas las entradas de la base de datos.

```
127.0.0.1:6397> ttl foo
(integer) 50
127.0.0.1:6397> ttl foo
(integer) -50
127.0.0.1:6397> get foo
(nil)
```

El comando expire requiere una indicación de tiempo en segundos. En este ejemplo, hemos decidido que la entrada debe durar 100 segundos. Una vez transcurrida la mitad del tiempo, hemos usado el comando *ttl* para solicitar el time-to-live, es decir, el tiempo restante. Si esperamos aún más, el TTL pasará a ser negativo a partir del momento en el que la entrada ya haya desaparecido.

El comando *setex* permite asignar un TTL a una entrada de la base de datos ya desde su creación.

```
127.0.0.1:6397> setex foo 100 "bar"
OK
```

Una vez se ha creado una entrada, esta se puede ampliar: el comando *append* añade otro valor al que ya existía.

```
127.0.0.1:6397> set foo "Hello"
OK
127.0.0.1:6397> append foo " World"
(integer) 11
127.0.0.1:6397> get foo
"Hello World"
127.0.0.1:6397> set bar 5
OK
127.0.0.1:6397> append bar 10
(integer) 3
127.0.0.1:6397> get bar
"510"
```

Como se puede ver, al solicitar los valores correspondientes, aparecen los nuevos elementos simplemente tras los que ya estaban. Si la entrada en cuestión aún no existe, *append* realiza entonces la misma función que *set*.

Además, también se puede cambiar el nombre de las entradas usando el comando *rename*.

```
127.0.0.1:6397> set foo 100
OK
127.0.0.1:6397> rename foo bar
OK
127.0.0.1:6397> get foo
(nil)
127.0.0.1:6397> get bar
"100"
```

Existen muchos otros comandos para trabajar con Redis. En la descripción oficial se pueden consultar los detalles de todos los comandos disponibles.

2.7 LAMP y LEMP

LAMP y LEMP son acrónimos usados para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

Para LAMP:

- Linux, el sistema operativo
- Apache, el servidor de Web
- MySQL/MariaDB, el gestor de bases de datos
- PHP, el lenguaje de programación

Para LEMP

- Linux, el sistema operativo
- Nginx, el servidor de Web
- MySQL/MariaDB, el gestor de bases de datos
- PHP (Hypertext Preprocessor), el lenguaje de programación

La combinación de estas tecnologías es usada principalmente para definir la infraestructura de un servidor Web, utilizando un paradigma de programación para el desarrollo del sistema. A pesar de que el origen de estos programas de código abierto no fue específicamente diseñado para trabajar entre sí, la combinación se popularizó debido al bajo costo de adquisición y ubicuidad de sus componentes.

Para instalar apache2 usamos:

```
# apt install apache2 libapache2-mod-evasive apachetop
```

Para instalar Nginx usamos:

```
# apt install nginx
```

En este caso tenemos dos opciones, instalar MariaDB que ya es parte integral de múltiples aplicaciones dentro de GNU/Linux o usar MySQL, en cada caso el uso es prácticamente el mismo.

Para instalar el paquete MariaDB usamos:

```
# apt install mariadb-server mariadb-client
```

y para configurarlo, usamos:

```
# mysql
> use mysql
> update user set plugin=" where user='root';
> alter user root@localhost identified by 'XXXXXXX';
> flush privilegies;
> exit
```

y para usarlo

```
$ mysql -u root -p
```

Para instalar el paquete MySQL usamos:

```
# apt install mysql-common mysql-client mysql-server mytop
mysql-admin
```

y para configurarlo (las bases de datos se guardan en /var/lib/mysql), usamos:

```
# mysql_secure_installation
```

o

```
# mysql
```

en caso de no crear contraseña, podemos ingresar así:

```
$ mysql -u root -p
```

una vez en la consola del cliente de MySQL creamos un nuevo usuario

```
> create user nombre identified by 'XXXXX';
```

y a continuación le damos privilegios totales

```
> gran all privileges on *.* to nombre with grant option;
> flush privileges;
> exit;
```

si requerimos las pruebas de eficiencia, podemos instalar:

```
# apt install ferret mysql-workbench mysql-workbench-data
mycli
```

Para instalar la interface para PHP, usamos:

```
# apt install php libapache2-mod-php php-mysql php-gd ph-
pmyadmin
```

3 Lenguajes de Programación y Bases de Datos SQL

Quizás te estés preguntando, ¿por qué debería preocuparme por conectar un lenguaje de programación como Java o Python y una base de datos SQL?. Hay muchos casos de uso en los que alguien querría conectar Java o Python a una base de datos SQL. Es posible que se esté trabajando en una aplicación Web. En este caso, necesitaríamos conectar una base de datos SQL para poder almacenar los datos provenientes de la aplicación Web. Quizás estemos trabajando en ingeniería de datos y necesitemos crear una canalización ETL automatizada. Conectar Java o Python a una base de datos SQL nos permitirá utilizar las capacidades de automatización del lenguaje de programación. También podremos comunicar entre diferentes fuentes de datos, no tendremos que cambiar entre diferentes lenguajes de programación.

Conectar Java o Python y una base de datos SQL también hará que nuestro trabajo de ciencia de datos sea más conveniente. Podremos utilizar nuestras habilidades en el lenguaje de programación para manipular datos de una base de datos SQL. No necesitaremos un archivo *CSV*.

Bases de Datos Relacionales las bases de datos de Software libre más utilizadas son PostgreSQL, MariaDB y MySQL que rivalizan con la base de datos Oracle comercial. PostgreSQL es adecuada incluso para organizaciones y proyectos de gran tamaño.

Otras bases de datos relevantes son H2 una base de datos implementada en Java con características avanzadas que es posible utilizar para los tests de integración al no requerir de un servidor y ser posible ejecutarla en memoria. Para realizar las pruebas de integración utilizando la misma base de datos que en producción es posible utilizar *Testcontainers* que utiliza Docker para iniciar una instancia de la base de datos en un contenedor. La conexión a la base de datos por seguridad requiere de un usuario y contraseña que la aplicación ha de conocer, para aún mayor seguridad es posible generar las credenciales de conexión a la base de datos de forma dinámica por *Vault* en una aplicación de *Spring*.

El problema de Seguridad de SQL Injection es un grave problema de seguridad que afecta a las aplicaciones que construyen sentencias de forma dinámica a partir de datos provenientes de origen no confiable. Un origen

no confiable es cualquier dato proveniente de forma externa a la aplicación, en el caso de las aplicaciones Web o servicios *REST* es un parámetro de la petición o un dato de un *JSON*.

El SQL injection es un grave problema de seguridad ya que permite al atacante tener acceso a datos de la base de datos, obtener acceso con una cuenta de otro usuario o realizar operaciones sobre los datos de forma no autorizada. El problema se produce en la construcción de forma dinámica de la sentencia SQL mediante la concatenación de cadenas y datos de origen no confiable.

3.0.1 Java y el Acceso a Base de Datos Relacionales

Java ofrece soporte para las bases de datos relacionales desde prácticamente las primeras versiones del JDK hasta día de hoy incorporando un conjunto de clases en el paquete *java.sql* en la denominada *API* en Java de *Java Database Connectivity* o *JDBC*.

Las principales clases de la *API* de *JDBC* son las clases *Statement* y *PreparedStatement* que representan la sentencia SQL ya sea de inserción, actualización, eliminación o consulta así como las sentencias *DDL* para la creación de tablas, índices, campos o procedimientos almacenados. Normalmente se utiliza la clase *PreparedStatement* ya que tiene beneficios en cuanto a rendimiento al ejecutarse de forma repetida y utilizada correctamente permite evitar el grave problema de seguridad de SQL injection común en las aplicaciones que construyen de forma dinámica con concatenaciones sentencias SQL utilizando datos procedentes de fuentes no confiables como parámetros de una petición *HTTP*, de *JSON* u otras fuentes externas a la aplicación.

La clase *ResultSet* es la clase que proporciona el acceso a los datos cuando se ejecuta una sentencia SQL de consulta, la clase se itera en los resultados y se obtienen los datos según los nombres o índices asignados en la consulta para las columnas.

La clase *Connection* representa una conexión a una base de datos, hay que crear una conexión ya que las bases de datos trabajan con una arquitectura de cliente y servidor, la aplicación actúa de cliente y la base de datos actúa de servidor. A través de la clase *Connection* se inicia una transacción y finaliza con el *commit* o el *rollback*. Para crear la conexión de la base de datos se proporcionan las credenciales de usuario y contraseña.

Las bases de datos relacionales soportan transacciones para proporcionar

las propiedades *ACID* para lo que se utilizan las transacciones. Atomicidad donde un grupo de operaciones individuales se ejecutan todas o ninguna, consistencia mediante la cual los cambios son válidos según las reglas incluyendo restricciones, cambios en cascada, y disparadores, aislamiento donde los cambios de una transacción no se ven afectados por los cambios realizados en otras transacciones concurrentes y finalmente durabilidad que garantiza que en caso de completarse la transacción perdura en el tiempo aún cuando el sistema sufra un fallo posterior.

Crear una conexión a una base de datos es costoso, para evitar incurrir en este tiempo de creación y destrucción de conexiones o limitar el número de conexiones que una aplicación utiliza como máximo las aplicaciones utilizan un pool de conexiones. Las conexiones se crean al iniciar la aplicación o bajo demanda según se van necesitando más hasta el límite máximo definido. Cuando la aplicación necesita una conexión la obtiene de forma rápida del pool de conexiones y cuando termina de utilizarla la devuelve al pool de conexiones para que sea reutilizada en posteriores usos.

Cada base de datos utiliza un protocolo diferente de comunicación con los clientes por lo que es necesario un componente que abstrae de las peculiaridades de cada base de datos y proporcione un marco común de trabajo independiente de cada base de datos. Cada base de datos requiere de un *Driver* compatible también con la versión de la base de datos. Generalmente, son los desarrolladores de la propia base de datos los que proporcionan un *Driver* específico adecuado para el acceso a la base de datos desde Java que cumple con las *APIs* de *JDBC*.

Ejemplo de Conexión y Consulta a un Base de Datos Relacional con la API de Java En este ejemplo de código se muestra el uso de las clases fundamentales de Java para usar una base de datos relacional. El primer paso es establecer una conexión con la base de datos, en este caso usando la base de datos H2 en memoria.

Posteriormente se ejecuta una sentencia *DDL* para crear una tabla, se insertan varias filas con la sentencia *insert* y se obtienen los datos de la tabla con una sentencia *select*. Después de las inserciones se realiza un commit que completa una transacción en la base de datos, si en vez del *commit* se hiciese un *rollback* al obtener los datos con la consulta posterior la tabla aparecería vacía.

Las clases *Connection*, *Statemente*, *PreparedStatement* y *ResultSet* al fi-

nalizar su uso hay que invocar su método *close* para liberar los recursos que tienen reservados, especialmente en el caso de las conexiones ya que son un recurso limitado. Estas clases implementan la interfaz *AutoCloseable* con lo que son adecuadas para las sentencias *try-with-resources* de Java.

Establecer la Conexión a la Base de Datos por defecto después de cada sentencia Java emite un *commit*, esto no es lo deseado en el caso de querer agrupar la ejecución de varias sentencias en una transacción, para evitarlo hay que usar la opción *setAutoCommit* a *false*.

```
package io.github.picodotdev.blgbitix.javasql;
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.DecimalFormat;
import java.util.Locale;
public class Main {
    public static void main(String[] args) {
        DriverManager.drivers().forEach(d -> {
            System.out.printf("Driver: %s%n", d.getClass().getName());
        });
        try (Connection connection = DriverManager.getConnection("jdbc:h2:mem:database",
"sa", "")) {
            connection.setAutoCommit(false);
        ...
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Ejecutar una Sentencia con Statement

```
Statement statement = connection.createStatement();
statement.execute("CREATE TABLE product(id INT IDENTITY NOT NULL PRIMARY KEY,
name VARCHAR(255), price DECIMAL(20, 2))");
```

Ejecutar una Sentencia con PreparedStatement

```
PreparedStatement preparedStatement = connection.prepareStatement("INSERT INTO product
(name, price) values (?, ?)", new String[] { "id" });
preparedStatement.setString(1, "PlayStation 5");
preparedStatement.setBigDecimal(2, new BigDecimal("499.99"));
preparedStatement.executeUpdate();
ResultSet resultSet1 = preparedStatement.getGeneratedKeys();
while (resultSet1.next()) {
    System.out.printf("Primary key: %s%n", resultSet1.getLong(1));
}
resultSet1.close();
preparedStatement.setString(1, "Xbox Series X");
preparedStatement.setBigDecimal(2, new BigDecimal("499.99"));
preparedStatement.executeUpdate();
ResultSet resultSet2 = preparedStatement.getGeneratedKeys();
while (resultSet2.next()) {
    System.out.printf("Primary key: %s%n", resultSet2.getLong(1));
}
resultSet2.close();
connection.commit();
```

Ejecutar una Consulta

```
PreparedStatement preparedStatement = connection.prepareStatement("SELECT id, name, price
FROM product");
ResultSet resultSet = preparedStatement.executeQuery();
while (resultSet.next()) {
    System.out.printf("Product (id: %s, name: %s, price: %s)%n", resultSet.getLong(1), result-
Set.getString(2),
    DecimalFormat.getCurrencyInstance(new Locale("es", "ES")).format(resultSet.getBigDecimal(3)));
}
resultSet.close();
```

Cerrar la Conexión de Forma Explícita

```
connection.close();
```

Resultado el resultado del programa en la terminal es el siguiente.

```
Driver: org.h2.Driver
Primary key: 1
Primary key: 2
Product (id: 1, name: PlayStation 5, price: 499,99 EUR)
Product (id: 2, name: Xbox Series X, price: 499,99 EUR)
```

Dependencia con el Driver de la Base de Datos en el archivo de construcción hay que añadir la dependencia que contiene el Driver para la base de datos a conectarse.

```
plugins {
    id 'application'
}
repositories {
    mavenCentral()
}
dependencies {
    implementation 'com.h2database:h2:1.4.200'
}
application {
    mainClass = 'io.github.picodotdev.blgbitix.javasql.Main'
}
```

El problema de Seguridad de SQL Injection Las siguientes sentencias SQL sufren del problema de SQL injection, en la primera un atacante puede obtener un dato de cualquier columna de la tabla y con la segunda ejecutar una sentencia en este caso para eliminar todas las filas de cualquier tabla.

En la primera cambiando el valor de column se obtiene el dato de cualquier columna de la tabla, por ejemplo el campo password.

```
"SELECT id, " + column + " FROM users WHERE user_id = " + userId
```

```
"SELECT id, password FROM users WHERE user_id = 1
```

En esta sentencia se finaliza la sentencia original con ; y se inicia otra lo que provoca la eliminación de una tabla, con un valor para user_id especialmente construido para ejecutar la sentencia maliciosa de eliminación de la tabla.

```
"SELECT * FROM users WHERE user_id = " + userId
"SELECT * FROM users WHERE user_id = 105; DROP TABLE users;"
```

La solución al problema de seguridad de SQL *injection* en Java es no construir la sentencia de forma dinámica mediante concatenación de cadenas utilizando la clase *PreparedStatement* con argumentos para los datos que se inserten en la sentencia SQL.

```
PreparedStatement preparedStatement = connection.prepareStatement("SELECT * FROM users
WHERE user_id = ?"});
preparedStatement.setInt(1, 1);
preparedStatement.executeUpdate();
```

Librerías de Persistencia en Java Habitualmente no se utilizan directamente las clases de la API de Java sino que se utilizan otras librerías de más alto nivel. Una de las más conocidas es *Hibernate*, es un ORM que proporciona acceso a los datos con una correspondencia entre el modelo relacional de las bases de datos y el modelo de objetos de Java. La aplicación trabaja con objetos y relaciones entre los objetos e *Hibernate* se encarga de transformar esos objetos en el modelo relacional de la base de datos, la aplicación no ejecuta sentencias SQL de forma directa sino que es *Hibernate* el encargado de emitir las sentencias adecuadas según los cambios realizados en los datos. Es la implementación más utilizada de ORM en Java para la especificación JPA.

- Aplicación de ejemplo con Hibernate, jOOQ y Liquibase

Spring Data es una capa de abstracción para el acceso a datos ya sean de un modelo relacional, de bases de datos NoSQL o algunos otros sistemas de datos. Spring Data hace más sencillo el acceso a los datos utilizando de forma subyacente JDBC o JPA. Spring Data proporciona algunas clases e interfaces que la aplicación implementa.

jOOQ también es otra librería de acceso a bases de datos relacionales, proporciona DSL para la construcción de sentencias SQL mediante un API Java. A diferencia de JPA con su lenguaje JPQL, jOOQ soporta características avanzadas del lenguaje SQL como windows functions. Otra ventaja de jOOQ es que al utilizar su DSL el compilador de Java realiza validación de tipos y comprobaciones en la sintaxis de la construcción de la SQL.

- Alternativa a Hibernate u ORM y ejemplo de jOOQ

Algunas aplicaciones combinan el uso de varias de estas librerías en la misma aplicación según el caso, por ejemplo utilizando Hibernate para el modelo de escritura y jOOQ o Spring Data para el modelo de lectura. También es posible utilizar jOOQ para generar las sentencias SQL y posteriormente ejecutarlas con Hibernate o Spring Data.

Liquibase es otra librería de utilidad que permite lanzar scripts de migración con sentencias SQL para realizar cambios en la base de datos como modificar el esquema de tablas, insertar, actualizar o eliminar datos. Es necesario si los cambios en el código en una nueva versión de la aplicación requiere cambios en el esquema de la base de datos. Estas no son las únicas librerías existentes pero sí son de las más conocidas y utilizadas.

3.0.2 Python y el Acceso a Base de Datos Relacionales

Las bases de datos de Python y SQL se conectan a través de bibliotecas personalizadas de Python. Podemos importar estas bibliotecas en su secuencia de comandos de Python. Las bibliotecas de Python específicas de la base de datos sirven como instrucciones complementarias. Estas instrucciones guían a nuestra computadora sobre cómo puede interactuar con un base de datos SQL.

Cómo Configurar el Proyecto Tomemos una base de datos PostgreSQL, *AWS Redshift*, por ejemplo. Primero, necesitamos importar la biblioteca *psycopg*. Es una biblioteca de Python universal para bases de datos de por ejemplo PostgreSQL:

```
#Library for connecting to AWS Redshift
import psycopg
#Library for reading the config file, which is in JSON
import json
#Data manipulation library
import pandas as pd
```

Notarás que también importamos las bibliotecas *JSON* y *pandas*. Importamos *JSON* porque crear un archivo de configuración *JSON* es una forma segura de almacenar las credenciales de su base de datos. ¡No queremos que nadie más los mire!.

La biblioteca de *pandas* nos permitirá utilizar todas las capacidades estadísticas de *pandas* para su secuencia de comandos de Python. En este caso, la biblioteca permitirá que Python almacene los datos que devuelve su consulta SQL en un marco de datos.

A continuación, necesitamos acceder al archivo de configuración. La función *json.load()* lee el archivo *JSON* para que pueda acceder a las credenciales de la base de datos en el siguiente paso:

```
config_file = open(r"C:\Users\yourname\config.json")
config = json.load(config_file)
```

Ahora que la secuencia de comandos de Python puede acceder a el archivo de configuración *JSON*, queremos crear una conexión de base de datos. Deberá leer y usar las credenciales de su archivo de configuración:

```
con = psycopg2.connect(dbname= "db_name", host=config[hostname], port = config["port"],user=config["user_id"],
password=config["password_key"])
cur = con.cursor()
```

¡Acabamos de crear una conexión a la base de datos! Cuando importamos la biblioteca *psycopg*, tradujo el código Python que escribió anteriormente para hablar con la base de datos PostgreSQL (AWS Redshift).

En sí mismo, AWS Redshift no entendería el código anterior. Pero debido a que importó la biblioteca *psycopg*, ahora habla un idioma que AWS Redshift puede entender. Lo bueno de Python es que tiene bibliotecas para SQLite, MySQL y PostgreSQL. Podremos integrar las tecnologías con facilidad.

Cómo Escribir una Consulta SQL siéntase libre de descargar European Soccer Data a su base de datos PostgreSQL. Usaré sus datos para este ejemplo.

La conexión de base de datos que creó en el último paso le permite escribir SQL para luego almacenar los datos en una estructura de datos compatible con Python. Ahora que hemos establecido una conexión a la base de datos, podemos escribir una consulta SQL para comenzar a extraer datos:

```
query = "SELECT *
FROM League
JOIN Country ON Country.id = League.country_id;"
```

Sin embargo, el trabajo aún no está terminado. Necesitamos escribir código Python adicional que ejecute la consulta SQL:

```
#Runs your SQL query
execute1 = cur.execute(query)
result = cur.fetchall()
```

Luego, debemos almacenar los datos devueltos en un marco de datos de pandas:

```
#Create initial dataframe from SQL data
raw_initial_df = pd.read_sql_query(query, con)
print(raw_initial_df)
```

Ahora deberíamos obtener un marco de datos de pandas (raw_initial_df).

3.1 Seguridad en Bases de Datos

Las fugas de datos son extremadamente prominentes en el mundo cibernético debido a la falta de una implementación de seguridad adecuada. Proteger las bases de datos es una práctica esencial para garantizar que se eviten consecuencias como la pérdida de datos incluso por acceso no autorizado o el tiempo de inactividad del sistema.

Hay muchos desafíos cuando se trata de la seguridad de la base de datos, ya que cuanto más accesible es una base de datos, menos segura es. Algunos de los desafíos de seguridad comunes cuando se trata de la seguridad de la base de datos incluyen errores humanos, Malware, seguridad de la ubicación física y vulnerabilidades de Software. Sin embargo, hay muchos pasos que se pueden tomar para garantizar que una base de datos esté protegida y, por lo tanto, evitar cualquiera de los problemas mencionados anteriormente. Además, el cifrado de la base de datos es un paso esencial para proteger todos los datos.

El Software libre tiene una variedad de herramientas que pueden usarse para garantizar la seguridad de la base de datos. Brevemente presentaremos algunas de estas herramientas que se pueden usar para la evaluación de bases de datos y otras que ayudarán a proteger estas bases de datos.

Prácticas Recomendadas para Bases de Datos Algunas de las amenazas de seguridad comunes incluyen ataques DDoS, ataques de inyección SQL, ataques Man in the Middle, contraseñas débiles, corrupción de datos, condición de carrera, mala gestión del acceso a la cuenta, así como vulnerabilidades ocasionales. Afortunadamente, hay muchas prácticas que se pueden adoptar para poder evitar estos problemas de seguridad.

3.1.1 Cifrado

El cifrado es una de las prácticas más efectivas e importantes que siempre debe usarse al almacenar datos. Algunas bases de datos pueden cifrar los datos directamente o cifrar el contenedor en el que se encuentran. Para MySQL, algunas de las herramientas que se pueden usar para el cifrado son el cifrado asimétrico, el cifrado simétrico, la generación de claves públicas/privadas, las firmas digitales y los datos transparentes. En cuanto a MariaDB, hay cifrado de datos en reposo, cifrado de datos en tránsito, certificados TLS/SSL y Cipher Block Chaining.

Además, para garantizar la seguridad de la base de datos, otras prácticas útiles a tener en cuenta son el acceso condicional y la auditoría. Para el acceso condicional, siempre es útil aplicar el privilegio mínimo. Es muy importante otorgar sólo la menor cantidad de privilegios necesarios para que un usuario pueda completar el trabajo. En cuanto a la auditoría, existe una variedad de herramientas que se pueden usar para auditar bases de datos como ClusterControl y Cloud SQL para MySQL, mientras que MariaDB tiene su propio complemento de auditoría que se puede usar.

3.1.2 Restringir el Acceso y Personalizar la Configuración Predeterminada

Los próximos pasos a seguir para proteger una base de datos MySQL deben incluir la eliminación de cuentas predeterminadas, asignaciones de puertos y la personalización de la configuración predeterminada. Es muy importante eliminar la base de datos de prueba ya que todos los usuarios tienen acceso completo a ella.

El acceso remoto también debe estar restringido. El acceso a la red solo debe permitir el mínimo requerido, como es el caso de muchas prácticas de seguridad, incluido el enfoque de privilegio mínimo mencionado. Todo acceso remoto debe ser monitoreado y controlado.

3.1.3 Asegúrese de que los Servidores sean Físicamente Seguros

Un paso importante que con frecuencia se ignora es asegurarse de que los servidores estén físicamente seguros. Si está utilizando proveedores de la nube como AWS, GCP o Azure, esto está cubierto automáticamente.

Uno de los pasos más esenciales para asegurar cualquier cosa, no solo las bases de datos, es garantizar el uso de credenciales de autenticación sólidas. Las contraseñas débiles pueden ser fácilmente forzadas otorgando a los atacantes acceso a los servidores. Por lo tanto, debemos asegurarnos de que solo se utilicen contraseñas seguras que contengan una combinación de caracteres en mayúsculas y minúsculas, números y caracteres especiales.

Además, authn y authz, que se refieren a autenticación y autorización, nos permiten controlar el acceso de los usuarios a los elementos dentro de la base de datos.

3.1.4 Usar Herramientas de Evaluación de la Seguridad de la Base de Datos

Un paso fundamental para proteger una base de datos es utilizar herramientas de evaluación de la seguridad de la base de datos. Estas herramientas revisan el entorno de una base de datos y reconocen las amenazas que están presentes en el entorno. Además de resaltar las vulnerabilidades existentes presentes en el entorno, estas herramientas también evalúan las medidas de seguridad implementadas para verificar su efectividad. Presentamos cinco de las mejores herramientas de evaluación de seguridad de bases de datos de Linux para garantizar que tenga una base de datos segura.

SQLMap es una herramienta de inyección SQL de código abierto basada en Python. Esta es una gran herramienta para usar, ya que ofrece una variedad de características. Estas características incluyen la detección y explotación de diferentes tipos de ataques de inyección, Hash de contraseñas de fuerza bruta, carga o descarga de archivos de la base de datos, se pueden usar para conectarse directamente a la base de datos a través de la inyección de SQL y se pueden personalizar.

SQLNinja es una herramienta basada en Perl que se puede utilizar con aplicaciones que utilizan servidores Microsoft SQL como Backend para explotar las vulnerabilidades de inyección de SQL. Las características de SQL-

Ninja incluyen la capacidad de tomar huellas dactilares de servidores SQL remotos, extracción de datos, carga de ejecutables, explotación de vulnerabilidades de inyección para obtener acceso a la base de datos, contraseñas de fuerza bruta mediante ataques de diccionario y ataques de escalada de privilegios si la fuerza bruta tiene éxito.

BBQSQL como SQLMap, es una herramienta de código abierto basada en Python. BBQSQL se usa para explotar las vulnerabilidades de inyección de SQL y se enfoca en la vulnerabilidad ciega de SQL. Esta herramienta requiere cierta información para ser utilizada, que incluye URL, método HTTP, Cookies, etc. Sin embargo, es una gran herramienta ya que es muy rápida para encontrar y explotar vulnerabilidades, es personalizable, realiza la validación de entrada en todas las opciones de configuración e incluso puede parchear las vulnerabilidades detectadas

JSQL Injection como su nombre lo indica, también es una herramienta de inyección SQL basada en Java. Se puede utilizar para encontrar y explotar vulnerabilidades de inyección. Se puede utilizar en 33 motores de bases de datos, realizar varios tipos de ataques de inyección, crear un Shell Web y un Shell SQL en un servidor remoto, utilizar Hashes de fuerza bruta y mucho más.

OScanner es un marco de evaluación de Oracle basado en Java. Se puede utilizar para enumeración de SID, contraseñas de fuerza bruta, versión de enumeración, roles de cuenta, privilegios y políticas de contraseña, información de auditoría y enlaces a bases de datos. OScanner da los resultados en un formato de árbol java gráfico.

Estas herramientas que hemos mencionado son fundamentales para garantizar que su base de datos esté segura. Se pueden usar para verificar qué pasos se deben tomar para garantizar que las bases de datos sean seguras, y también se pueden usar después de tomar estos pasos para evaluar la efectividad del procedimiento seguido.

3.1.5 Pruebas de Penetración

Cuando se trabaja con bases de datos que envían y reciben peticiones desde páginas Web o aplicaciones en internet, es necesario hacer pruebas de pe-

netración (o Pentest) que nos permiten analizar los riesgos de los programas desarrollados y del sistema completo antes de entrar en la fase de producción.

Hay tres tipos de pruebas de penetración:

- Prueba de caja negra (Blackbox): Tratamos de atacar el sistema sin el conocimiento de la infraestructura y el funcionamiento del mismo. Esta es la forma menos efectiva de las pruebas Pentest.
- Pruebas de caja gris (Greybox): Tratamos de atacar el sistema con poco conocimiento de la infraestructura y el funcionamiento interno del mismo. Las pruebas de caja gris son más efectivas que las de caja negra.
- Pruebas de caja blanca (Whitebox): Tratamos de atacar el sistema con toda la información sobre la infraestructura y código fuente. Esta es la forma más efectiva de las pruebas Pentest.

Las pruebas de penetración tienen como objetivo prevenir ciberataques. ¿Qué es un ataque cibernético?

- Ataque cibernético: Es el que se ejecuta a través del espacio cibernético con el objetivo de dislocar, dañar o apropiarse de la gestión de la infraestructura de una empresa o dependencia gubernamental.
- Superficie del ataque cibernético: Es el ámbito en el cual se ataca a un sistema de cómputo. Esto puede ser muy amplio y abarca las irrupciones en las redes, los protocolos, el sistema operativo a las aplicaciones.
- Exploit: Es un ataque contra los elementos vulnerables de un sistema llevado a cabo mediante el uso de Software, un comando o una metodología específica. Un Exploit puede ser un elemento Malware; para el Hackeo ético un Exploit es una forma de identificar vulnerabilidades del sistema.

Debido a que se practicará con aplicaciones inseguras, es importante crear un ambiente seguro, pero siempre es deseable hacer pruebas con toda la infraestructura computacional disponible sobre la que se montará el sistema desarrollado, pero en muchas ocasiones esto no es posible o viable, por ello

se opta por generar un sistema de prueba aislado y seguro usando máquinas virtuales[7] con la misma configuración que el sistema de interés.

Dependiendo del tamaño del sistema, es posible usar uno o más equipos anfitriones (Hosts) que alberguen nuestras máquinas virtuales y usar la red física o virtual para interconectar las aplicaciones que se montarán en las máquinas virtuales para implementar el sistema de pruebas de penetración.

Por ejemplo, si todo nuestro sistema desarrollado cabe en un solo equipo de cómputo (no importa que sistema operativo tenga el anfitrión que usemos para las pruebas), entonces podríamos usar una primer máquina virtual[8] para instalar el ambiente de pruebas como por ejemplo Kali Linux[9], dentro de esta instalación instalamos otro manejador de máquinas virtuales para soportar el sistema operativo y la paquetería necesaria[10] que soporte el sistema a probar, dentro de esta máquina virtual además instalaremos el proyecto abierto para la seguridad de aplicaciones OWASP-BWA[11] (Web Open Web Application Security Project), que es un proyecto de código abierto para profesionales de la seguridad, el cual permite crear un ambiente de prueba seguro donde podemos identificar vulnerabilidades en las aplicaciones desarrolladas.

De esta forma podemos crear y usar un ambiente seguro para evitar los controles y detectar vulnerabilidades en los sistemas desarrollados e informar de los hallazgos de las pruebas de penetración antes de que estos se conviertan en brechas de seguridad y se generen daños por ellas.

[7] Entendamos por una máquina virtual a un programa de cómputo que simula a una computadora, en la cual se puede instalar y usar otros sistemas operativos de forma simultánea como si fuese una computadora real sobre nuestro sistema operativo huésped.

[8] Existen una gran variedad de manejadores de máquinas virtuales como: QEMU/KVM, VirtualBox, Vmware, Parallels, Windows Virtual PC, Gnome Box, Xen, LXC, Docker, etc.

[9] Kali Linux (https://www.kali.org), es una distribución para pruebas de penetración estándar de la industria. Es una de las distribuciones más populares entre Pentesters, Hackers éticos e investigadores de seguridad en todo el mundo y contiene cientos de herramientas para el trabajo forense, esta distribución está basada en Debian.

[10] Puede ser por ejemplo Linux, Apache, MySQL, PHP, Perl, Python, etc.

[11] Es un proyecto comunitario que ayuda a organizaciones e instituciones educativas a diseñar, implementar y administrar aplicaciones seguras. Encontrando amenazas como: Inyección (SQL, OS, XXE, LDAP), irrupción en la autentificación y la sesión, Cross Site Scripting, irrupción en el control de acceso, errores en la configuración de la seguridad, exposición de datos sensibles, protección insuficiente contra ataques, uso de componentes con vulnerabilidades conocidos, API con protección insuficiente, etc.

3.2 Programando desde la Web

Existen diferentes servicios WEB que permiten editar, compilar y ejecutar código de diversos lenguajes y paquetes desde el navegador, esto en aras de que los estudiantes y profesores que cuenten con algún sistema de acceso a red y un navegador puedan programar en los más diversos lenguajes, IDEs y Terminales sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular.

Algunos ejemplos de estos servicios son:

3.2.1 SQLite

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://www.codechef.com/ide

Por ser un servicio en red, no es necesario conocer qué bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las tablas necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos la tabla de Productos e insertamos los valores mediante:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
select * from Productos;
```

ahora, creamos la tabla de ventas e insertamos los valores mediante:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
```

```
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
from Ventas
INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

3.2.2 MySQL y MariaDB

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://www.jdoodle.com/

Por ser un servicio en red, no es necesario conocer qué bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las tablas necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos la tabla de Productos e insertamos los valores mediante:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
select * from Productos;
```

ahora, creamos la tabla de ventas e insertamos los valores mediante:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
```

```
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
from Ventas
INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

3.2.3 PostgreSQL

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://rextester.com/l/postgresql_online_compiler

Por ser un servicio en red, no es necesario conocer qué bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las tablas necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos la tabla de Productos e insertamos los valores mediante:

```
create table Productos(
    id_prod int,
    descripcion text not null,
    precio money,
    precioVenta money,
    primary key (id_prod)
);
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
select * from Productos;
```

ahora, creamos la tabla de ventas e insertamos los valores mediante:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
```

```
);
insert into Ventas
values (1, 2, '12/01/2020 8:01:00.00'), (2, 5, '12/01/2020 10:15:00.00'),
(2, 4, '12/01/2020 13:34:00.00'), (1, 3, '12/01/2020 21:56:00.00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
from Ventas
INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

3.2.4 MongoDB

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://www.jdoodle.com/online-mongodb-terminal/

Por ser un servicio en red, no es necesario conocer que bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las colecciones necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos el document de Productos e insertamos los valores mediante:

```
var miProductos =
[
    {"_id" : 1, "descripcion" : "Cigarros", "precio" : 25.5,
"precioVenta" : 30.0},
    {"_id" : 2, "descripcion" : "Refresco", "precio" : 4.5,
"precioVenta" : 6.0},
];
db.Productos.insert(miProductos);
db.Productos.find()
```

ahora, creamos documento de ventas e insertamos los valores mediante:

```
var miVentas =
[
    {"_id" : 1, "id_prod" : 1, "Cantidad" : 2, "Fecha" :
"2020/12/01 8:01:00"},
    {"_id" : 2, "id_prod" : 2, "Cantidad" : 5, "Fecha" :
"2020/12/01 10:15:00"},
    {"_id" : 3, "id_prod" : 2, "Cantidad" : 4, "Fecha" :
"2020/12/01 13:34:00"},
    {"_id" : 4, "id_prod" : 1, "Cantidad" : 3, "Fecha" :
"2020/12/01 21:56:00"},
];
db.Ventas.insert(miVentas);
db.Ventas.find()
```

de esta forma, ahora podemos usar Find para hacer el equivalente a un JOIN como en los ejemplos anteriores y mostrar el resultado de la solicitud, mediante:

```
db.Ventas.find().forEach(
    function (object) {
        var commonInBoth=db.Productos.findOne({ "_id":
object.id_prod} );
        if (commonInBoth != null) {
            print(object.Fecha, object.Cantidad, common-
InBoth.descripcion, commonInBoth.precioVenta, (commonInBoth.precioVenta
* object.Cantidad));
        }
    }
);
```

4 Intercambio de Información

Hay una gran variedad de tecnologías disponibles para producir y almacenar datos. Como son: hojas de cálculo, bases de datos, Software estadístico más específico y más. Esto genera una enorme diversidad de formatos, a veces esto es por decir lo menos, caótico. Algunos de estos formatos, no siempre se adecuan a los niveles de apertura deseados, aquí ofrecemos algunas pautas y recomendaciones que facilitan la adaptación y/o transformación de estos formatos hacia otros más abiertos y fácilmente reutilizables.

XML es un metalenguaje simple pero estricto, desarrollado por W3C. Desarrolla un papel fundamental en el intercambio de una gran variedad de datos. XML es un formato que permite la interpretación de datos a través de varias aplicaciones. Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos. En realidad, XML es una manera de definir lenguajes para diferentes necesidades.

```
<catalogo>
<libro>
<autor>Raúl González Duque</autor>
<titulo>Python para todos</titulo>
<genero>Computación</genero>
</libro>
</catalogo>
```

JSON es un formato de texto popular para el intercambio de datos, es un acrónimo de JavaScript Object Notation. Por su característica de ser un formato de tipo estructurado es especialmente útil para el intercambio de datos entre máquina (machine - readable format). El formato JSON ha sido definido por la especificación RFC 7159 y, tal como CSV, también es un estándar abierto.

```
{
    libro: {
        autor: Raúl González Duque,
        titulo: Python para todos,
        genero: Computación
    }
}
```

YAML es un lenguaje que fue diseñado para ser legible (y como tal será fácil de editar con cualquier editor de texto estándar). Su noción es a menudo similar a reStructuredText o una sintaxis Wiki, que también intentan ser legibles tanto por seres humanos y computadoras. YAML 1.2 incluye también una noción de taquigrafía que sea compatible con JSON, y como tal, cualquier documento JSON es también válido YAML; Sin embargo esto no se sostiene al revés.

```
---
libro:
    autor: Raúl González Duque
    titulo: Python para todos
    genero: Computación
```

KML - KMZ gramática XML y formato de archivo para la creación de modelos y almacenamiento de funciones geográficas como puntos, líneas, imágenes, polígonos y modelos que se mostrarán principalmente en aplicaciones de mapas. Se utiliza para compartir lugares e información entre aplicaciones. Es el estándar del Open Geospatial Consortium y se puede utilizar a través de Google Earth. Los archivos KML se distribuyen comprimidos como archivos KMZ.

SHP Shapefile es un formato propietario de datos espaciales que es el estándar para el intercambio de información geográfica entre Sistemas de Información Geográfica (GIS). Es un formato vectorial de almacenamiento digital donde se guarda la localización de elementos geográficos y los atributos asociados a ellos, pero sin capacidad para almacenar información topológica. Lo generan varios archivos, mínimo 03 y tiene 03 tipos de extensiones: .shp, .shx y .dbf

RDF un formato recomendado por W3C llamado RDF permite representar datos en una forma que facilita la combinación de información de diferentes fuentes. Los datos RDF pueden ser almacenados en XML y JSON, entre otras serializaciones. RDF promueve el uso de URLs como identificadores, lo que brinda una manera conveniente de interconectar iniciativas de "Datos Abiertos" existentes. RDF todavía no se ha generalizado, pero es una tendencia entre iniciativas de Gobierno Abierto, incluso las iniciativas

de Linked Data de los gobiernos británico y español. El inventor de la Web, Tim Berners-Lee, propuso recientemente un esquema de puntuación de cinco estrellas que incluye Linked Data en formato RDF como una meta a alzanzar en las iniciativas de Datos Abiertos.

REBOL es un lenguaje que fue diseñado para ser legible y fácil de editar con cualquier editor de texto estándar. Para lograr que utiliza una sintaxis simple de forma libre con puntuación mínima y un rico conjunto de tipos de datos. REBOL datatypes como direcciones URL, correos electrónicos, valores de fecha y hora, tuplas, cuerdas, etiquetas, etc. respetar las normas comunes. REBOL está diseñado para no necesitar ningún adicional metalenguaje, siendo diseñado de una manera metacircular. El metacircularity de la lengua es la razón por qué por ejemplo, utilizan el dialecto Parse (no exclusivamente) para las definiciones y transformaciones de dialectos REBOL constituye también un dialecto de REBOL. REBOL fue utilizado como una fuente de inspiración del diseñador de JSON.

Gellish inglés es un subconjunto del inglés natural, que incluye una gramática simple y una gran formalizado extensible Diccionario Inglés-taxonomía Eso define la terminología específica general y dominio (términos de conceptos), mientras que los conceptos se organizan en una jerarquía de subtipo-supertipo (una taxonomía), que apoya la herencia de los conocimientos y requisitos. El diccionario-taxonomía también incluye tipos estandarizados hechos (también llamados tipos de relación). Los términos y tipos de relación juntos pueden utilizarse para crear e interpretar las expresiones de hechos, conocimientos, requisitos y otras informaciones. Gellish puede ser utilizado en combinación con SQL, RDF/XML, BÚHO y varias otras lenguas meta. El Gellish estándar está siendo adoptada como ISO 15926-11.

PDF documento de formato portable universal que mantiene la apariencia del documento independientemente del sistema operativo que se utilice (multiplataforma). Incluye cualquier combinación de texto, multimedia e hipertexto y además se puede cifrar el contenido y firmarlo digitalmente. Es el estándar ISO, desde 2008, para ficheros contenedores de documentos electrónicos de cara a la conservación de larga duración. Es una especificación que se puede crear, visualizar o modificar con herramientas de Software libre. Este formato originalmente era propietario (hasta 2008).

Hojas de Cálculo muchas autoridades tienen información que queda en la hoja de cálculo, por ejemplo Microsoft Excel. Estos datos a menudo pueden ser utilizados inmediatamente con las descripciones correctas de lo que las distintas columnas significan. Sin embargo, en algunos casos puede ser macros y fórmulas en hojas de cálculo, las cuales pueden ser más incómodas para manipular. Por tanto, es recomendable documentar este tipo de cálculos junto a la hoja de cálculo, ya que generalmente es más accesible para los usuarios al leer.

Archivos separado por comas los archivos CSV pueden ser un formato útil debido a que son compactos y por lo tanto adecuado para transferir grandes conjuntos de datos con la misma estructura. Sin embargo, el formato es tan espartano que los datos son frecuentemente inservibles sin documentación, ya que puede ser casi imposible adivinar el significado de las diferentes columnas. Por tanto, es particularmente importante para los formatos separados por comas que la documentación de los campos individuales sea precisa.

Además, es esencial que la estructura del archivo sea respetada, como una omisión única de un campo puede perturbar la lectura de todos los datos restantes en el archivo sin ninguna posibilidad real de rectificarla, porque no se puede determinar cómo los datos restantes deben ser interpretados.

Documento de Texto documentos clásicos en formatos como Word, ODF, OOXML, o PDF, puede ser suficiente para mostrar ciertos tipos de datos, por ejemplo, listas de correo o equivalente. Puede que sea barato para exponerlos, siempre que ese sea el formato en que la data se haya creado. El formato no da soporte para mantener la estructura coherente, lo cual significa que a veces es difícil para introducir datos por medios automatizados. Asegúrese de utilizar las plantillas como base de documentos en las que se mostrarán los datos para su reutilización, por lo que es al menos posible extraer información de los documentos. También se puede apoyar el uso posterior de los datos que utilice el marcador de la tipografía tanto como sea posible para que sea más fácil para una máquina para distinguir las partidas (de cualquier tipo) a partir del contenido y así sucesivamente. En general se recomienda no exhibir en formato de procesamiento de texto, si hay datos en un formato diferente.

Texto los documentos de texto (.Txt) son muy fáciles de leer en las computadoras. Por lo general, no incluyen metadatos estructurales, lo que significa que los desarrolladores necesitan para crear un programa de análisis que pueda interpretar cada documento tal y como aparece. Algunos problemas pueden ser causados al cambiar archivos de texto plano entre sistemas operativos. MS Windows, Mac OSX y otras variantes de Unix tienen su propia forma de decirle a la computadora que se ha llegado al final de la línea.

Imagen escaneada probablemente la forma menos adecuada para la mayoría de los datos, pero ambos TIFF y JPEG 2000-al menos puede marcarlos con la documentación de lo que está en la imagen - adecuada para marcar la imagen de un documento con contenido de texto completo del documento. Puede ser relevante a los datos que muestran como imágenes aquellos datos que no nacen por la vía electrónica -un ejemplo obvio es la antigua escuela de registros y el material de archivo- y una imagen es mejor que nada.

Formatos propietarios algunos sistemas tienen sus propios formatos de datos que pueden guardar o exportar datos. A veces puede ser suficiente para exponer los datos en un formato - sobre todo si se espera que el uso de más estaría en un sistema similar del que vienen. Sobre dónde obtener más información sobre estos formatos, debe estar siempre indicado, por ejemplo, al proporcionar un enlace a la página web del proveedor. En general se recomienda mostrar los datos en formatos libres, cuando sea factible.

HTML hoy en día, muchos datos se encuentran disponibles en formato HTML en varios sitios. Esto sería suficiente, si se trata de datos estables y de alcance limitado. En algunos casos, sería preferible tener los datos de una forma fácil para descargar y manipular, pero como es barato y fácil de hacer referencia a una página en un sitio web, podría ser un buen punto de partida en la pantalla de datos.

Generalmente, sería más apropiado el uso de tablas en documentos HTML para almacenar datos y, es importante que los distintos campos de datos se muestren y sean dadas identificaciones, que hacen más fácil encontrar y manipular los datos. Yahoo ha desarrollado una herramienta

$$(https://developer.yahoo.com/yql/)$$

que puede extraer información estructurada de un sitio web, y ciertas herramientas que pueden hacer mucho más con los datos siempre que se encuentren cuidadosamente etiquetados.

HDF (https://www.hdfgroup.org/solutions/hdf5/)

En el ámbito científico, se genera gran cantidad de información en formato digital, que es necesario distribuir. Dada la diferente naturaleza de ésta, es necesario utilizar distintos formatos especialmente adaptados a cada uno de los tipos de datos que se vayan a almacenar. El principal problema radica en la estandarización y la interoperabilidad, ya que, pese a que existen numerosos estándares de almacenamiento de información, no todos son soportados por las distintas aplicaciones informáticas utilizadas, lo que puede presentar problemas en el momento de la difusión y reutilización de dicha información. El formato HDF (Hierarchical Data Format), representa una alternativa eficaz al ser adoptado como estructura de almacenamiento de datos, cuyas características más representativas son las siguientes:

- Permite obtener información acerca de los datos de un archivo desde dentro de ese archivo, sin necesidad de recurrir a fuentes externas.
- Permite almacenar datos de distinta naturaleza en un mismo archivo y relacionarlos entre ellos.
- Estandariza los formatos y las descripciones de los tipos de datos más comúnmente empleados.
- Se trata de un formato abierto, con sus especificaciones publicadas, lo que permite su implementación en diversas aplicaciones informáticas, facilitando la portabilidad, así como permitiendo al usuario desarrollar sus propias aplicaciones específicas.
- Es flexible y puede ser adaptado para almacenar cualquier tipo de dato.

Con el fin de dar respuesta a las necesidades de almacenamiento y difusión de datos científicos de diversa naturaleza, en un formato flexible e independiente de la plataforma, comenzó a desarrollarse el formato HDF en los laboratorios de la NCSA (National Center for Supercomputing Applications) a partir del año 1988, siendo soportado en la actualidad por el HDF Group, dependiente de la Universidad de Illinois. Actualmente, este formato

se utiliza por múltiples organismos, tanto públicos como privados, para la difusión de sus datos y resultados, entre las que se encuentran la NASA, la Agencia Europea del Espacio (ESA), que los aplican en datos procedentes de los sensores MODIS, MERIS o ETM+, entre otros.

La estructura de los archivos HDF permite el almacenamiento de diversos tipos de datos, tales como:

- Scientific Data Sets (SD), utilizados para almacenar matrices n-dimensionales de datos enteros (8, 16 y 32 bits, con o sin signo) o reales (32 o 64 bits) en su formato estándar, o por medio de las APIs, crear datos en otros rangos de valores (1 a 32 bits), junto con sus metadatos (dimensiones de la matriz, atributos, etc.)

- Raster Images (RI), que permiten almacenar imágenes de 8 bits (0-255 niveles de gris) o 24 bits (RGB), o bien utilizar nuevamente las librerías de programación para guardar imágenes en otros formatos (16 a 32 bits enteros, 32 a 64 bits coma flotante). Además, se permite almacenar información sobre las dimensiones de la imagen, así como la paleta de color asociada a ella. También es posible utilizar compresión (RLE, JPEG, GZIP y adaptativa Huffman) para reducir el tamaño de los archivos resultantes.

- Text Annotations (TA), para almacenar cualquier tipo de información textual: Etiquetas, descripciones o información de archivo.

- VDatas (VD), para almacenar datos vectoriales sin topología.

- Vgroups (VG), que nos permiten asociar datos relacionados dentro de un archivo.

4.1 Archivos de Formatos Abiertos

Incluso si la información se proporciona en formato electrónico, formato legible por máquina, y en detalle, puede haber problemas relacionados con el formato del archivo en sí. Los formatos en los cuales la información es publicada - en otras palabras, la base digital en la cual la información es almacenada - puede ser "abierto" o "cerrado". Un formato abierto es aquel donde las especificaciones del Software están disponibles para cualquier persona, de forma gratuita, así cualquiera puede usar dichas especificaciones en su propio

Software sin ninguna limitación en su reutilización que fuere impuesta por derechos de propiedad intelectual.

Si el formato del archivo es "cerrado", esto puede ser debido a que el formato es propietario y sus especificaciones no están disponibles públicamente, o porque el formato es propietario y aunque las especificaciones se han hecho públicas, su reutilización es limitada. Si la información es liberada en un formato de archivo cerrados, esto puede causar grandes obstáculos para reutilizar la información codificada en él, forzando a aquellos que deseen usar la información a comprar Software innecesario.

La ventaja de los archivos de formatos abiertos, es que permiten a los desarrolladores producir varios paquetes de Software y servicios utilizando esos formatos. Esto entonces reduce al mínimo los obstáculos para la reutilización de la información que contienen.

El uso de formatos de archivo con propiedad, para el que la especificación no está disponible públicamente, puede crear dependencia de Software de terceros o de los titulares de licencias de los formatos de archivos. En el peor de los casos, esto puede significar que la información sólo se puede leer con ciertos Softwares especiales, que pueden ser muy caros, o que pueden quedar obsoletos.

La preferencia del término Gobierno de Datos Abiertos, es que la información se publicará en formatos de archivo abiertos, los cuales son de lectura mecánica.

¿Cómo usar un formato determinado? cuando una autoridad debe mostrar nuevos datos -datos que no han sido expuestos antes- se debe elegir el formato que proporciona el mejor equilibrio entre el costo y la idoneidad para el propósito. Para cada formato, hay algunas cosas que usted debe tener en cuenta, y esta sección tiene como objetivo explicarlas.

Servicios Web cuando los datos cambian frecuentemente, y el tamaño de cada descarga es limitado, es muy importante exponer la información a través de web services. Hay muchas maneras de crear web services, pero alguno de los más usados son SOAP y REST. Generalmente SOAP antes que REST. Los servicios REST son fáciles de desarrollar y entender, por eso es un estándar muy usado.

Base de datos como los servicios webs, las bases de datos permiten el acceso directo a los datos dinámicamente. Las bases de datos tienen la ventaja de extraer aquella información en la cual estén interesados. Hay algunas preocupaciones de seguridad en cuanto a permitir extracciones remotas a bases de datos, y el acceso a las bases de datos es útil si su estructura y la importancia de las tablas y campos individuales están bien documentados. Generalmente es relativamente simple y no tiene costo crear web services que expongan datos de una base de datos, esta puede ser una manera fácil de lidiar con las problemáticas de seguridad.

Intercambio de datos único dominio a menudo hay un par de docenas de diferentes fuentes y esquema de destino (formatos de datos propietarios) en un dominio específico. A menudo las personas desarrollan una formato de intercambio de o formato de intercambio de para algunos un solo dominio y luego escribir unas docena diferentes rutinas para traducir (indirectamente) cada esquema de origen de cada esquema de destino utilizando el formato de intercambio como un paso intermedio. Eso requiere mucho menos trabajo que escribir y depurar los cientos de diferentes rutinas que se necesitaría para traducir directamente cada esquema de origen directamente a cada esquema de destino. (Por ejemplo, Formato de intercambio estándar para la infraestructura de datos espaciales, Formato de intercambio de datos para los datos de la hoja de cálculo, GPS eXchange Format o Keyhole Markup Language para indicar las coordenadas GPS del mundo, Formato de intercambio de Quicken para los datos financieros, GDSII diseño de circuitos integrados, etc.)

Intercambio de datos de idiomas un lenguaje de intercambio de datos es un lenguaje que es independiente del dominio y puede utilizarse para cualquier tipo de datos. Su capacidad de expresión semántica y sus cualidades se determinan en gran medida en comparación con las capacidades de lenguajes naturales. El término también se aplica a cualquier formato de archivo puede ser leído por más de un programa, incluyendo formatos propietarios como Microsoft Office documentos. Sin embargo, un formato de archivo no es un lenguaje real como carece de una gramática y vocabulario.

La práctica ha demostrado que ciertos tipos de lenguajes formales son los más adecuados para esta tarea que otros, puesto que su especificación es impulsado por un proceso formal en lugar de una implementación de Software

en particular las necesidades. Por ejemplo XML es un lenguaje de marcado fue diseñado para permitir la creación de dialectos (la definición de dominio específico sublanguages) y una opción popular ahora en particular en internet. Sin embargo, no contiene diccionarios específicos de dominio o tipos de hecho. Beneficiosa para un intercambio de datos fiable es la disponibilidad de diccionarios-taxonomías estándar y las bibliotecas de herramientas tales como Analizadores, esquema Validadores y herramientas de transformación.

Antes de seguir, introduciremos dos conceptos que se usarán a lo largo de toda la guía:

- Distribución o Recurso: Una distribución o recurso es la unidad mínima en la que se publican datos. Se trata de los archivos que pueden ser descargados y re-utilizados por un usuario. Los recursos pueden tener diversos formatos (.csv, .shp, etc.).

- Dataset: Un conjunto de datos o dataset agrupa recursos referidos a un mismo tema que respetan una estructura de la información. Los mismos podrían diferir en el formato en que se los presenta (Ej. .csv, .json, .xls, etc.), la fecha a la que se refieren, el área geográfica cubierta o estar separados bajo algún otro criterio.

Un recurso en formato tabular es un archivo plano que se ajusta a un esquema predefinido de columnas, incluyendo el nombre de la columna y el tipo de datos. En la mayoría de los casos, corresponde a datos que llegan de bases de datos, reportes y planillas de cálculo en general. A diferencia de los formatos tabulares, los archivos JSON siguen una estructura diferente donde se definen listas de objetos con pares "clave" - "valor".

Recomendamos con énfasis la publicación de los datos en formato CSV y/o JSON. En caso de utilizar formatos propietarios o aún no estandarizados, es útil indicar Software, versión y aplicación que permite procesar esos formatos.

CSV es un formato de archivo de texto plano donde las campos (columnas) se separan por comas; y los registros (filas), por saltos de línea. Algunas versiones alternativas de esta forma de publicar datos usan otros separadores como punto y coma (";") o pipe ("|"). Las recomendaciones, siguiendo lo indicado por la especificación RFC4180 y las pautas de la W3C, para la publicación de archivos en formato CSV son:

- Las filas deben finalizar con los caracteres de "retorno de carro" (\r) y "salto de línea" (\n) unidos (\r\n). Esta forma de separar las líneas se denomina usualmente CRLF ("Carriage Return Line Feed").
- La primera fila siempre contiene los nombres de los campos.
- No se deben repetir nombres entre los campos.
- No se debe colocar espacios al principio ni al final del nombre de un campo, o de un valor.
- Tanto los campos como los valores deben estar separados por comas (",").
- En el caso de que un valor contenga el carácter separador (",") o cualquiera de los caracteres que separan las líneas ("\r", "\n" o "\r\n"), el valor completo debe ser encerrado entre comillas dobles "". Esto indica que el carácter no cumple el rol de separar columnas o filas, sino que es parte de un valor. Ejemplo:

  col1,col2\r\n
  "La tasa de Juan, está vacía",La tasa de Pablo está llena\r\n
  "La tasa de Juan\nestá vacía",La tasa de Pablo está llena\r\n
  "La tasa de Juan\r\nestá vacía",La tasa de Pablo está llena\r\n

- En el caso de que un valor contenga el carácter comilla doble ("), el valor debe ser encerrado entre comillas dobles como en el caso anterior ("") y, además, los caracteres comilla doble que se encuentren dentro del valor deben escribirse dos veces (""). Ejemplo:

  col1,col2\r\n
  "La tasa de ""Juan"" está vacía",La tasa de Pablo está llena\r\n

- Para todos los tipos de datos se considera válido el valor indefinido. Este se expresará con la ausencia de todo carácter y no con un carácter o string especial como podrían ser ".", "null", "none", "nan", etc. Ejemplo:

  col1,col2,col3\r\n
  a„b\r\n
  a,"",b\r\n

Fragmentación de archivos Para garantizar la accesibilidad a los datos, es necesario fragmentar los archivos excesivamente grandes, que superen el millón de filas.

Para esto, recomendamos usar conceptos simples, fragmentando:

- Por períodos en caso de tratarse de información temporal (Ej. Años, semestres, trimestres, meses, semanas, días),
- Por zonas en caso de tratarse de información geográfica (Ej. provincias, municipios, barrios, secciones, o manzanas) o
- Por dimensiones temáticas propias del dominio particular de la información.

Sin embargo, siempre que se decida fragmentar un archivo para garantizar su accesibilidad, recomendamos publicar también una versión no fragmentada que contenga el conjunto de datos completo (aunque sea muy grande), a los fines de evitar la tarea de consolidación.

Para eso, sugerimos usar protocolos de compresión, en especial para archivos muy grandes, y altamente compresibles. De hacerlo, aconsejamos protocolos sin pérdida, y abiertos.

Nomenclatura de archivos Recomendamos estas convenciones para nombrar archivos:

- Usar palabras siempre en minúsculas.
- No usar artículos.
- Usar únicamente letras y números ASCII, siempre en minúsculas, comprendidos en el rango "a-z" y "0-9".
- Separar las palabras con guión medio "-".
- En caso de corresponder, ubicar la referencia temporal o del atributo de fragmentación siempre al final.

Ejemplos:

acceso-informacion-publica.csv: Versión completa del recurso.

acceso-informacion-publica-2013.csv: Versión del recurso fragmentada por año.

acceso-informacion-publica-201302.csv: Versión del recurso fragmentada por mes.

acceso-informacion-publica-caba.csv: Versión del recurso fragmentada por división político-territorial (provincia o caba).

acceso-informacion-publica-caba-2013.csv: Versión del recurso fragmentada por división político-territorial (provincia o caba) y año.

acceso-informacion-publica-jujuy-20130208.csv: Versión del recurso fragmentada por división político-territorial (provincia o caba) y fecha.

Para la fragmentación temporal, recomendamos el estándar de los ejemplos, ya que es compacto y ordena los recursos por tiempo: YYYYMMDD. Por favor, recordemos mantener siempre dos dígitos para el mes y el día, incluso si el número es menor a 10.

Codificación Todos los recursos de datos, incluyendo los geográficos, deben publicarse usando la codificación UTF-8 siguiendo las recomendaciones de la W3C en lugar de ASCII

https://cio-wiki.org/wiki/American_Standard_Code_for_Information_Interchange_(ASCII)

y

$$https://en.wikipedia.org/wiki/ASCII.$$

Una de las principales razones es que UTF-8

$$https://es.wikipedia.org/wiki/UTF-8$$

soporta una gran variedad de lenguajes, según la W3C es un "estándar en el que se definen todos los caracteres necesarios para la escritura de la mayoría de los idiomas hablados en la actualidad. Su objetivo es ser, y, en gran medida, ya lo ha logrado, un superconjunto de todos los sets de caracteres que se hayan codificado".

Estructura y características de los datos tabulares En esta sección veremos:

- Recomendaciones generales para el trabajo con datos, y
- Recomendaciones para el trabajo con planillas de cálculo, orientadas tanto a facilitar su exportación a formatos abiertos, como a su propia usabilidad en el contexto de cualquier aplicación de planillas de cálculo.

Recomendaciones generales Estas son recomendaciones generales para el trabajo con datos tabulares. Sugerimos adoptarlas sea cual sea la tecnología usada.

Muchas de las recomendaciones aquí presentadas se encuadran en los principios de Tidy Data delineados por Hadley Wickham. Éstos establecen, por ejemplo, que en una tabla de datos "cada variable es una columna, cada observación es una fila, y cada tipo de unidad observacional es una tabla". Sugerimos complementar la lectura de esta guía con la del trabajo que mencionamos.

Nomenclatura de los campos (nombres de las columnas) La "nomenclatura de los campos" es el nombre de las columnas en los datos de estructura tabular. Estas recomendaciones aplican a la generalidad de los casos, pero cuando haya convenciones particulares según la temática o rubro de datos de que se trate y éstas entren en conflicto, puede ser conveniente privilegiar primero la convención de la temática específica y luego la convención general.

Los nombres de los campos deben:

- Estar en español.
- Ser lo más explícitos, descriptivos y declarativos como sea posible.
  - Es preferible que el nombre de un campo sea claro antes que corto, pero se recomienda no superar los 50 caracteres.
  - No usar abreviaturas si no es estrictamente necesario o recomendado por una convención ampliamente difundida. En caso de usarlas, incluirlas en el diccionario de datos.
- Estar en minúsculas, no incluir caracteres especiales, ni estar subrayados.

- Usar palabras compuestas únicamente de caracteres en minúsculas comprendidos en el rango a-z (letras sin tildes) y en el rango 0-9 (dígitos).
- Las palabras deben unirse con guión bajo " _ ".
- No contener espacios.
- Las palabras deben separarse siempre con " _ ", en lugar de no tener separación alguna: fecha_audiencia_solicitada en lugar de fechaaudienciasolicitada

- Referirse a un sólo atributo de los datos, indivisible en más de un campo.

  - Los campos deben separar los atributos de los datos en la forma más desagregada que sea posible.
  - Se debe evitar definir campos que contengan más de un tipo de información (por ejemplo: e-mail y sitio web, número de teléfono, etc bajo "datos_de_contacto").

- Si existe una entidad que engloba varias características separadas en campos diferentes, comenzar nombrando los campos con esa entidad y luego con los atributos más específicos (de lo más general a lo más específico).

  - Ej.: solicitante y solicitante_documento son entidades más generales que se repiten en varios campos, que corresponden a atributos más específicos.

    solicitante_nombre
    solicitante_cargo
    solicitante_documento_tipo
    solicitante_documento_numero

- Resulta más fácil identificar qué campos están relacionados entre sí porque configuran atributos de una misma entidad, en lugar de parecer campos conceptualmente independientes. Además, el ordenamiento alfabético de los campos los dejaría automáticamente agrupados por su pertenencia a una entidad más importante.

- Incluso cuando la entidad de un atributo parezca evidente (ej.: un dataset llamado "audiencias" donde todos los campos son atributos de la entidad "audiencia"), se recomienda nombrar el campo incluyendo la entidad a la que hace referencia el atributo.

  No recomendado: "fecha_hora"
  Recomendado: "audiencia_id", "audiencia_fecha_hora".

Los campos que sean identificadores o códigos, deberán incluir el sufijo "_id" en el nombre del campo, salvo casos excepcionales donde un nombre alternativo sea más conveniente porque ofrece información sobre el sistema de identificación usado.

- En cuanto a los campos que contengan la descripción de ese identificador, se recomienda que incluya el sufijo "_desc" (por "descripción") en caso de que no exista una forma más conveniente de nombrar el campo.

Nivel de granularidad de los datos Por favor, no incluir totales, subtotales ni agrupamientos de datos. Un dataset debe ser consistente en el nivel de granularidad de los datos que contiene. Está bien tener un dataset con la cantidad de convenios firmados por provincia y está bien tener un dataset con la cantidad de convenios firmados por municipio. No está bien tener un dataset que mezcle ambos.

Dicho esto, el dato agregado "convenios firmados por provincia" siempre se puede calcular a partir de un proceso del dataset más desagregado, pero esto no es así a la inversa (es imposible recuperar los datos a nivel de municipio desde el dataset provincial).

Usar orientación vertical en lugar de horizontal Es preferible que la orientación de los datos sea "vertical" en lugar de "horizontal" en los casos en que esto sea posible. La principal razón es que los datos orientados de manera vertical facilitan el tratamiento y análisis de los datos.

Incluir sólo un atributo por campo Se recomienda definir los campos de forma atómica de modo de incluir un sólo atributo por elemento, en lugar de datos múltiples, generando campos adicionales de ser necesario.

Valores nulos, desconocidos o en blanco en campos numéricos Los valores de los datos deben ser siempre explícitos y respetando el tipo de datos del campo de que se trate. Los elementos o celdas en blanco se interpretarán siempre como "valor ausente". Si existen distintas interpretaciones posibles de un "valor ausente", éstas deben ser explicitadas en un campo aparte. Si sólo hay "valores ausentes" (no hay distintas interpretaciones, es siempre un "valor ausente") no es necesario agregar una columna adicional.

Es importante destacar, por ejemplo, que cuando un valor numérico sea "0" siempre debe ponerse un "0" como dato (y no un valor nulo, en blanco o vacío).

Recomendaciones para estructurar planillas de cálculo Las recomendaciones de esta sección aplican exclusivamente al trabajo sobre planillas de cálculo.

Usar celdas simples recomendamos usar celdas simples y, en ningún caso, combinar celdas.

Fila de encabezado Los datos deben contener sólo una fila de encabezado. Desde la segunda fila en adelante, sólo debe haber datos, pero nunca un encabezado.

Celdas vacías en filas para agrupar conceptos Recomendamos no dejar celdas vacías en filas bajo la presunción de que valores en blanco posteriores a un valor positivo contienen implícitamente a ese mismo valor en una suerte de "agrupamiento conceptual". Este error es muy común en la construcción de planillas de cálculo y suele generar problemas graves cuando cambia el orden original de las filas. Además, impide el uso de tablas dinámicas y otras formas de analizar los datos.

Formato de celdas Las celdas de una planilla de cálculo deben estar formateadas acorde al tipo de datos de que se trate. Específicamente, los números siempre deben estar en celdas de formato/tipo "número", los campos de tipo textual deben estar en celdas de formato/tipo "texto" y los campos de tipo fecha deben estar en celdas de formato/tipo "fecha".

Mantener el formato correcto de las celdas según el tipo de datos que contengan:

- Mejora las probabilidades de que una exportación a otro formato salga correctamente.
- Hace que los datos sean más operables en la propia planilla de cálculo, aprovechando mejor sus funcionalidades.

Exportación a CSV Insistimos: CSV es el formato más recomendado para la publicación de archivos tabulares. A la hora de exportar una planilla de cálculo a CSV hay 3 parámetros que deben ser especificados durante la exportación, independientemente del Software de que se use:

- Codificación (encoding, en inglés): siempre debe ser UTF-8.
- Carácter separador (separator character, en inglés): siempre debe ser "," (coma).
- Carácter calificador (quote character o enclosing character, en inglés): siempre debe ser " " " (comillas dobles).

4.2 Estándares Según el Tipo de Datos

El formato recomendado para los distintos tipos de datos está mayormente basado en las especificaciones de la W3C. En los otros casos, las recomendaciones surgen de la experiencia de trabajo del equipo de la Dirección Nacional de Datos e Información Pública y del esfuerzo realizado en la búsqueda de estándares más adecuados.

Texto

- Los campos de texto no deben contener espacios en blanco innecesarios al principio ni al final.

Entidades Las entidades que aparezcan entre los datos de un campo textual deben tener una descripción única. Es decir, toda mención que se realice a una entidad dada debe hacerse usando exactamente la misma cadena de caracteres cada vez:

- Las descripciones de entidades deberían elegirse siempre de forma tal que cumplan con el estándar específico que las describe, en caso de que este exista.

- Cuando este estándar no existe y hay dudas respecto del criterio a adoptar para elegir la descripción única de una entidad, debe privilegiarse siempre aquella que sea lo más explícita, descriptiva y declarativa posible.

Siempre que sea posible, la elección deberá fundamentarse en el estándar establecido para ese tipo de entidad (para más información ver la Guía para la identificación y uso de entidades interoperables). En el caso de no existir un estándar, deberá adecuarse a las pautas generales del contexto del dataset de que se trate.

Nombres propios Se capitalizan (la primera letra de cada palabra es mayúscula, el resto de las letras son minúsculas) todas las palabras significativas, salvo las siglas. Las palabras significativas son aquellas que no cumplen la función de artículos o preposiciones.

Siglas Todas las siglas se escriben en mayúsculas, sin usar puntos ni espacios intermedios.

Número

- El separador decimal debe ser el carácter ".".
- No se usará separador de miles.
- No se deberán usar espacios en blanco.
- Para los números negativos debe incluirse el símbolo menos "-" inmediatamente antes del número, sin espacio en blanco intermedio.

Moneda Los valores numéricos que sean además valores monetarios se consideran números y, por lo tanto, valen las mismas recomendaciones que para ellos. Además, agregamos las siguientes recomendaciones:

- La cantidad de decimales debe limitarse a dos, salvo que el uso de una mayor cantidad de decimales sea significativo para el caso particular.
- En ningún caso se incluirán símbolos o letras en el campo numérico -ya sea "$", "DOL", "USD", etc.

Si en el recurso los valores monetarios están expresados en diferentes monedas, se recomienda indicarlo en un campo aparte (que puede llamarse "moneda_id") usando los códigos alfabéticos definidos en la ISO 4127.

Números telefónicos En este apartado, proponemos una solución para incluir números telefónicos nacionales en los recursos de datos. A nivel internacional, el estándar para los números telefónicos fue desarrollado por el "Sector de Normalización de las Telecomunicaciones de la Unión Internacional de Telecomunicaciones" (ITU Telecommunication Standardization Sector) bajo la recomendación E.164.

Para el caso de los números nacionales, el ENACOM tiene la competencia sobre el sistema de numeración telefónica. Este organismo determina que el número nacional de abonado debe estar compuesto por 10 dígitos. Estos 10 dígitos están conformados por un indicativo interurbano más un número de abonado. Pudiendo el indicativo interurbano tener entre 2 y 4 dígitos, y el número de abonado entre 6 y 8 dígitos.

Coordenadas Para registrar datos de coordenadas geográficas de puntos, usamos números decimales. Los campos deberán llamarse "latitud" y "longitud". Cuando sea conveniente especificar el nombre de la entidad de la cual se consignan las coordenadas, se usarán los sufijos "_latitud" y "_longitud".

Tiempo Se usará el estándar ISO 8601

$$(YYYY - MM - DDTHH : MM : SS[.mmmmmm][+HH : MM]).$$

A menos que se indique lo contrario, se asumirá que la zona horaria.

- Fecha: YYYY-MM-DD
- Hora: HH:MM:SS[.mmmmmm][+HH:MM]
- Fecha y Hora: YYYY-MM-DDTHH:MM:SS[.mmmmmm][+HH:MM]
- Duración: YYYY-MM-DDTHH:MM:SS[.mmmmmm]

Rangos horarios

- Los rangos estarán divididos en dos partes separadas por un doble guión bajo "__", la primera indica el día y la segunda, la hora.
- Se puede omitir la parte del día o bien de la hora pero nunca ambas.
- Si se omite la parte que indica el día se asumirá que el rango abarca todo el horario indicado.
- Si se omite la parte que indica el horario se asumirá que el rango abarca todo el día indicado.
- El día se puede indicar tanto mediante rangos separando los días con guiones medios "-" o bien como particulares con el guión bajo "_".
- La hora se indica mediante rangos, separando los horarios con guiones medios ("-"). También se pueden indicar varios horarios con el guión bajo "_".
- En caso de que se necesite cubrir más de una franja horaria y esta sintaxis sea insuficiente, se pueden incluir varias separadas por espacios.
- Los días se indicarán con sus iniciales en castellano: LUN, MAR, MIE, JUE, VIE, SAB y DOM

Booleano A menos que se indique lo contrario, se identificarán con los valores true o false.

- Esta convención puede variar en algunos rubros específicos de datos, pero en caso de no existir una convención clara y definida aplicable al rubro o contexto del dataset, se recomienda utilizar true o false.
- Este campo puede contener "valores ausentes". En ese caso, el campo deberá estar totalmente vacío, no conteniendo ningún caracter.
- Si existe la posibilidad de que haya otro valor que no sea true, false o "valor ausente" significa que se eligió un tipo de datos incorrecto: este no es booleano, el tipo de dato booleano es binario y sólo admite 2 valores de verdad (aparte del caso del "valor ausente").

Datos de entidades interoperables ¿Qué son? Las entidades interoperables son aquellas que se repiten y usan frecuentemente dentro de datasets de:

- Temáticas diversas entre sí.
- Una misma temática (ej.: Salud), pero no de otras (como Educación, Economía, Transporte, etc).

La mayoría de los datasets incluyen campos que responden al dónde, quién, cuándo y qué. Estos campos permiten que los datasets sean interoperables entre sí.

4.3 ¿Qué son los Metadatos?

Los metadatos son los datos sobre los datos. Es decir: elementos descriptivos que dan contexto a un conjunto de datos, y acercan al usuario la información necesaria para entenderlos y usarlos eficazmente.

Un título y una breve descripción son los metadatos básicos que cualquier conjunto de datos a publicar debería tener. Después, existen muchos otros elementos que ayudan al lector a hacer un buen uso de los datos. Por ejemplo:

- Nombre, tipo de datos y descripción de los campos: ¿qué significa cada campo? ¿qué datos puedo encontrar en esa columna? ¿qué dicen y qué no dicen esos datos, cómo debo leerlos?
- Palabras clave: clasifican a un dataset como perteneciente a un conjunto de tópicos.
- Tema: clasifican a un dataset como perteneciente a un determinado tema, dentro de una jerarquía temática.
- Fecha de publicación: ¿cuándo se publicó por primera vez este dataset?
- Fecha de última modificación: ¿cuándo se actualizó por última vez este dataset?
- Frecuencia de actualización: ¿cada cuánto se actualiza este dataset?
- URL de descarga: ¿cómo dispongo de los datos, desde dónde puedo descargarlos?

Una lista curada de campos de metadatos, junto con las instrucciones de cómo deben utilizarse, define un perfil de metadatos.

¿Cómo se publican los metadatos? la publicación de los metadatos puede ser muy diversa en detalle, calidad y forma. Una publicación muy elemental es un documento de texto que ofrece una descripción del dataset y de cada uno de los recursos que lo componen.

Sin embargo, las computadoras no pueden leer fácilmente documentos de texto. La organización sistemática de colecciones de datasets (es decir, la creación de un catálogo de datos) exige un nivel de complejidad mayor para facilitar su descubrimiento, indexación, y reutilización por parte de scripts y aplicaciones de todo tipo.

4.4 Qué son los NFTs ¿Burbuja, Moda o Nuevo Paradigma?

A fines de los noventa y principios del nuevo siglo, muchos académicos consideraron que la Economía como ciencia estaba experimentando una experiencia similar a la de la Teología con Galileo o Darwin. Habiéndose basado en la idea de la escasez de un elemento primordial (oro, petróleo, silicio) ahora debería reinventarse para una era basada en un elemento superabundante; la información.

Con la aparición de la impresión 3D permitiendo la producción a pequeña escala de productos hasta entonces reservados a grandes plantas industriales y, la de las criptomonedas que desafiaba el poder de los Estados en la producción de divisas, esa opinión pareció confirmarse.

¿Qué son los NFTs? Los NFTs nacieron de la mano de la tecnología de la cadena de bloques[12] para garantizar que un artículo se mantenga único en un ambiente donde todo puede copiarse. Las letras representan las siglas en inglés de *Tokens No Fungibles*. Esto significa que no se pueden reemplazar ni intercambiar porque tienen propiedades únicas. De la misma forma, no se pueden falsificar ni manipular porque su autenticidad está respaldada por un certificado creado utilizando la tecnología de cadena de bloques.

[12]Es una colección de información almacenada en forma electrónica a la que se puede acceder, filtrar y manipular rápida y fácilmente cualquier número de usuarios a la vez.

Los NFT, como dijimos, son Tokens individuales con información valiosa almacenada en ellos. Esta información contienen los datos únicos de los NFTs son los que permiten la verificación y validación de su propiedad y la transferencia de Tokens entre propietarios. La creación y circulación de NFTs falsos no funciona porque cada artículo puede rastrearse hasta el creador o emisor original.

Dependiendo del mercado en que se comercialice un NFT, el creador original puede seguir obteniendo regalías de las ventas sucesivas.

Diferencias entre NFTs y criptomonedas Aunque la tecnología detrás de las criptomonedas y los NFTs es la mísma, hay una diferencia fundamental.

Las criptomonedas son fungibles, es decir que pueden intercambiarse sin que se note la diferencia. La firma digital que acompaña a cada Token No Fungile hace esto imposible.

Productos que pueden comercializarse como NFTs

- Arte digital.
- Gif animados.
- Vídeos de acontecimientos destacables.
- Avatares virtuales y otros complementos para videojuegos.
- Música.

¿Moda, burbuja o nuevo paradigma? En principio, los NFTs parecen una idea genial. los creadores pueden vender su contenido directamente al consumidor como un NFT, lo que también les permite quedarse con un mayor porcentaje de ganancia que la que dejan la mayoría de las plataforma. Como dijimos, se pueden programar regalías para recibir un porcentaje de las ventas cada vez que se produzca un cambio de manos.

Los precios también son bastante atractivos. Nyan Cat, un GIF de un gato con un cuerpo de tartaleta que apareció en el 2011, se vendió por casi $ 600,000 a principio de año. Por otra parte, videos con los momentos más destacados de la NBA generaron más de $ 500 millones en ventas a fines de marzo del 2021. Un solo NFT destacado de LeBron James se vendió por más de 200.000 dólares. Pero, los expertos aconsejan cautela. ya que se trata de una modalidad relativamente nueva.

Tokens En principio, la palabra Token puede usarse como sinónimo de «criptomoneda» o criptodivisa. Aunque, lo más correcto, teniendo en cuenta el tema, deberíamos hablar de «criptoactivos». Siendo más específicos se utiliza para describir ciertos activos digitales que se ejecutan sobre la cadena de bloques de otras criptomonedas, como lo hacen muchos Tokens de finanzas descentralizadas (o DeFi) o los propios NFTs. Gracias a los Tokens se hacen posible los intercambios descentralizados, por ejemplo la venta de complementos para videojuegos. Siempre manteniendo la posibilidad de pueden negociarse o atesorarse como cualquier criptodivisa.

Algunos tipos de Tokens

- Tokens DeFi: Existen protocolos basados en criptomonedas que tienen como objetivo reproducir las funciones tradicionales del sistema financiero; préstamos, ahorros, seguros, comercio internacional. Dichos protocolos emiten Tokens que realizan una amplia variedad de funciones, pero también se pueden comerciar o mantener como cualquier otra criptomoneda.

- Tokens de gobernanza: Es el equivalente digital a las acciones de una sociedad. Estos son Tokens DeFi especializados que les dan a los propietarios derecho a voto en cuestiones sobre el protocolo.

- Tokens no fungibles: Estos Tokens representan derechos de propiedad sobre un activo digital o del mundo real único. Se utilizan para restringir la copia y el intercambio de creaciones digitales. También sirven para emitir un número limitado de obras de arte digitales o vender activos virtuales únicos.

- Tokens de propiedad (Security Tokens): Son el equivalente criptográfico a las acciones y bonos. Se utilizan para vender participaciones en empresas o proyectos sin la necesidad de utilizar un agente de bolsa o cumplir las normas de los entes reguladores.

Cadena de Bloques (Blockchain) Para definirlo en forma simple, una cadena de bloques es un tipo de base de datos. Es decir que es una colección de información almacenada en forma electrónica a la que se puede acceder, filtrar y manipular rápida y fácilmente cualquier número de usuarios a la vez.

Sin embargo, existe una diferencia importante con las bases de datos tradicionales. La forma en que se estructuran los datos. En una cadena de bloques se recopila información en grupos (los bloques que le dan nombre), estos grupos contienen conjuntos de información. Los bloques tienen ciertas capacidades de almacenamiento y, cuando se llenan, se encadenan al bloque previamente llenado, formando una cadena de datos conocida como «Blockchain». Toda la información nueva que sigue a ese bloque recién agregado se ingresa en un bloque recién formado que luego también se agregará a la cadena una vez que se complete.

Este sistema también crea una línea de tiempo irreversible de datos cuando se implementa de manera descentralizada. Cuando se llena un bloque, se hace inmodificable y se convierte en parte de esta línea de tiempo. A cada bloque de la cadena se le asigna una marca de tiempo exacta cuando se agrega a la cadena.

En una cadena de bloques, cada nodo tiene un registro completo de los datos que se han almacenado en la cadena de bloques desde su inicio. Si un nodo tiene un error en sus datos, puede usar los miles de otros nodos como punto de referencia para corregirse. De esta manera, ningún nodo dentro de la red puede alterar la información contenida en ella.

Contratos Inteligentes Se trata de un programa que puede integrarse en la cadena de bloques para facilitar, verificar o negociar un acuerdo contractual. Los contratos inteligentes trabajan de acuerdo a un conjunto de reglas que los usuarios aceptan. Cuando se cumplen esas condiciones, los términos del acuerdo se llevan a cabo automáticamente.

Billetera Criptográfica (Crypto Wallet) Se trata de un Software que interactúa con la cadena de bloques permitiendo generar claves públicas y privadas, monitorear los saldos y enviar y recibir criptomonedas.

Los Tokens No Fungibles Debido al estrecho parentesco de los NFTs con las criptomonedas, el término correcto para referirse a su creación, es acuñar. El primer paso para la creación y venta es elegir una plataforma de cadena de bloques. La preferida por la mayoría de los creadores parece ser *Ethereum*.

La idea de los creadores del proyecto fue crear un ordenador mundial descentralizado imparable, resistente a la censura y autosuficiente. En él

corre un Software que garantiza que los datos y los pequeños programas informáticos llamados contratos inteligentes se replican y procesan en todos los ordenadores de la red, sin un coordinador central.

Habíamos dicho que los contratos inteligentes son programas que garantizan que cuando una de las partes cumple una condición la otra cumpla con lo comprometido. Es decir que a diferencia de las cadenas de bloques tradicionales, Ethereum no solo almacena datos, además realiza operaciones con esos datos y almacena los resultados.

El segundo criterio es contar con una billetera compatible. El uso de la red Ethereum requiere el pago de una tarifa, para abonarla necesitarás tener ETH, su moneda nativa, en la billetera. La puedes adquirir en cualquier sitio de intercambio.

La gran ventaja de usar Ethereum para el intercambio de los NFTs, es que es compatible con los estándares de creación de código abierto. Los NFTs se guardan en tu billetera. Esto hace que no tengas que comprometerte con ninguna plataforma de compraventa.

Plataformas de Creación de NFTs Dado que los NFTs están basados en estándares abiertos, es posible crearlos sin utilizar una plataforma. Sin embargo, iniciemos por el camino más fácil:

- OpenSea Cuenta con una función que permite crear NFT sin pagar tarifas en ETH. La plataforma publicó una guía paso a paso sobre cuáles son los pasos a seguir. Primero hay que crear una colección en la cual se almacenarán los NFTs. A partir de esto, se pueden crear utilizando archivos de imagen, archivos de video, modelos 3D, archivos de música o cualquier otro tipo de archivo de contenido digital. U paso adicional pero optativo es incluir un nombre, una descripción y establecer la rareza.

- Mintbase es otra plataforma ideal para principiantes. Utiliza un procedimiento muy parecido a OpenSea salvo que en lugar de crear una colección, lo que se crea es una tienda. Un punto en contra es que Mintbase solo admite imágenes por el momento, por lo que solo sirve para artistas visuales. Después de crear la tienda, llega el momento de acuñar los NFTs con su nombre, descripción y cantidad. La opción predeterminada es que una vez creado, un NFT sale a la venta, pero es posible retirarlos de manera momentánea.

- Foundation es para artistas con pretensiones de exclusividad. Cualquiera puede inscribirse, pero solo aquellos que reciben una invitación pueden crear sus NFTs. Si quieres saber cuales son los requisitos, no tienes más que visitar su página. Foundation permite la acuñación de NFT con imágenes, archivos de video, archivos de audio y modelos 3D. Se puede elegir el nombre, la descripción y la cantidad de ejemplares de cada uno.

Costo Energético El consumo ligado a los NFTs es preocupante... Es difícil estimar la huella de carbono ligada al proceso de 'acuñado' (Minting) y comercialización de un NFT porque hay muchos procesos implicados. De hecho lo que se ha estudiado sobre todo no es el impacto medioambiental de la creación de un NFT en particular, sino de una transacción en la red Ethereum en general.

Hay varios estudios que intentan estimar esa cantidad. Digiconomist sitúa la huella de carbono de una transacción con Ethereum en 46.54 kg de CO2, mientras que el artista y programador Memo Akten afirma que esa huella es era de 20 kg de CO2.

Atken iba más allá y explicaba cómo ese clic de ratón con el que compramos un NFT implica que se ponga en marcha esa compleja maquinaria del minado de la criptomoneda ETH en la red Ethereum. Ese minado, que actualmente se realiza porque Ethereum usa un algoritmo Proof-of-Work (PoW), hace que el consumo energético sea elevado, pero en NFT implica además otros consumos que llevan la huella de carbono hasta los 48 kg de CO2 según sus datos.

De hecho si tenemos en cuenta el resto de procesos implicados en su comercialización (acuñado, subastas, cancelación de subastas, venta, transferencia de propiedad), Atken afirma que la huella total va mucho más allá y roza los 200 kg de CO2.

Una sola transacción de Ethereum consume lo mismo que un hogar medio en EE.UU. durante tres días, y su huella de carbono equivale a casi 8.000 horas viendo YouTube. Fuente: Digiconomist.

El dato es desde luego preocupante, sobre todo cuando lo comparamos con acciones mucho más «inocuas» como el envío de un correo electrónico (unos pocos gramos dependiendo del contenido, 50 si tiene muchos adjuntos) o ver una hora Netflix (unos 22 gramos según este estudio). La huella de carbono de un NFT es por lo tanto miles de veces mayor que el de esos dos

escenarios que, eso sí, son muchísimo más comunes.

Eso es cierto: en realidad la mayor parte de las transacciones se deben al Trading, es decir, a las operaciones de compraventa de distintas criptodivisas que ahora están teniendo especial popularidad en esta red por el auge de las finanzas descentralizadas (DeFi) y de plataformas como Uniswap.

Aquí de hecho Marcos hace una comparación con el consumo de la red bancaria mundial o con el consumo energético que por ejemplo generan Netflix o YouTube: entre ambos, indicaban en Statista, son responsables de una cuarta parte de todo el tráfico de internet mundial. Ahí es nada.

Pero si hablamos de comparaciones, hay otras que también dejan otra perspectiva sobre el consumo de Ethereum y los NFTs. En Loop News hablaban precisamente de este tema y revelaban que según la WWF fabricar una sola camiseta de manga corta de algodón requiere 494 kWh, que es la cantidad equivalente de hervir 2.700 litros de agua.

Eso hace que la energía consumida por ese aparentemente sencillo proceso sea 10 veces superior a la energía que consume una transacción de Ethereum. Teniendo en cuenta que crear y vender un NFT equivale unas cuatro de esas transacciones (sin incluir las subastas), esto parece dejar claro que aunque los NFTs consumen mucho, cosas aparentemente mundanas como fabricar una camiseta (o ver Netflix) también consumen una barbaridad.

En un artículo publicado en Bit2Me News, se habló sobre el informe emitido por la Agencia Internacional de la Energía (IEA) donde revela que Bitcoin consume solo el 0.3% de toda la energía que se usa en el planeta al año. En términos más precisos, la red Blockchain ocupa cerca de 80 TWh anuales, mientras que Ethereum consume cerca de 25 TWh al año.

Como explica el informe, estas cifras pueden parecer grandes en comparación con países como Irlanda, que consume cerca de 26 TWh; o de tecnologías emergentes como los vehículos eléctricos, que consumen cerca de 58 TWh al año, según cifras de 2018.

Pero en comparación con el consumo de la refrigeración, por ejemplo, el consumo energético de Bitcoin es bastante pequeño. Los sistemas de refrigeración en el mundo consumían cerca de 2,020 TWh anuales para 2018. Y si comparamos a Bitcoin con el consumo de YouTube, una popular plataforma de video, también se queda pequeño. Para finales de 2019, YouTube consumía cerca del 2% de la energía global, unos 243.6 TWh al año. Ahora imagina cuánta energía consumen plataforma gigantescas como Google, Apple o Facebook para mantener sus redes operativas. Estas plataformas están buscando alternativas que les permite reducir su huella de carbono en el

planeta, pero las criptomonedas también lo están haciendo.

En las criptomonedas, los mineros de Bitcoin están diseñando nuevos equipos mineros que requieran menos consumo energético para funcionar, y están preocupados por crear nuevas fuentes de energía limpia con las que garantizar la operatividad de la red con el menor impacto ambiental posible. En Estados Unidos y Rusia ya se desarrollan granjas mineras con fuentes de energía limpia que aprovechan las emisiones de CO2 de las grandes plantas petroleras y las transforman en electricidad aprovechable.

En el caso de Ethereum, la red se prepara para migrar hacia un nuevo protocolo de consenso llamado prueba de participación o Proof of Stake (PoS) que elimina la necesidad de los mineros, y que por ende, consume menos energía eléctrica.

Mientras todo esto ocurre, el criptoartista Beeple vendió su colección de criptoarte "Everydays - The First 5000 Days" por 69 millones de dólares, y el CryptoPunk 7804 se vendió por 7.5 millones de dólares recientemente.

La Mayoría de NFTs no Valen (al menos, por ahora) El fenómeno de los NFTs parece conquistarlo todo. Conquista el mundo del cine, del fútbol, de los videojuegos o de internet. Su avance parece imparable, y todos los que apuestan por esta tecnología parecen tener claro que este es el futuro del arte y los activos y contenidos digitales.

Eso, claro, ha hecho que se tengan muchas expectativas sobre los NFTs para invertir y como forma de ganar potencialmente mucho dinero, pero lo cierto es que la mayoría de los NFTs no tienen apenas valor. No lo decimos nosotros, lo dice un reciente estudio científico.

Solo Unos Pocos Elegidos Ganan Dinero el célebre caso del artista digital Beeple es continuamente puesto como recordatorio de hacia dónde pueden ir los NFTs. Que vendiera su NFT por 69 millones de dólares en la casa de subastas Christie's fue un verdadero bombazo, pero esa fue más bien la excepción que confirma la regla.

Un estudio publicado por The Alan Turing Institute en el 2021 y realizado por investigadores de la Universidad de Londres trató de estudiar si estos activos tienen realmente el valor que muchos les asignan. Andrea Baronchelli, profesor asociado de matemáticas, y Matthieu Nadini, científico de datos, analizaron la venta de 4.7 millones de NFTs que se intercambiaron entre 500,000 compradores y vendedores y que representaban casi 1,000 millones

de dólares en transacciones.

¿Su conclusión? La mayoría de NFTs no valen nada. Solo un 1% de todos los analizados se vendieron por más de 1,500 dólares, y el 75% se vendió por 15 dólares o menos. De hecho, explicaban "la mayoría de estos activos ni siquiera logran ser vendidos, así que ni siquiera llegaron a entrar en nuestros análisis".

Eso significa que la amplísima mayoría de NFTs ni siquiera entraban a formar parte de esos porcentajes, que serían mucho más bajos si se tuvieran en cuenta todos esos NFTs que están esperando a que alguien los compre.

Para Mauro Martino, responsable del Visual AI Lab en IBM, "la gente se gasta dinero en crear el NFT y ahí queda todo. Sería difícil sugerirle a un amigo artista que se metiera en este segmento para hacerse rico, porque muy poca gente puede lograr beneficios en este mercado".

Martino lleva también un año "mapeando la revolución NFT" en un proyecto que dio como resultado un vídeo y un artículo en la prestigiosa Nature. En su opinión, no es probable que los NFTs puedan hacerle a uno rico, pero al estudiar esta tecnología se dio cuenta de que los NFTs están aquí para quedarse.

4.5 La Web 3

En los últimos días del 2021, el término «Web3» se ha puesto de moda. Algunos creen que puede cambiar Internet para siempre, mientras que otros consideran que no es más que otra vuelta de tuerca en ese control de la Red que se lleva desarrollando desde hace años, pero desde una perspectiva diferente.

Entre los segundos, Jack Dorsey. El ex-CEO de Twitter ha estado en el centro de la polémica cuando el pasado 22 de diciembre utilizó su red social preferida para decir que «la Web3 no es un fuerza para democratizar la web, sino una herramienta para los fondos de capital riesgo». La declaración no sentó muy bien en determinados círculos y como el ambiente ya llevaba unos días enrarecido, Marc Andreessen, uno de los principales VC con intereses en lo que la Web3 representa, acabó bloqueando a Dorsey.

Pero para entender la razón del enfado de algunos fondos de inversión con Dorsey, o por qué el creador de Twitter se ha mostrado tan abiertamente contrario a esta idea, es necesario dar un paso atrás y explicar que es Web3, sobre qué principios se asienta y si finalmente, estos van a importar en algún momento.

De la Web 1.0 a la Web3 En el desarrollo de internet, los usuarios nos hemos relacionado con la Web 1.0 (desde sus inicios hasta probablemente 2004), la Web 2.0 (desde 2004 al momento actual) y algunos consideran que tímidamente comenzamos a hacerlos con la Web3.

La Web 1.0 se basaba en una idea auténticamente democratizadora de la Red y el uso de estándares y tecnologías abiertas. Además de las páginas HTML, es la era de los foros On-line, los grupos de noticias en Usenet, las charlas a través de clientes IRC, o el uso del FTP para la descarga de archivos. También en muchos casos representaba una conversación unidireccional, de uno hacia muchos.

El desarrollo de la Web 2.0 supone una democratización en esa charla, ya que permite conversaciones masivas de muchos a muchos, gracias a entre otras cosas, la popularización de las redes sociales. Pero también supone el desembarco de las grandes corporaciones en Internet, con sus cosas buenas como el impulso del comercio electrónico o el ayudarnos a conectar con otros, y las no tan buenas, como la creación de plataformas propietarias, el control de la conversación y la compra-venta indiscriminada de los datos personales de los usuarios con fines publicitarios y de control.

En el apogeo de esta Web 2.0 en la que todavía nos encontramos, comienza a fraguarse el concepto de Web3: una posible Internet del futuro en la que todos los datos y contenidos se registran en Blockchains, se Tokenizan y se accede a ellos en redes distribuidas P2P con el fin de democratizar Internet, poner el poder en manos de los creadores de contenidos y arrebatar el control a gobiernos y empresas.

Criptomonedas, NFT y Blockchain En esta definición de Web3 encontramos algunas de las ideas que ya estaban presentes en la Web 1.0, como la democratización y la descentralización de la información, pero que de alguna forma, se perdieron con el desarrollo de una por otro lado muy bien intencionada Web 2.0 en sus orígenes.

Por supuesto la idea de Web3 puede sonar bien...para los entusiastas de las criptomonedas, los que han empezado a hacer negocios vendiendo sus propios NFTs y sí, los fondos de capital riesgo que consideran que en ambos casos hay una gran oportunidad de negocio que se puede aprovechar.

Y sin embargo, no está del todo claro de qué forma se va a pasar de las palabras a los hechos o incluso si en algún momento esto va a pasar. El propio Elon Musk, que de criptomonedas sabe algo, ha afirmado recientemente que,

como el Metaverso, Web3 es solo «Marketing-hype» y él no lo entiende.

Llama la atención cómo dos de los términos que más se han repetido en los últimos meses (Metaverso y Web3) describen plataformas que no existen en realidad y de las que no se espera nada realmente interesante hasta como mínimo dentro de una década o más. Es probable por supuesto, que el negocio de las divisas virtuales, de las cadenas de bloques y los NFTs siga creciendo. Pero que la relación de estos tres conceptos de una forma aún por determinar acabe dando lugar a una nueva forma de concebir internet, es algo que se antoja muy lejano. En todo caso, la Web3 se quiere parecer más a ese «Next Big Thing» que buscan las compañías tecnológicas que a una revolución democratizadora que juegue a favor de los intereses de los usuarios. Y ahí Jack Dorsey seguramente esté en lo cierto.

Del Hype a la Realidad Por mucho que ese empeñen los defensores de Web3, lo cierto es que como aseguran en ComputerWord, a corto y medio plazo resultará básicamente imposible conseguir que usuarios, empresas y organizaciones se pongan de acuerdo en el desarrollo de una nueva internet basada en la cadena de bloques. Entre otras cosas porque para el usuario medio, que no crea contenidos monetizables, esto no tiene ningún valor.

Y no es que no se haya intentado. Frente a plataformas como Facebook o Twitter, empresas como iniciativas como Mastodon han propuesto a los usuarios el apostar por redes sociales auténticamente descentralizadas y no controladas por ninguna empresa. ¿El resultado? Los usuarios han seguido utilizando Facebook, Twitter, TikTok y YouTube.

Por otro lado, no parece muy probable que compañías como Meta, Google, Amazon o incluso Apple, que se han hecho mil millonarias gracias al desarrollo de la Web 2.0 estén muy por la labor de apostar por una red completamente descentralizada en la que son los usuarios los que toman el control y las grandes corporaciones el que lo pierden.

Incluso si esto fuera así, en un mundo ideal Web3 supone una participación activa de cada uno de los usuarios de esa futura Internet, como inversores de su propia «empresa personal». Como esto no es posible, son las Startups que previsiblemente actuarían como intermediarias en esta internet descentralizada (albergando por ejemplo esos contenidos Tokenizados), las que más tendrían que ganar y que de hecho aspiran, por la propia lógica capitalista de los negocios, a convertirse a grandes empresas que controlen esa nueva «internet democrática». Porque seamos serios: los inversores en

capital riesgo invierten para ganar dinero, no para hacer felices a los internautas.

Ni siquiera en estos momentos, el mundo de las criptomonedas, uno de esos pilares sobre los que se quiere fundar la nueva Web, es democrático o igualitario. Basta recordar en este sentido que un nuevo estudio publicado por Baystreet ponía el foco de atención en lo que cada vez resulta más evidente: el grueso de los Bitcoins en circulación está en cada vez menos manos (se calcula que el 0,01% de los titulares de Bitcoin ya controlan el 27% de las monedas en circulación).

En definitiva, una cosa es tener servicios basados en Blockchain y Tokenizados que funcionen sobre la Web que ya conocemos y otra muy distinta pensar que van a sustituir la infraestructura existente. Lo primero es desde luego muy real y va a ir a más; lo segundo, altamente improbable.

4.6 Protegiendo Nuestros Metadatos

¿Qué son los metadatos? los metadatos son "datos sobre datos" o "información sobre información" que se incluyen en archivos informáticos, normalmente de forma automática. Los metadatos se utilizan para describir, identificar, categorizar y ordenar archivos. Sin embargo, los metadatos también se pueden utilizar para desanonimizar a los usuarios y exponer información privada.

Ejemplos de metadatos incluyen:

- En archivos de imagen y vídeo:

  El lugar donde se tomó la foto o vídeo.

  La fecha y hora en que se tomó la foto o vídeo.

  El modelo y número de serie de la cámara utilizada.

- En archivos de documentos de texto:

  El autor del documento.

  Cambios en el documento.

Algunos tipos de archivos que guardan metadatos que pueden poner en peligro el anonimato de los usuarios:

- Audio Interchange File Format (.aiff)
- Audio Video Interleave (.avi)
- Electronic Publication (.epub)
- Free Lossless Audio Codec (.flac)
- Graphics Interchange Format (.gif)
- High Efficiency Image Format (.heic, .heif)
- Hypertext Markup Language (.html, .xhtml)
- Portable Network Graphics (PNG)
- JPEG (.jpeg, .jpg, ...)
- MPEG Audio (.mp3, .mp2, .mp1, .mpa)
- MPEG-4 (.mp4)
- Office Openxml (.docx, .pptx, .xlsx, ...)
- Ogg Vorbis (.ogg)
- Open Document (.odt, .odx, .ods, ...)
- Portable Document Fileformat (.pdf)
- Portable Pixmap Format (.ppm)
- Scalable Vector Graphics (.svg)
- Tape ARchive (.tar, .tar.bz2, .tar.gz, .tar.zx)
- Torrent (.torrent)
- Waveform Audio (.wav)
- Windows Media Video (.wmv)
- ZIP (.zip)

Es imposible encontrar y eliminar de manera confiable todos los metadatos en formatos de archivos complejos. Por ejemplo, los documentos de Microsoft Office pueden contener imágenes incrustadas, audio y otros archivos que contienen sus propios metadatos que no se pueden eliminar. Por ello se debe eliminar los metadatos de cualquier archivo antes de incrustarlo en otro documento.

Además, siempre que sea posible, debes guardar los archivos en formatos más simples. Por ejemplo, en lugar de guardar un documento de texto como un archivo .docx, puede guardarlo como un archivo .txt simple.

Metadata Anonymisation toolkit v2 permite eliminar metadatos de archivos antes de publicarlos o compartirlos, funciona en muchos formatos de archivos, incluidos:

- Archivos de imagen, como .jpeg, .png y .gif
- Archivos de LibreOffice, como .odt y .ods
- Documentos de Microsoft Office, como .docx, .xlsx y .pptx
- Archivos de audio, como .mp3, .flac y .ogg
- Archivos de vídeo, como .mp4 y .avi
- Archivar archivos, como .zip y .tar

En GNU/Linux podemos instalar el paquete mat2, mediante:

```
# apt install mat2
```

Para conocer la lista de archivos soportados usamos:

```
$ mat2 -l
```

Para visualizar los metadatos del archivo usamos:

```
$ mat2 -s nombreArchivo
```

Para remover los metadatos del archivo usamos:

```
$ mat2 -V nombreArchivo
```

en la hipótesis de eliminación de metadatos, debes tener en cuenta que mat2 no elimina el archivo en el que interviene porque deja el archivo original como está, sino que crea otro archivo.

Exchangeable Image File al tomar fotografías o vídeos es recomendable por seguridad desactivar el guardado de datos Exif (Exchangeable Image File) también conocidos como metadatos, ya que estos contienen información sobre la cámara, sobre la fotografía y sobre su origen como ubicación por GPS, la hora de creación, la última modificación, etc.

En GNU/Linux podemos instalar el paquete ExifTool que permite conocer y borrar los datos Exif en fotografías. Para instalarlo usamos:

```
# apt install libimage-exiftool-perl
```

Para visualizar los datos Exif, usamos:

```
$ exiftool imagen.gif
```

Para borrar todos los datos Exif, usamos:

```
$ exiftool -all=imagen.gif
```

5 Apéndice A: Software Libre y Propietario

Con el constante aumento de la comercialización de equipos de cómputo y/o comunicación (teléfonos inteligentes, tabletas, computadoras portátiles y de escritorio, etc.) y su relativo bajo costo, estos equipos se han convertido en objetos omnipresentes en nuestra vida diaria, ya que estos permiten realizar un creciente número de actividades cotidianas de miles de millones de usuarios.

Dichos equipos de cómputo y/o comunicación por sí solos tienen poca utilidad, pero su uso en conjunción con el Software adecuado forman un dúo que nos ha permitido tener los avances de los que actualmente disfrutamos. El Software -sistema operativo y los programas de aplicaciones- son los que realmente generan las soluciones al interactuar uno o más paquetes informáticos con los datos del usuario. También, es común que al comprar un equipo de cómputo y/o comunicación, en el costo total, se integre el del sistema operativo, aplicaciones ofimáticas y de antivirus, sean estos usados por el usuario o no y en la mayoría de los casos no es posible solicitar que no sean incluidos en el costo del equipo.

Por otro lado, el Software comercial suele quedar obsoleto muy rápido, ya que constantemente se le agregan nuevas funcionalidades al mismo y estas en general son vendidas como versiones independientes de la adquirida originalmente. Esto obliga al usuario -si quiere hacer uso de ellas- a comprar las nuevas versiones del Software para satisfacer sus crecientes necesidades informáticas y la obsolescencia programada.

Por lo anterior y dada la creciente complejidad de los paquetes de cómputo y el alto costo de desarrollo de aplicaciones innovadoras, en muchos casos, el costo total del Software que comúnmente los usuarios instalan -y que no necesariamente usan las capacidades avanzadas del programa, por las cuales el Software tiene un alto costo comercial- en sus equipos, suele ser más caro que el propio equipo en el que se ejecutan.

Hoy en día los usuarios disponemos de dos grandes opciones para adquirir el Software necesario para que nuestros equipos funcionen, a saber:

- Por un lado, podemos emplear programas comerciales (Software propietario), de los cuales no somos dueños del Software, sólo concesionarios al adquirir una licencia de uso del Software y nos proporcionan un instalable del programa adquirido. La licencia respectiva es en la gran mayoría de los casos muy restrictiva, ya que restringe su uso a un solo

equipo y/o usuario simultáneamente.

- Por otro lado, existe el Software libre[13], desarrollado por usuarios y para usuarios que, entre otras cosas, comparten los códigos fuente, el programa ejecutable y dan libertades para estudiar, adaptar y redistribuir a quien así lo requiera el programa y todos sus derivados.

Sobre la Obsolescencia Programada Es un conjunto de estrategias deliberadas destinadas a asegurarse que la versión actual de un determinado producto quedará desfasada o inservible en un plazo de tiempo predeterminado. De esta manera, los fabricantes se aseguran que los consumidores se verán obligados a reemplazarlo aunque funcione adecuadamente.

La obsolescencia puede lograrse mediante la introducción de un modelo con características superiores o diseñando intencionadamente un producto para que deje de funcionar correctamente en un plazo determinado. En cualquiera de los dos casos, se espera que los consumidores opten por el nuevo producto de la misma marca. Muchas veces la obsolescencia no es sobre el propio producto sino aplicando restricciones al producto de un competidor con la ayuda de una tercera empresa.

Tipos de Obsolescencia Programada Podemos dividir la obsolescencia programada en 4 tipos:

1- Establecimiento artificial del plazo de duración: Los productos se fabrican con piezas cuya duración tienen una vida útil limitada cuando, si se usaran otras de calidad superior ese plazo se extendería.

2- Actualizaciones de Software: Los desarrolladores de Software sacan nuevas versiones de sus aplicaciones que en un momento determinado dejan de ser compatibles con dispositivos antiguos. En muchos casos se ha podido comprobar que esa incompatibilidad es absolutamente artificial ya que al «engañar» al Software este funcionaba sin problemas.

[13] A veces también se han usado términos como FOSS y FLOSS. Ambas cosas son similares, ya que FOSS (Free and Open Source Software) traducido como "Software de código abierto" y FLOSS (Free/Libre and Open Source Software) "Software libre y de código abierto". Según quienes adoptan estos términos, lo hacen por tener una imparcialidad entre la carga filosófica del Software libre y el aspecto técnico y/o las ventajas que brinda este modelo de desarrollo. Richard Stallman nos invita a no usarlas y no se trata de un ad hóminem. Stallman y el proyecto GNU nos aconsejan que hablemos siempre de Software libre y aquí no cabe imparcialidad.

3- Obsolescencia percibida: Esta es una táctica psicológica, se trata de convencer al consumidor mediante publicidad y el uso de influenciadores de que el producto que se tiene actualmente está viejo y que se necesita uno nuevo. Como por ejemplo: ¿cuantos megapíxeles necesitas en tu teléfono para sacar una buena foto de tu mascota?

4- Trabas a la reparación: En el caso de los teléfonos por ejemplo, lo de impedir sacar la batería (con la excusa de hacer los teléfonos más delgados) es una forma de obligar a los consumidores a recurrir a los servicios oficiales y a disuadirlos de reemplazarlas por sustitutos más económicos. Otras tácticas son la utilización de piezas no estándar o que necesitan herramientas específicas para la reparación. Muchas veces se suele restringir el acceso a estas piezas o hacer una reducida producción de las mismas para aumentar artificialmente el costo.

Ejemplos de Obsolescencia Programada

- iPhone cada vez más lentos: La Justicia francesa comprobó que actualizaciones de Software hacían cada vez más lento el rendimiento de los modelos más viejos. La empresa le echó la culpa a las baterías, pero pagó una compensación de decenas de millones de dólares. Además rebajó los precios de sus baterías de repuesto para que los teléfonos fueran más rápidos con el nuevo Software y se comprometió a hacer más en el futuro para garantizar que los teléfonos no volvieran a ser más lentos. Con la salida de un nuevo modelo de teléfono cada año, seguro que hay algo de obsolescencia planificada en alguna parte.

- Impresoras: Esto es algo que todos conocemos. Muchas veces nos encontramos con impresoras a precio rebajado, pero al momento de tener que comprar un cartucho de tinta nos encontramos con que este tiene un precio igual o superior a comprar una nueva. Además, se ponen restricciones a la recarga o al uso de cartuchos alternativos. Hubo denuncias de que algunos modelos dejaban de funcionar a partir de cierta cantidad de páginas impresas o cierto tiempo desde la primera impresión.

- Certificados de seguridad: Por ejemplo, el pasado 30 de septiembre de 2021 caduco otro certificado de autentificación (CA de DST Root CA X3 de Let's Encrypt) que ayudaba a validar la conexión en internet a

los dispositivos que no fueron actualizados a otro certificado más actual -en la mayoría de los casos por no ser del interés económico de sus creadores-. Esto ocasionó que millones de dispositivos (teléfonos inteligentes, Smart TV, tabletas, computadoras portátiles y de escritorio, etc.) con algunos años de ser creados y perfectamente funcionales dejarán de conectarse a internet de un día para otro, forzando a sus dueños a desechar el dispositivo por carecer del servicio de internet en las aplicaciones instaladas.

- Cambio de la versión del sistema operativo: En el caso del sistema operativo Windows 10 a 11, la solicitud de requisitos mínimos de Hardware es para muchos equipos excesivo, ya que se estima que dejará fuera en su actualización a casi todos los equipos con más de 4 años de antigüedad por no contar por ejemplo con el Chip TPM 2.0 o GPU compatible con DirectX 12, siendo perfectamente funcionales con la versión actual del sistema operativo. Si bien Windows 10 seguirá con soporte hasta 2025, los usuarios que deseen tener las nuevas características del sistema operativo tendrán que cambiar de equipo.

5.1 Software Propietario

No existe consenso sobre el término a utilizar para referirse al opuesto del Software libre. La expresión «Software propietario (Proprietary Software)» (véase [4]), en la lengua anglosajona, "Proprietary" significa «poseído o controlado privadamente (Privately Owned and Controlled)», que destaca la manutención de la reserva de derechos sobre el uso, modificación o redistribución del Software. Inicialmente utilizado, pero con el inconveniente de que la acepción proviene de una traducción literal del inglés, no correspondiendo su uso como adjetivo en el español, de manera que puede ser considerado como un barbarismo.

El término "propietario" en español resultaría inadecuado, pues significa que «tiene derecho de propiedad sobre una cosa», por lo que no podría calificarse de "propietario" al Software, porque éste no tiene propiedad sobre nada (es decir, no es dueño de nada) y además, no podría serlo (porque es una cosa y no una persona). Así mismo, la expresión "Software propietario" podría ser interpretada como: "Software sujeto a propiedad" (derechos o titularidad) y su opuesto, el Software libre, también está sujeto al derecho de autor. Otra interpretación es que contrariamente al uso popular del término, se puede

afirmar que "todo Software es propietario", por lo que la forma correcta de referirse al Software con restricciones de uso, estudio, copia o mejora es la de Software privativo, según esta interpretación el término "propietario" podría aplicarse tanto para Software libre como Software privativo, ya que la diferencia entre uno y otro está en que el dueño del Software privativo lo licencia como propiedad privada y el de Software libre como propiedad social.

Con la intención de corregir el defecto de la expresión "Software propietario" aparece el llamado "Software con propietario", sin embargo se argumenta contra el término "con propietario" y justamente su similitud con Proprietary en inglés, que sólo haría referencia a un aspecto del Software que no es libre, manteniendo una de las principales críticas a éste (de "Software sujeto a derechos" o "propiedad"). Adicionalmente, si "propietario" se refiere al titular de los derechos de autor -y está claro que no se puede referir al usuario, en tanto éste es simplemente un cesionario-, no resuelve la contradicción: todo el Software libre tiene también titulares de derechos de autor.

La expresión Software no libre (en inglés Non-Free Software) es usado por la FSF para agrupar todo el Software que no es libre, es decir, incluye al llamado en inglés "Semi-Free Software" (Software semilibre) y al "Propietary Software". Asimismo, es frecuentemente utilizado para referirse al Software que no cumple con las Directrices de Software libre de Debian GNU/Linux, las cuales siguen la misma idea básica de libertad en el Software, propugnada por la FSF y sobre las cuales está basada la definición de código abierto de la Open Source Initiative.

Adicionalmente el Software de código cerrado nace como antónimo de Software de código abierto y por lo tanto se centra más en el aspecto de ausencia de acceso al código que en los derechos sobre el mismo, éste se refiere sólo a la ausencia de una sola libertad por lo que su uso debe enfocarse sólo a este tipo de Software y aunque siempre signifique que es un Software que no es libre, no tiene que ser Software de código cerrado.

La expresión Software privado es usada por la relación entre los conceptos de tener y ser privado. Este término sería inadecuado debido a que, en una de sus acepciones, la palabra "privado" se entiende como antónimo de "público", es decir, que «no es de propiedad pública o estatal, sino que pertenece a particulares», provocando que esta categoría se interpretará como no referente al Estado, lo que produciría la exclusión del Software no libre generado por el aparato estatal. Además, el "Software público" se asocia generalmente con Software de dominio público.

5.2 Software Libre

La definición de Software libre (véase [6], [7], [2], [3], [1] y [5]) estipula los criterios que se tienen que cumplir para que un programa sea considerado libre. De vez en cuando se modifica esta definición para clarificarla o para resolver problemas sobre cuestiones delicadas. «Software libre» significa que el Software respeta la libertad de los usuarios y la comunidad. En términos generales, los usuarios tienen la libertad de copiar, distribuir, estudiar, modificar y mejorar el Software. Con estas libertades, los usuarios -tanto individualmente como en forma colectiva- controlan el programa y lo que hace.

Cuando los usuarios no controlan el programa, el programa controla a los usuarios. Los programadores controlan el programa y a través del programa, controlan a los usuarios. Un programa que no es libre, llamado «privativo o propietario», es considerado por muchos como un instrumento de poder injusto.

El Software libre es la denominación del Software que respeta la libertad de todos los usuarios que adquirieron el producto y por tanto, una vez obtenido el mismo puede ser usado, copiado, estudiado, modificado y redistribuido libremente de varias formas. Según la Free Software Foundation (véase [6]), el Software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir y estudiar el mismo, e incluso modificar el Software y distribuirlo modificado.

Un programa es Software libre si los usuarios tienen las cuatro libertades esenciales:

0. La libertad de usar el programa, con cualquier propósito.

1. La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.

2. La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.

3. La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.

Un programa es Software libre si los usuarios tienen todas esas libertades. Por tanto, el usuario debe ser libre de redistribuir copias, tanto con o sin modificaciones, ya sea gratuitamente o cobrando una tarifa por la distribución,

a cualquiera en cualquier parte. El ser libre de hacer estas cosas significa, entre otras cosas, que no tiene que pedir ni pagar el permiso.

También debe tener la libertad de hacer modificaciones y usarlas en privado para su propio trabajo o pasatiempo, sin siquiera mencionar que existen. Si publica sus cambios, no debe estar obligado a notificarlo a nadie en particular, ni de ninguna manera.

La libertad de ejecutar el programa significa que cualquier tipo de persona u organización es libre de usarlo en cualquier tipo de sistema de computación, para cualquier tipo de trabajo y finalidad, sin que exista obligación alguna de comunicarlo al programador ni a ninguna otra entidad específica. En esta libertad, lo que importa es el propósito de los usuarios, no el de los programadores. El usuario es libre de ejecutar el programa para alcanzar sus propósitos y si lo distribuye a otra persona, también esa persona será libre de ejecutarlo para lo que necesite; nadie tiene derecho a imponer sus propios objetivos.

La libertad de redistribuir copias debe incluir las formas binarias o ejecutables del programa, así como el código fuente, tanto para las versiones modificadas como para las que no lo estén. Distribuir programas en forma de ejecutables es necesario para que los sistemas operativos libres se puedan instalar fácilmente. Resulta aceptable si no existe un modo de producir un formato binario o ejecutable para un programa específico, dado que algunos lenguajes no incorporan esa característica, pero debe tener la libertad de redistribuir dichos formatos si encontrara o programara una forma de hacerlo.

Para que se de la libertad que se menciona en los puntos 1 y 3 de realizar cambios y publicar las versiones modificadas tenga sentido, el usuario debe tener acceso al código fuente del programa. Por consiguiente, el acceso al código fuente es una condición necesaria para el Software libre. El «código fuente» compilado no es código fuente real y no cuenta como código fuente.

La libertad 1 incluye la libertad de usar su versión modificada en lugar de la original. Si el programa se entrega con un producto diseñado para ejecutar versiones modificadas de terceros, pero rechaza ejecutar las suyas, una práctica conocida como «tivoización» o «arranque seguro» [«Lockdown»] la libertad 1 se convierte más en una ficción teórica que en una libertad práctica, esto no es suficiente, en otras palabras, estos binarios no son Software libre, incluso si se compilaron desde un código fuente que es libre.

Una manera importante de modificar el programa es agregándole subrutinas y módulos libres ya disponibles. Si la licencia del programa especifica que no se pueden añadir módulos que ya existen y que están bajo una licencia

apropiada, por ejemplo si requiere que usted sea el titular de los derechos de autor del código que desea añadir, entonces se trata de una licencia demasiado restrictiva como para considerarla libre.

La libertad 3 incluye la libertad de publicar sus versiones modificadas como Software libre. Una licencia libre también puede permitir otras formas de publicarlas; en otras palabras, no tiene que ser una licencia de Copyleft. No obstante, una licencia que requiera que las versiones modificadas no sean libres, no se puede considerar libre.

«Software libre» no significa que «no es comercial». Un programa libre debe estar disponible para el uso comercial, la programación comercial y la distribución comercial. La programación comercial de Software libre ya no es inusual; el Software libre comercial es muy importante, ejemplo de ello es la empresa RedHat (ahora propiedad de IBM). Puede haber pagado dinero para obtener copias de Software libre, o puede haber obtenido copias sin costo. Pero sin tener en cuenta cómo obtuvo sus copias, siempre tiene la libertad de copiar y modificar el Software, incluso de vender copias.

El término Software no libre se emplea para referirse al Software distribuido bajo una licencia de Software más restrictiva que no garantiza estas cuatro libertades. Las leyes de la propiedad intelectual reservan la mayoría de los derechos de modificación, duplicación y redistribución para el dueño del Copyright; el Software dispuesto bajo una licencia de Software libre rescinde específicamente la mayoría de estos derechos reservados.

Los manuales de Software deben ser libres por las mismas razones que el Software debe ser libre y porque de hecho los manuales son parte del Software. También tiene sentido aplicar los mismos argumentos a otros tipos de obras de uso práctico, es decir, obras que incorporen conocimiento útil, tal como publicaciones educativas y de referencia. Wikipedia es el ejemplo más conocido.

La lista de proyectos de este tipo es realmente impresionante, algunos han conseguido un uso y alta calidad, por ejemplo el compilador GCC, el Kernel de Linux y el sistema operativo Debian GNU/Linux y Android. Mientras que otros proyectos han caído en el olvido, pero de la gran mayoría se tiene copia del código fuente que permitiría a quienes estén interesados en dicho proyecto poder reusarlo y en su caso ampliarlo.

La característica más importante que aparece típicamente en un proyecto de este tipo, es que un conjunto de personas separadas a gran distancia, sean capaces, a través de la Web, de los E-mail y de foros de aunar sus esfuerzos para crear, mejorar y distribuir un producto, de forma que todos

ellos se benefician unos de otros. Evidentemente, la mayor parte del peso recae en los desarrolladores, pero también es necesaria una difusión para que los usuarios documenten, encuentren errores, hagan foros de discusión, etc.

Si bien, el Software libre no es más seguro (en el sentido de invulnerable) que el propietario, la diferencia estriba en que el código fuente en el Software libre está disponible para todos y cualquiera puede aportar una solución, y por lo general al poco tiempo de detectarse una vulnerabilidad (a veces en cuestión de horas) se puede disponer de una solución para la misma. Además, al tener acceso al código fuente se puede detectar fácilmente si alguien introdujo código malicioso a una determinada aplicación.

¿Por qué se Interesan los Autores, Alumnos y Profesores Universitarios en el Software Libre? La ventaja principal es porque bajo el Software libre subyace la idea de compartir conocimiento y favorecer la existencia de nuevas ideas[14]; y ¿qué es investigar y enseñar?, sino crear conocimiento y procurar que los alumnos aprendan e incluso vayan más allá de lo aprendido. Se comparte la idea, que el espíritu del Software libre es similar al que debería reinar en las instituciones educativas:

- Porque así no se condiciona a los estudiantes a usar siempre lo mismo.
- No se fomenta la piratería en los estudiantes y se evita pagar licencias que no son necesarias al existir alternativas gratuitas.
- Es mucho más seguro ya que el Software libre es público y se puede ver qué hace exactamente sin recelos.
- Se ofrece libertad de elección a los estudiantes y profesores al no limitarlos a usar una solución determinada, ampliando sus opciones y permitiendo un mayor aprendizaje.

Concretando estas ideas, profesores e investigadores necesitan herramientas para la investigación y docencia y estas deben tener una calidad mínima y ser fácilmente distribuibles entre los alumnos. En muchos casos las compañías desarrolladoras y distribuidoras de programas de cómputo no han

[14] ¿Por qué el Software creado con dinero de los impuestos no se publica como Software Libre?

¡El código pagado por los ciudadanos debería estar disponible para los ciudadanos y el mismo gobierno!

sabido ofrecer sus productos con la flexibilidad adecuada para las labores docentes o, en otros casos, los productos desarrollados no tienen la calidad esperada.

El Software libre es aún joven, pese a las decenas de miles de proyectos actuales -en los que se trabaja constantemente en mejorar la parte computacional de los algoritmos involucrados en el proyecto, haciendo y puliendo interfaces gráficas, generando ayuda en línea así como la documentación necesaria para que usuarios noveles y avanzados usen la mayor cantidad de opciones programadas- existen muchas otras necesidades profesionales y de investigación que requieren el desarrollo innovador de programas de cómputo para automatizarlas y hacerlas eficientes. Esto queda plasmado en las decenas de proyectos que a diario son registrados en las páginas especializadas en busca de difusión y apoyo para su proyecto.

En los últimos años, muchos proyectos han pasado de ser simples programas en línea de comandos a complejas aplicaciones multiplataforma -se ejecutan en distintos sistemas operativos como son Windows, Linux, Unix, Mac OS, Android- con ambientes gráficos multimedia que en muchos casos han superado a sus contrapartes comerciales -por ejemplo los navegadores Web-. Para muestra de este maravilloso avance, tomemos el proyecto del sistema operativo Android, que actualmente se ejecuta en millones de equipos -como celulares, tabletas, electrodomésticos, etc.- y en los cuales se pueden descargar miles de aplicaciones y está soportado por una gran cantidad de usuarios y empresas comerciales como Google, IBM y últimamente Microsoft -que años atrás era acérrima enemiga del Software libre-.

El Software libre ha logrado desplazar a muchos de sus competidores por sus múltiples bondades y bajo costo de desarrollo -es el caso de Windows Phone que fue reemplazado por Android de Google-, al reusar miles de aplicaciones ya existentes que usan Software libre y permitir desarrollar otro tanto de aplicaciones bajo una plataforma que se ejecuta en los más diversos procesadores. Además, el uso de Software libre y su posibilidad de ampliarlo y/o especializarlo según sea necesario, ha permitido crear de forma cada vez más rápida y confiable; para poner a disposición de un gran público programas de uso común, así como especializado que satisfagan las nuevas necesidades de los usuarios.

Software Libre en Ciencia y Educación Algunos puntos y reflexiones sobre porqué se considera que es interesante el Software libre en Ciencia y

Educación son:

- Accesible a todo el mundo aunque no sea rentable su desarrollo: El Software libre entre sus libertades permite que se pueda ejecutar por terceros, copiarlo, distribuirlo y estudiarlo/modificarlo. Eso hace que si en ciencia se usa Software libre el acceso a esos programas no suponga una barrera (se puede distribuir y ejecutar).

- Muchos de los desarrollos en el campo de la accesibilidad se realizan en universidades y son distribuidos como Software libre: Aunque no sea rentable muchas veces el desarrollo de herramientas de accesibilidad (no sea algo monetizable) en la universidad se consigue escapar a esa la lógica capitalista de solamente invertir en lo que pueda ofrecer beneficio económico.

- Transparencia: En ciencia es importante ver las costuras para comprobar si es verdad lo que se afirma, tener acceso al código fuente del Software empleado permite poder estudiarlo por si realiza algún cálculo mal.

- Propicia el espíritu crítico: Si no tienes acceso a las revistas o el acceso es privativo para los bolsillos de mucha gente no puedes comprobar la información. Se nos pide que seamos críticos con la información que se nos da del mundo científico pero no podemos entrenar el espíritu crítico sin acceso al conocimiento libre.

- Caramelos con droga en la puerta del colegio: Muchas empresas buscan introducir en los colegios, institutos y universidad su Software. Ofrecer un programa que permita trabajar y genere una dependencia quedando los datos muchas veces en las nubes (ordenadores de otras personas). Un ejemplo es Microsoft con Office 365. Dando cuentas gratuitas durante un tiempo para que se use su Software. Otro ejemplo podría ser Unity3D en vez de por ejemplo Godot.

- Especificaciones de protocolos abiertas VS cerradas: Gracias a que los protocolos TCP/IP, HTTP, POP, SNMP, DHCP, etc. son abiertos es posible construir herramientas por cualquier con conocimientos de programación. Con protocolos cerrados solamente quienes tuvieran acceso a las especificaciones podrían desarrollar y conocer cómo funcionan.

- Uso de estándares: Existiendo un estándar para documentos ofimáticos (procesador de textos, hoja de cálculo, presentaciones, etc.) algunas empresas como Microsoft se empeñan en ir con su propio formato y estándar en vez de sumarse a que sea más sencillo ir a una y que el usuario pueda optar por que herramientas usar para editar o trabajar con documentos ofimáticos.

- Software libre para la Ciencia ciudadana: Un ejemplo en el que es importante la colaboración ciudadana es el cambio climático y la defensa medioambiental del territorio. Desde imvec.tech usan herramientas de Software libre para medición y monitorización de contaminación.

Software Libre: Beneficios Más Allá de la Informática El uso de las tecnologías de código abierto supone cultivar el conocimiento y la puesta en valor de la libertad individual, lo personal y lo privado, todo ello sin menospreciar lo público y la construcción de una sociedad. El movimiento del Software libre, con Linux a la cabeza, ha capitaneado durante décadas planteamientos para un cambio en el modo de producción: el abaratamiento de costes empresariales para grandes y pequeños, el trabajo en línea, el desarrollo de Software y Hardware a pequeña escala, el replanteamiento del negocio informático, el sistema de normas éticas que rigen los grupos, la documentación abierta, etc.

Todo ello derivado de una simple idea: la libertad. Ha sido la clave que ha llevado a todo este movimiento hacia una autonomía y motivación que pocas veces se ve en otros sectores. El Open Source está lleno de alternativas con la libertad como pilar y consecuencia filosófica, de hecho muchos Forks (derivaciones) de proyectos aparecen cuando la disputa sobre la misma toma relevancia. Esta manera de hacer las cosas debería trasladarse directamente a la sociedad promoviendo esos valores para evitar caer en un mundo despótico que toma fuerza a pasos agigantados.

No estaría mal promover la comprensión de las licencias libres. Leer y entender una licencia GPL, MIT o BSD es infinitamente más sencillo y rápido que hacerlo con otras, siendo unas normas fáciles de cumplir porque encajan con un modelo ético y práctico comprensible por muchos.

Podríamos decir que GPL, BSD y MIT son las Constituciones que vertebran todo el movimiento del Software Libre, cosechando derechos de uso y logros como el ahorro o la legítima copia privada. Lo mismo podría ocurrir con las licencias Creative Commons para cierto contenido, un arma poderosa

en el sector divulgativo que está poco extendida por desconocimiento y el Status Quo de la propiedad intelectual.

La cooperación libre y voluntaria supuso un éxito hasta ahora pero está siendo amenazada por la dinámica actual de la Web, donde la centralización de las principales plataformas supone estar bajo el yugo de normas que rastrean con lupa y cambian sus términos con demasiada frecuencia. Ya ni digamos cuando eso se junta con el ansia de monetización: si quieres dinero más te vale no cabrear a los anunciantes o no tener un Strike por uso de contenido que podría ser reclamado.

Los claroscuros de este sistema hace que los creadores cada vez hagan menos de forma libre y altruista, y ello podría verse potenciado por las búsquedas con inteligencia artificial, lo que apunta a un empobrecimiento del contenido cultural fresco y renovador. Las polémicas con GitHub Copilot o ChatGPT sobre el entrenamiento de las IAs hace sobrevolar una vez más la cuestión del Copyright y el uso legítimo de los resultados brindados por éstas, lo que también condiciona la creación.

La libertad de expresión, de uso, de creación, de modificación ... esas cuestiones llevan irremediablemente a una libertad de pensamiento y acción, a una adaptación creativa que puede ser la motivación para romper moldes en todos los aspectos. El código abierto y sus licencias son, por tanto, un beneficio personal y social mucho más grande que el simple hecho de usar Linux o Software libre. El contenido despreocupado, que no prioriza el dinero y el posicionamiento/visualizaciones, se vuelve esencial para el inconformismo.

5.3 Seguridad del Software

Si bien, el Software Libre no es más seguro (en el sentido de invulnerable) que el propietario, la diferencia puede estribar en que el código fuente en el Software libre está disponible para todos y cualquiera puede aportar una solución y por lo general al poco tiempo de detectarse una vulnerabilidad (a veces en cuestión de horas) se puede disponer de una solución para la misma. Además, al tener acceso al código fuente se puede detectar si alguien introdujo código malicioso a una determinada aplicación.

Pero de todos es sabido, que los usuarios de Software de código abierto, como por ejemplo los que de manera habitual trabajan con equipos comandados por sistemas Linux, por regla general se sienten orgullosos de la seguridad que estos programas aportan con respecto a los sistemas cerrados propios de otras firmas, dígase Microsoft Windows o Mac de Apple.

¿Es Seguro el Software Libre? En primer lugar definiremos el concepto de "seguridad" como salvaguarda de las propiedades básicas de la información. Entre las características que debe cumplir para ser seguro, encontramos la integridad, es decir, que sólo los usuarios autorizados pueden crear y modificar los componentes del sistema, la confidencialidad, sólo estos usuarios pueden acceder a esos componentes, la disponibilidad, que todos los componentes estén a disposición de los usuarios siempre que lo deseen y el "no repudio", o lo que es lo mismo, la aceptación de un protocolo de comunicación entre el servidor y un cliente, por ejemplo, mediante certificados digitales.

Entre las diferencias de seguridad entre un Software Libre y el Software Propietario, podemos destacar:

- Seguridad en el Software Propietario: En el caso de tener "agujeros de seguridad", puede que no nos demos cuenta y que no podamos repararlos. Existe una dependencia del fabricante, retrasándose así cualquier reparación y la falsa creencia de que es más seguro por ser oscuro (la seguridad por oscuridad determina los fallos de seguridad no parcheados en cada producto).

- Seguridad en el Software Libre: Por su carácter público y su crecimiento progresivo, se van añadiendo funciones y se nos permite detectar más fácilmente los agujeros de seguridad para poder corregirlos. Los problemas tardan mucho menos en ser resueltos por el apoyo que tiene de los Hackers y una gran comunidad de desarrolladores y al ser un Software de código libre, cualquier empresa puede aportar soporte técnico.

Sin embargo esta es una pregunta sobre la que los expertos al día de hoy, tras muchos años de discusiones, siguen sin ponerse de acuerdo. ¿Es más seguro el Software de código abierto que los programas cerrados, o viceversa? Lo cierto es que, en términos generales, ambos bandos tienen sus razones con las que defender sus argumentos. Por un lado, los usuarios de las aplicaciones y sistemas de código abierto, defienden que, al estar el código fuente disponible a los ojos de todo el mundo, es mucho más fácil localizar posibles agujeros de seguridad y vulnerabilidades que pongan en peligro los datos de los usuarios.

Por otro lado, aquellos que consideran que los sistemas cerrados son más seguros en este sentido, afirman que al tener acceso tan solo los expertos al código fuente de sus aplicaciones, es más complicado que se produzcan

filtraciones o inserciones de Software malicioso en este tipo de sistemas. Hay que tener en cuenta que, por ejemplo, Google premia a las personas que descubren fallos de seguridad en su Software como Chrome, aunque no es el único gigante de la tecnología en utilizar estas tácticas.

De hecho muchas empresas están gastando miles de millones de dólares y/o euros en hacer que sus propuestas sean lo más seguras posible, argumentando que la seguridad de sus proyectos es una de sus prioridades, todo con el fin de intentar frenar que los atacantes vulneren sus sistemas. Por otro lado, otros aseguran que cuando el código fuente es público, más ojos están disponibles para detectar posibles vulnerabilidades o errores en dicho código, por lo que siempre será más rápido y sencillo poner soluciones con el fin de ganar en seguridad.

Sea como sea, en cualquiera de los dos casos, lo que ha quedado más que demostrado es que la seguridad no está garantizada en ningún momento, ya sean propuestas de código abierto, o no. Pero también es cierto que lo que se procura es que los riesgos de ser atacados se reduzcan en medida de lo posible. Los sistemas Linux son considerados desde hace mucho tiempo como un sistema operativo seguro, en buena parte debido a las ventajas que ofrece su diseño. Dado que su código está abierto, son muchas las personas que incorporan mejoras de las que el resto de usuarios de Linux se benefician, a diferencia de las propuestas de Windows o MacOS, donde estas correcciones generalmente se limitan a las que detectan Microsoft y Apple.

No obstante, en nuestra defensa del Software libre, diremos que su código abierto permite que los errores sean encontrados y solucionados con mayor rapidez, por lo que determinamos que es el Software más recomendable.

En general, puede afirmarse que el Software libre es más seguro, ya que debido a su carácter abierto y distribuido, un gran número de programadores y personas expertas pueden estar atentas al código fuente -especialmente en los grandes proyectos-, lo cual permite hacer auditorías con objeto de detectar errores y puertas traseras (Backdoor, en inglés) que pongan en riesgo nuestros datos.

Así, los grandes programas y proyectos de Software libre, con una extensa comunidad de desarrollo y usuarios que lo respalden, presentan niveles muy altos de seguridad, un alto grado de protección y una rápida respuesta a posibles vulnerabilidades.

5.4 Tipos de Licencias

Tanto la Open Source Initiative como la Free Software Foundation mantienen en sus páginas Web (véase [6], [7], y [5]) listados oficiales de las licencias de Software libre que aprueban.

Una licencia es aquella autorización formal con carácter contractual que un autor de un Software da a un interesado para ejercer "actos de explotación legales". Pueden existir tantas licencias como acuerdos concretos se den entre el autor y el licenciatario. Desde el punto de vista del Software libre, existen distintas variantes del concepto o grupos de licencias:

Licencias GPL Una de las más utilizadas es la Licencia Pública General de GNU (GNU GPL). El autor conserva los derechos de autoría (Copyright), y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del Software permanecen bajo los términos más restrictivos de la propia GNU GPL. Esto hace que sea imposible crear un producto con partes no licenciadas GPL, el conjunto tiene que ser GPL.

En la práctica, esto hace que las licencias de Software libre se dividan en dos grandes grupos, aquellas que pueden ser mezcladas con código licenciado bajo GNU GPL (y que inevitablemente desaparecerán en el proceso, al ser el código resultante licenciado bajo GNU GPL) y las que no lo permiten al incluir mayores u otros requisitos que no contemplan ni admiten la GNU GPL y que por lo tanto no pueden ser enlazadas ni mezcladas con código gobernado por la licencia GNU GPL.

GPL Versión 1 la versión 1 de GNU GPL, fue presentada el 25 de febrero de 1989, impidió lo que eran las dos principales formas con las que los distribuidores de Software restringían las libertades definidas por el Software libre. El primer problema fue que los distribuidores publicaban únicamente los archivos binarios, funcionales y ejecutables, pero no entendibles o modificables por humanos. Para prevenir esto, la GPLv1 estableció que cualquier proveedor de Software libre además de distribuir el archivo binario debía liberar a su vez código fuente entendible y que pudiera ser modificado por el ser humano bajo la misma licencia (secciones 3a y 3b de la licencia).

El segundo problema era que los distribuidores podían añadir restricciones adicionales, añadiendo restricciones a la licencia o mediante la combinación del Software con otro que tuviera otras restricciones en su distribución. Si

esto se hacía, entonces la unión de los dos conjuntos de restricciones sería aplicada al trabajo combinado, entonces podrían añadirse restricciones inaceptables. Para prevenir esto, GPLv1 obligaba a que las versiones modificadas en su conjunto, tuvieran que ser distribuidas bajo los términos GPLv1 (secciones 2b y 4 de la licencia). Por lo tanto, el Software distribuido bajo GPLv1 puede ser combinado con Software bajo términos más permisivos y no con Software con licencias más restrictivas, lo que entraría en conflicto con el requisito de que todo Software tiene que ser distribuido bajo los términos de la GPLv1.

GPL Versión 2 según Richard Stallman, el mayor cambio en GPLv2 fue la cláusula "Liberty or Death" («libertad o muerte»). Esta sección dice que si alguien impone restricciones que le prohíben distribuir código GPL de tal forma que influya en las libertades de los usuarios (por ejemplo, si una ley impone que esa persona únicamente pueda distribuir el Software en binario), esa persona no puede distribuir Software GPL. La esperanza es que esto hará que sea menos tentador para las empresas el recurrir a las amenazas de patentes para exigir una remuneración de los desarrolladores de Software libre.

En 1991 se hizo evidente que una licencia menos restrictiva sería estratégicamente útil para la biblioteca C y para las bibliotecas de Software que esencialmente hacían el trabajo que llevaban a cabo otras bibliotecas comerciales ya existentes. Cuando la versión 2 de GPL fue liberada en junio de 1991, una segunda licencia Library General Public License fue introducida al mismo tiempo y numerada con la versión 2 para denotar que ambas son complementarias. Los números de versiones divergieron en 1999 cuando la versión 2.1 de LGPL fue liberada, esta fue renombrada como GNU Lesser General Public License para reflejar su lugar en esta filosofía.

GPL Versión 3 A finales de 2005, la Free Software Foundation (FSF) anunció estar trabajando en la versión 3 de la GPL (GPLv3). El 16 de enero de 2006, el primer borrador de GPLv3 fue publicado y se inició la consulta pública. La consulta pública se planeó originalmente para durar de nueve a quince meses, pero finalmente se extendió a dieciocho meses, durante los cuales se publicaron cuatro borradores. La GPLv3 oficial fue liberada por la FSF el 29 de junio de 2007.

Según Stallman los cambios más importantes se produjeron en el campo

de las patentes de Software, la compatibilidad de licencias de Software libre, la definición de código fuente y restricciones a las modificaciones de Hardware. Otros cambios están relacionados con la internacionalización, cómo son manejadas las violaciones de licencias y cómo los permisos adicionales pueden ser concedidos por el titular de los derechos de autor. También añade disposiciones para quitar al DRM su valor legal, por lo que es posible romper el DRM (Digital Rights Management) en el Software de GPL sin romper leyes como la DMCA (Digital Millennium Copyright Act).

GPLv2 vs GPL v3 GPLv3 contiene la intención básica de GPLv2 y es una licencia de código abierto con un Copyleft estricto. Sin embargo, el idioma del texto de la licencia fue fuertemente modificado y es mucho más completo en respuesta a los cambios técnicos, legales y al intercambio internacional de licencias.

La nueva versión de la licencia contiene una serie de cláusulas que abordan preguntas que no fueron o fueron cubiertas de manera insuficiente en la versión 2 de la GPL. Las nuevas regulaciones más importantes son las siguientes:

- GPLv3 contiene normas de compatibilidad que hacen que sea más fácil combinar el código GPL con el código que se publicó bajo diferentes licencias. Esto se refiere en particular al código bajo la licencia de Apache v. 2.0.

- Se insertaron normas sobre gestión de derechos digitales para evitar que el Software GPL se modifique a voluntad, ya que los usuarios recurrieron a las disposiciones legales para protegerse mediante medidas técnicas de protección (como la DMCA o la directiva sobre derechos de autor).

- La licencia GPLv3 contiene una licencia de patente explícita, según la cual las personas que licencian un programa bajo licencia GPL otorgan derechos de autor y patentes, en la medida en que esto sea necesario para utilizar el código que ellos otorgan. Por lo tanto, no se concede una licencia de patente completa. Además, la nueva cláusula de patente intenta proteger al usuario de las consecuencias de los acuerdos entre los titulares de patentes y los licenciatarios de la licencia pública general que solo benefician a algunos de los licenciatarios (correspondientes al

acuerdo Microsoft / Novell). Los licenciatarios deben garantizar que todos los usuarios disfrutan de tales ventajas (licencia de patente o liberación de reclamos) o que nadie puede beneficiarse de ellos.

- A diferencia de la GPLv2, la GPLv3 establece claramente que no es necesario divulgar el código fuente en un uso ASP (Application Service Provider) de los programas GPL, siempre que no se envíe una copia del Software al cliente. Si el efecto Copyleft debe extenderse al uso de ASP, debe aplicarse la Licencia pública general de Affero, versión 3 (AGPL) que solo difiere de la GPLv3 en esta consideración.

Licencias Estilo BSD Llamadas así porque se utilizan en gran cantidad de Software distribuido junto a los sistemas operativos BSD. El autor, bajo tales licencias, mantiene la protección de Copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación, incluso si dichos trabajos tienen propietario. Son muy permisivas, tanto que son fácilmente absorbidas al ser mezcladas con la licencia GNU GPL con quienes son compatibles. Puede argumentarse que esta licencia asegura "verdadero" Software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al Software y que puede decidir incluso redistribuirlo como no libre.

Licencia Copyleft Hay que hacer constar que el titular de los derechos de autor (Copyright) de un Software bajo licencia Copyleft puede también realizar una versión modificada bajo su Copyright original y venderla bajo cualquier licencia que desee, además de distribuir la versión original como Software libre. Esta técnica ha sido usada como un modelo de negocio por una serie de empresas que realizan Software libre (por ejemplo MySQL); esta práctica no restringe ninguno de los derechos otorgados a los usuarios de la versión Copyleft.

Licencia estilo MIT Las licencias MIT son de las más permisivas, casi se consideran Software de dominio público. Lo único que requieren es incluir la licencia MIT para indicar que el Software incluye código con licencia MIT.

Licencia Apache License La licencia Apache trata de preservar los derechos de autor, incluir la licencia en el Software distribuido y una lista de

los cambios realizados. En modificaciones extensivas del Software original permite licenciar el Software bajo otra licencia sin incluir esas modificaciones en el código fuente.

Licencia Mozilla Public License MPL Esta licencia requiere que los archivos al ser distribuidos conserven la misma licencia original pero pueden ser usados junto con archivos con otra licencia, al contrario de la licencia GPL que requiere que todo el código usado junto con código GPL sea licenciado como código GPL. También en caso de hacer modificaciones extensivas permite distribuirlas bajo diferentes términos y sin incluir el código fuente en las modificaciones.

Licencia Código de Dominio Público Es un código que no está sujeto a derechos de autor que puede utilizarse sin restricciones.

Licencia Creative Commons Las licencias de Creative Commons son más utilizadas para cualquier creación digital que para el Software, entendiendo como creación digital desde fotos, artículos en blogs, música, vídeos, este trabajo, etc. Hay varios tipos de licencias de Creative Commons diferenciando entre permitir modificaciones a la obra original, solicitando crédito de la creación o permitiendo un uso comercial de la obra.

Licencias de Código Abierto Las licencias de código abierto son un intermedio entre las licencias privativas y las licencias de Software libre. Las licencias de código abierto permiten el acceso al código fuente pero no todas se consideran licencias de Software libre al no otorgar otros derechos que se requieren para considerar un Software como Software libre como el derecho al uso o con cualquier propósito, modificación y distribución.

Dado el éxito del Software libre como modelo de desarrollo de Software algunas empresas cuyo Software era privativo pueden decidir hacerlo de código abierto con la intención de suplir algunas carencias de Software privativo pero sin perder ciertos derechos que son la fuente de sus ingresos como la venta de licencias.

Las expresiones «Software libre» y «código abierto» se refieren casi al mismo conjunto de programas. No obstante, dicen cosas muy diferentes acerca de dichos programas, basándose en valores diferentes. El movimiento del Software libre defiende la libertad de los usuarios de ordenadores, en un

movimiento en pro de la libertad y la justicia. Por contra, la idea del código abierto valora principalmente las ventajas prácticas y no defiende principios. Esta es la razón por la que gran parte de la comunidad de Software libre está en desacuerdo con el movimiento del código abierto y nosotros no empleamos esta expresión en este texto.

Licencia Microsoft Public License La Microsoft Public License es una licencia de código abierto que permite la distribución del Software bajo la misma licencia y la modificación para un uso privado. Tiene restricciones en cuanto a las marcas registradas.

En caso de distribuir el Software de forma compilada o en forma de objeto binario no se exige proporcionar los derechos de acceso al código fuente del Software compilado o en forma de objeto binario. En este caso esta licencia no otorga más derechos de los que se reciben, pero si permite otorgar menos derechos al distribuir el Software (compilado o en forma de objeto binario).

Modelo de Desarrollo de Software Bazar y Catedral El tipo de licencia no determina qué Software es mejor o peor, si el privativo o el Software libre, la diferencia entre las licencias está en sus características éticas y legales. Aunque el modelo de desarrollo con una licencia de código abierto a la larga suele tener un mejor desarrollo y éxito que el Software privativo, más aún con un medio como internet que permite colaborar a cualquier persona independiente de donde esté ubicada en el mundo.

Comparación con el Software de Código Abierto Aunque en la práctica el Software de código abierto y el Software libre comparten muchas de sus licencias, la Free Software Foundation opina que el movimiento del Software de código abierto es filosóficamente diferente del movimiento del Software libre. Los defensores del término «código abierto (Open Source)», afirman que éste evita la ambigüedad del término en ese idioma que es «Free» en «Free Software».

Mucha gente reconoce el beneficio cualitativo del proceso de desarrollo de Software cuando los desarrolladores pueden usar, modificar y redistribuir el código fuente de un programa. El movimiento del Software libre hace especial énfasis en los aspectos morales o éticos del Software, viendo la excelencia técnica como un producto secundario de su estándar ético. El movimiento de código abierto ve la excelencia técnica como el objetivo prioritario, siendo

el compartir el código fuente un medio para dicho fin. Por dicho motivo, la FSF se distancia tanto del movimiento de código abierto como del término «código abierto (Open Source)».

Puesto que la «iniciativa de Software libre Open Source Initiative (OSI)» sólo aprueba las licencias que se ajustan a la «definición de código abierto (Open Source Definition)», la mayoría de la gente lo interpreta como un esquema de distribución, e intercambia libremente "código abierto" con "Software libre". Aún cuando existen importantes diferencias filosóficas entre ambos términos, especialmente en términos de las motivaciones para el desarrollo y el uso de tal Software, raramente suelen tener impacto en el proceso de colaboración.

Aunque el término "código abierto" elimina la ambigüedad de libertad frente a precio (en el caso del inglés), introduce una nueva: entre los programas que se ajustan a la definición de código abierto, que dan a los usuarios la libertad de mejorarlos y los programas que simplemente tienen el código fuente disponible, posiblemente con fuertes restricciones sobre el uso de dicho código fuente. Mucha gente cree que cualquier Software que tenga el código fuente disponible es de código abierto, puesto que lo pueden manipular, sin embargo, mucho de este Software no da a sus usuarios la libertad de distribuir sus modificaciones, restringe el uso comercial, o en general restringe los derechos de los usuarios.

5.4.1 Licencias Creative Commons

Las Licencias[15] Creative Commons (CC) de forma general no tienen una definición oficial, sin embargo, entre las muchas definiciones aceptadas están la de la UNESCO, la cual expresa la siguiente descripción:

Las Licencias Creative Commons (CC) son modelos de contratos que sirven para otorgar públicamente el derecho de utilizar una publicación protegida por los derechos de autor. Entre menos restricciones implique una licencia, mayores serán las posibilidades de utilizar y distribuir un contenido. Las Licencias CC permiten a cualquier usuario descargar, copiar, distribuir, traducir, reutilizar, adaptar y desarrollar su contenido sin costo alguno.

Sin embargo, en la Web oficial de la Organización Creative Commons se nos dice sobre las mismas lo siguiente:

[15] Las licencias de Creative Commons son más utilizadas para cualquier creación digital que para el Software, entendiendo como creación digital desde fotos, artículos en blogs, música, vídeos, este trabajo, etc.

Las Licencias Creative Commons (CC) brindan a todos, desde creadores individuales hasta grandes instituciones, una forma estandarizada de otorgar permiso al público para usar su trabajo creativo bajo la ley de derechos de autor. Desde la perspectiva del reutilizador, la presencia de una licencia Creative Commons sobre una obra protegida por derechos de autor responde a la pregunta: ¿Qué puedo hacer con esta obra?.

Las «Licencias Creative Commons» que hoy en día pertenecen a la organización mundial Creative Commons[16], y buscan regularizar y mantener, de forma equilibrada y satisfactoria, todo lo relacionado con el derecho de utilizar una publicación protegida por los derechos de autor a nivel mundial, han logrado un buen trabajo, sin duda alguna. Y seguramente en el tiempo, se irán adaptando a las nuevas realidades sociales y tecnológicas para poder seguir manteniendo de forma armónica las posibilidades de utilizar y distribuir cualquier contenido libre y abierto sobre la Internet, y más allá.

¿Cuáles son y cómo funcionan o para qué se usan? Las 7 distintas Licencias Creative Commons son las siguientes:

CC BY Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, siempre que se otorgue la atribución al creador. La licencia permite el uso comercial.

CC BY-SA Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, siempre que se otorgue la atribución al creador. La licencia permite el uso comercial. Si remezcla, adapta o construye sobre el material, debe licenciar el material modificado bajo términos idénticos.

CC BY-NC Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, únicamente con fines no comerciales y siempre que se otorgue la atribución al creador.

[16] Una organización mundial sin ánimo de lucro que permite compartir y reutilizar la creatividad y el conocimiento mediante el suministro de herramientas legales gratuitas. Y cuyas herramientas legales (licencias) ayudan a quienes quieren fomentar la reutilización de sus obras ofreciéndolas para su uso bajo términos generosos y estandarizados; a quienes quieren hacer usos creativos de las obras; y a quienes quieren beneficiarse de esta simbiosis.

CC BY-NC-SA Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, únicamente con fines no comerciales y siempre que se otorgue la atribución al creador. Si remezcla, adapta o construye sobre el material, debe licenciar el material modificado bajo términos idénticos.

CC BY-ND Esta licencia permite a los reutilizadores copiar y distribuir el material en cualquier medio o formato únicamente en forma no adaptada, y siempre y cuando se otorgue la atribución al creador. La licencia permite el uso comercial.

CC BY-NC-ND Esta licencia permite a los reutilizadores copiar y distribuir el material en cualquier medio o formato únicamente en forma no adaptada, únicamente con fines no comerciales y siempre que se otorgue la atribución al creador.

CC0 (CC Cero) Esta licencia es una herramienta de dedicación pública que permite a los creadores renunciar a sus derechos de autor y poner sus obras en el dominio público mundial. CC0 permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, sin condiciones.

5.4.2 Nuevas Licencias para Responder a Nuevas Necesidades

El mundo de la tecnología avanza mucho más rápido que las leyes y estas tienen que esforzarse para alcanzarlo. En el caso del Software libre y de código abierto, tanto la Free Software Foundation como la Open Source Initiative (los organismos encargados de regular las diferentes licencias) enfrentan periódicamente el problema de cómo mantener sus principios y al mismo tiempo evitar que alguien se aproveche indebidamente.

En el último tiempo, la Open Source Initiative le dio el sello de aprobación a otras nuevas licencias para propósitos específicos.

Nuevas Licencias de Código Abierto

- Cryptographic Autonomy License version 1.0 (CAL-1.0)

Fue creada en el 2019 por el equipo del proyecto de código abierto Holochain, esta licencia fue desarrollada para ser utilizada con aplicaciones criptográficas distribuidas. El inconveniente con las licencias tradicionales es que no obligaba a compartir los datos. Esto podría perjudicar el funcionamiento de toda la red. Por eso la CAL también incluye la obligación de proporcionar a terceros los permisos y materiales necesarios para utilizar y modificar el Software de forma independiente sin que ese tercero tenga una pérdida de datos o capacidad.

- Open Hardware Licence (OHL)

De la mano de la Organización Europea para la Investigación Nuclear (CERN) llegó esta licencia con tres variantes enfocadas en la posibilidad de compartir libremente tanto Hardware como Software.

Hay que hacer una aclaración. La OSI fue creada en principio pensando en el Software por lo que no tiene mecanismos para la aprobación de licencias de Hardware. Pero, como la propuesta del CERN se refiere a ambos rubros, esto posibilitó la aprobación:

- CERN-OHL-S es una licencia fuertemente recíproca: El que utilice un diseño bajo esta licencia deberá poner a disposición las fuentes de sus modificaciones y agregados bajo la misma licencia.
- CERN-OHL-W es una licencia débilmente recíproca: Sólo obliga a distribuir las fuentes de la parte del diseño que fue puesta originalmente bajo ella. No así los agregados y modificaciones.
- CERN-OHL-P es una licencia permisiva: Permite a la gente tomar un proyecto, relicenciarlo y utilizarlo sin ninguna obligación de distribuir las fuentes.

Hay que decir que la gente del CERN parece haber encontrado la solución a un problema que viene afectando a algunos proyectos de código abierto. Una gran empresa utiliza ese proyecto para comercializar servicios y no solo hace ningún aporte al proyecto original (ya sea con código o dando apoyo financiero) si no que también compite en el mismo mercado.

Post Open Zero-Cost en el año 2024 se desarrolla la licencia «Post Open Zero-Cost» con la cual busca abordar los desafíos surgidos en la interacción entre desarrolladores de código abierto y empresas comerciales, especialmente en lo que respecta a la compensación justa por el uso comercial del código.

La característica distintiva de la licencia «Post-Open» en comparación con las licencias abiertas existentes, como la GPL, es la introducción de un componente contractual que puede ser rescindido en caso de violación de los términos. Esta licencia ofrece dos tipos de acuerdos contractuales: gratuitos y de pago. El acuerdo de pago permite negociar derechos adicionales para la distribución comercial de productos o modificaciones sin requerir su divulgación pública.

Las situaciones que podrían llevar a la terminación del acuerdo contractual incluyen: violación de los términos de la licencia; reclamaciones por infracción de patentes; imposición de condiciones adicionales (como sanciones en contratos con clientes por divulgación de información sensible); cambios sujetos a leyes de control de exportaciones; ocultamiento de información sobre vulnerabilidades; y uso del código para entrenamiento de modelos de aprendizaje automático bajo términos no permitidos. Las relaciones contractuales no se rescinden de inmediato, sino que se notifica la infracción y se otorgan 60 días para corregirla antes de la rescisión efectiva del acuerdo.

Uno de los problemas que la nueva licencia busca abordar está relacionado con las limitaciones de la GPL, la cual se centra en otorgar derechos sin la capacidad de revocarlos, lo que permite a las empresas encontrar formas de eludir sus requisitos, especialmente en lo que respecta al acceso al código fuente. Estas lagunas son utilizadas para restringir la disponibilidad del código subyacente en productos comerciales mediante la imposición de términos contractuales adicionales con los usuarios finales.

Un claro ejemplo es el de RHEL, la cual los clientes firman un acuerdo con Red Hat que limita la redistribución del código fuente al imponer condiciones sobre la coincidencia de las copias instaladas y compradas de RHEL. Esto coloca a los usuarios en la disyuntiva entre su libertad para disponer del software y mantener su estatus de cliente de Red Hat. Aunque la GPL permite la distribución de parches que solucionan vulnerabilidades en el código de RHEL, esto podría interpretarse como una violación del acuerdo con Red Hat y podría resultar en la terminación de los servicios por parte de la empresa.

5.5 Implicaciones Económico-Políticas del Software Libre

Una vez que un producto de Software libre ha empezado a circular, rápidamente está disponible a un costo muy bajo. Al mismo tiempo, su utilidad no decrece. El Software, en general, podría ser considerado un bien de uso inagotable, tomando en cuenta que su costo marginal es pequeñísimo y que no es un bien sujeto a rivalidad -la posesión del bien por un agente económico no impide que otro lo posea-.

5.5.1 Software Libre y la Piratería

Puesto que el Software libre permite el libre uso, modificación y redistribución, a menudo encuentra un hogar entre usuarios para los cuales el coste del Software no libre es a veces prohibitivo, o como alternativa a la piratería. También es sencillo modificarlo localmente, lo que permite que sean posibles los esfuerzos de traducción a idiomas que no son necesariamente rentables comercialmente, además:

- Porque así no se condiciona a los usuarios a usar siempre lo mismo.
- Porque así no se fomenta la piratería en los usuarios al no pagar licencias.
- Porque así no se obliga a usar una solución concreta y se ofrece libertad de elección a los usuarios.
- Porque es mucho más seguro ya que el Software libre es público y se puede ver qué hace exactamente sin recelos.

La mayoría del Software libre se produce por equipos internacionales que cooperan a través de la libre asociación. Los equipos están típicamente compuestos por individuos con una amplia variedad de motivaciones y pueden provenir tanto del sector privado, del sector voluntario o del sector público. En los últimos años se ha visto un incremento notable de grandes corporativos (como IBM, Microsoft, Intel, Google, Samsung, Red Hat, etc.) que han dedicado una creciente cantidad de recursos humanos y computacionales para desarrollar Software libre, ya que esto apoya a sus propios negocios.

5.5.2 ¿Cuánto Cuesta el Software Libre?

En esta sección intentaremos dar una idea de cuál es el costo del desarrollo del Software Libre, por supuesto que no se tratará más que de una conjetura aproximada basada en las cifras proporcionadas por desarrolladores de Software comercial (al año 2020).

Gratis no Significa Gratuito Supongamos que todos los recursos humanos participantes en el desarrollo de un proyecto de Software libre lo hagan de forma voluntaria. De todas formas tenemos lo que los contables llaman «Costo de oportunidad» esto es, los ingresos que podrían haber generado esas personas si hubieran dedicado el tiempo y los conocimientos invertidos en el proyecto a uno en el que les pagaran. Así, el calcular el costo promedio por hora que cobra un programador, por la cantidad de horas invertidas al proyecto, nos da un razonable costo mínimo. Lo mismo puede hacerse con los voluntarios dedicados a la difusión en las redes. El costo de una campaña de marketing digital puede estimarse fácilmente.

Muchos proyectos de código abierto como una distribución Linux, son construidos a partir de la integración de otros proyectos, por los que sus costos de desarrollo también deberían sumarse.

Por otra parte, necesitamos recursos físicos. Aún cuando los voluntarios trabajen desde su casa, siguen teniendo que comprar y mantener sus equipos, además de pagar la electricidad que los hace funcionar.

Bases para el Cálculo Hay muchos factores que determinan el costo de desarrollar una pieza de Software. En un extremo tenemos una aplicación simple que requiere muy poca interacción del usuario o procesamiento del lado del servidor. Tal es el caso de un cliente de escritorio para redes sociales. Por el otro sistemas operativos que deben operar en múltiples plataformas realizando múltiples tareas (por ejemplo Debían que aspira a ser el sistema operativo universal). Sin embargo, el costo de una aplicación simple puede elevarse debido a que tiene múltiples pantallas diferentes. Por ejemplo un juego desarrollado con HTML5 y Javascript.

Los dos aspectos claves son la cantidad de horas de trabajo necesarias y las tecnologías involucradas. Para una aplicación de escritorio como un procesador de textos con las prestaciones habituales, optimizado para un determinado escritorio Linux, se estima que se tendría que contar con al menos el equivalente a 42,000 euros en trabajo voluntario. Un gestor de contenidos

para comercio electrónico con seguimiento de pedidos e integración con las principales plataformas de pago implicaría desembolsar unos 210,000 euros o su equivalente en trabajo voluntario.

Tomando en cuenta que este cálculo incluye lo que costó el desarrollo de las bibliotecas y otros proyectos libres y de código abierto incluidos, pero no los gastos que efectivamente deben desembolsarse en efectivo como la compra de equipos, Software de seguridad y desarrollo y el pago de electricidad e internet.

Por otro lado, el proceso de medición de costes del Software es un factor realmente importante en el análisis de un proyecto. Hay distintos métodos de estimación de costes de desarrollo de Software (también conocido como métrica del Software). La gran mayoría de estos métodos se basan en la medición del número de Líneas de Código que contiene el desarrollo (se excluyen comentarios y líneas en blanco de los fuentes).

Desarrollo de Fedora 9 La Linux Foundation ha calculado que costaría desarrollar el código de la distribución Fedora 9 que fue puesta a disposición del público el 13 de mayo de 2008, en el informe citado "Estimating the Total Development Cost of a Linux Distribution" se calcula que Fedora 9 tiene un valor de 10.8 mil millones de dólares y que el coste únicamente del Kernel (2.6.25 con 8,396,250 líneas de código) tendría un valor de 1.4 mil millones de dólares.

Esta distribución tiene unas 205 millones de líneas de código, el proyecto debería ser desarrollado por 1000 - 5000 desarrolladores (el trabajo invertido por una única persona desarrollándolo se alargaría durante unos 60.000 años) y esa estimación no va muy desencaminada ya que en los 2 últimos años del desarrollo de esa versión contribuyeron unos 3,200 desarrolladores aunque el número de trabajadores en la historia de la distribución es mucho mayor.

¿Qué pasa con GNU/Linux? en el año 2015 (las estadísticas más actuales que conseguimos) la Linux Foundation analizó el costo de desarrollo del núcleo. Combinando el aporte de los recursos humanos (voluntarios y de pago) y los desembolsos necesarios, la cuenta sumó 476,767,860,000.13 euros.

Todos sabemos que el hecho de tener desarrolladores asalariados no garantiza necesariamente Software de calidad. Pero, tener desarrolladores que pueden dedicar toda su atención a un proyecto en lugar de hacerlo en sus horas libres si lo hace. Lamentablemente, por el momento el único modo de

lograr eso es obtener el apoyo de corporaciones (Intel, Google, IBM, AMD, Sun Microsystems, Dell, Lenovo, Asus, HP, SGI, Oracle, RedHat, etc.) que solo lo hacen con los proyectos que son de su interés como el Kernel de Linux, hay que notar que para el Kernel de Linux un porcentaje importante (más del 10 %) lo hacen programadores independientes.

Costes Recordemos que la segunda de las cuatro libertades de un programa para ser Software libre es:

- Libre redistribución

y esta puede ser a través de un pago o sin costo. Es por ello que existen distintas empresas, organizaciones y usuarios que pueden apoyar a los usuarios finales en el desarrollo y soporte de algún programa de Software libre o una distribución personalizada de Linux por un costo determinado.

Mucha gente, en especial ejecutivos de empresas, se acercan a Linux bajo la promesa de que es una solución de bajo costo -muchos piensan que incluso es gratis-. Pero la realidad es que detrás de Linux y los programas de Software libre (y aquí la traducción correcta de la palabra inglesa, Free es libre, no gratis) pueden llevar una serie de costos ocultos que deben ser considerados al momento de decidir si se implementa una solución propietaria o una libre.

Los costos ocultos aparecen cuando se intenta instalar y capacitar en el uso de algún Software y se necesita la ayuda de un informático, al que se tiene que pagar, o alguna empresa quiere personalizar la interfaz de un programa y necesita la ayuda de un programador, que también tienen un coste, por lo que finalmente el comentario suele ser "el Software libre no es barato".

El primer punto a considerar al evaluar ambas alternativas es el costo de la licencia. Los productos de Software libre no suelen tener un costo de licencia asociado, que sí existe en los programas propietarios. De hecho es allí en donde los fabricantes de Software recargan sus costos de investigación y desarrollo, de producción e incluso sus ganancias. En este primer punto el ganador claro es el Software libre y es lo que los adeptos de este esquema publicitan: "su compañía puede ahorrar miles de dólares al año usando Software libre".

El segundo punto a considerar es el costo de instalación, configuración y capacitación. Dependiendo de su complejidad, algunos productos comerciales no contemplan costos extra por este concepto y otros -como Windows, por ejemplo- son tan populares que se puede encontrar numerosas opciones

de instalación -a través de empresas o profesionales- donde escoger en el mercado. A veces en el Software libre la configuración puede implicar recompilar el producto con algunas opciones particulares, algo que sólo pueden realizar técnicos con un nivel adecuado de conocimientos y que puede que no sean tan fáciles de encontrar. Aquí por lo general la ventaja va hacia el Software comercial.

Una vez instalado el Software, toca realizar actualizaciones de mantenimiento. Si bien es cierto lo que algunos de los fanáticos del Software libre dicen, que nadie lo obliga a mejorar el Software con que cuenta, la realidad -especialmente en lo que a seguridad se refiere- obliga a las empresas a mantener su Software actualizado. Aún así, los costos de actualizar Software libre suelen ser significativamente más bajos que los de productos comerciales y suelen ser menos exigentes con el Hardware necesario para ejecutarlos. La mayoría de las veces la ventaja es para el Software libre, pero hay que evaluar ya que varía dependiendo de cada caso.

Por último hay algunas casas que desarrollan productos de Software libre -dan el código y permiten que cualquiera lo modifique o reutilice- pero fijan contratos con cargos mensuales o anuales para mantenimiento del mismo, algo que se parece mucho a un cobro por licencia, por lo que hay que estar seguro de conocer todas las condiciones cuando una empresa nos ofrece una solución propia y la califica como Software libre.

Además de estas consideraciones hay una que debe sumarse eventualmente a esta evaluación: el costo de migrar a otra solución. En Software libre suelen usarse estándares abiertos para almacenar los datos, lo que facilita las migraciones. En cambio muchas soluciones propietarias suelen tener formatos propietarios que pueden dejar "amarrados" los datos de la empresa a una aplicación específica.

Sólo después de evaluar estos aspectos del Software, que pueden tener implicaciones importantes en el presupuesto, es que un CIO (Chief Information Officer) puede decir si una solución de Software libre le conviene más a una empresa o no, algo que va más allá de que la aplicación sea gratis o no.

5.5.3 La Nube y el Código Abierto

Desde hace años se han creado nuevos desafíos para el código abierto que plantea la nube, un término que para el usuario promedio puede significar cosas diferentes, pero que para la empresa se resume en servicios. Y es que los beneficios económicos que genera el mero Software de código abierto no

son comparables a los que se obtienen cuando se ofrece ese mismo Software a través de servicios, más allá -pero incluyendo- del soporte.

Este hecho diferencial lleva tiempo provocando fricciones entre desarrolladores y proveedores y hay quien adelantó incluso el fin del modelo de desarrollo del código abierto tal y como lo conocemos. ¿Quién tiene la razón?, ¿es para tanto la situación?.

En sus exposiciones, representantes de compañías y proyectos de código abierto muy populares en el ámbito empresarial, explican el supuesto perjuicio que les ocasiona el uso que los grandes proveedores de servicios en la nube hacen del Software que ellos desarrollan y cómo algunos han considerado y aplicado un enfoque más cerrado para sus productos con el fin de evitar lo que denominan como expolio. Hay declaraciones que merecen ser rescatadas para dotar de contexto a la discusión:

- El papel que juega el código abierto en la creación de oportunidades comerciales ha cambiado, durante muchos años les permitimos que las empresas de servicios tomaran lo que se ha desarrollado y ganasen dinero con ello.

- Empresas como Amazon Web Services, Azure de Microsoft, etc. Han ganado cientos de millones de dólares ofreciendo a sus clientes servicios basados en Software libre sin contribuir tanto a la comunidad de código abierto que construye y mantiene ese proyecto. Es imposible saber exactamente de cuánto dinero estamos hablando, pero es cierto que los proveedores de la nube se benefician del trabajo de los desarrolladores de código abierto que no emplean.

- Hay un mito ampliamente instalado en el mundo de código abierto que dice que los proyectos son impulsados por una comunidad de contribuyentes, pero en realidad, los desarrolladores pagados contribuyen con la mayor parte del código en la mayoría de los proyectos de código abierto modernos.

En resumen, todas estas voces se quejan de dos cosas: los grandes beneficios que obtienen los proveedores de servicios en la nube con su Software sin retribuirles en consecuencia, y la falta de colaboración manteniendo los productos con los que lucran. Sin embargo, no nos engañemos, el quid de la cuestión está principalmente en el dinero: la opinión generalizada de la

comunidad es que el Software de código abierto nunca fue pensado para que las empresas de servicios en la nube lo tomasen y lo vendieran.

Por otro lado, si es posible bifurcar un proyecto libre que se cierra, ¿no hubiese sido mejor colaborar con él antes y haber evitado el cierre? Si no se invierte y se mantiene con salud aquello que da beneficios, puede terminar por desaparecer. A medio camino entre el depredador y el parásito: así es como ven muchas desarrolladoras de código abierto a los proveedores de servicios en la nube.

Vale la pena retomar ahora la frase «el Software de código abierto nunca fue pensado para que las empresas de servicios en la nube lo tomasen y lo vendieran». ¿Para qué fue pensado el código abierto entonces? No hay ninguna licencia de código abierto o Software libre reconocida por la Free Software Foundation o la Open Source Initiative que prohíba hacer negocio con el Software. Lo que prohíben es la discriminación en la capacidad y alcance de su uso en función de la parte, se trate de un individuo o de la mayor multinacional imaginable.

¿Cuál es la solución a un embrollo de tamaña envergadura? Lo único claro es que no es una cuestión de blancos y negros y las consideraciones son demasiadas como para seguir ahondando: empresas que cotizan en bolsa quieren más dinero de otras compañías -que también cotizan en bolsa y por mucho más-, que además del Software ponen la infraestructura sobre la que distribuyen sus ofertas y que tienen la capacidad de clonar tu producto en un abrir y cerrar de ojos, no solo porque tienen el capital, sino porque tienen la experiencia necesaria tras contribuir técnica, pero también en muchos casos, económicamente, durante largo tiempo.

Pese a ello, esta situación está alterando el paradigma actual, en el que el modelo de desarrollo del código abierto se ha impuesto como impulsor de la innovación en el sector empresarial y ya hay quien habla de que nos acercamos al fin, o al principio del fin de la Era del Open Source, cuya preponderancia estaría sentenciada por la revolución de la nube, a la postre el mayor estímulo que haya tenido el código abierto hasta la fecha.

El futuro, pues, pasaría por el Shared Source Software, bajo el cual diferentes compañías con intereses alineados colaborarán en el desarrollo de proyectos concretos, pero limitando su explotación comercial a sí mismas. Todavía no estamos ahí, no obstante, y no parece tampoco que el relevo se vaya a dar en breve. De suceder, será muy llamativo: la muerte del código abierto por un éxito mal entendido.

5.5.4 El Código Abierto como Base de la Competitividad

En septiembre del 2021, se publicó un amplio y detallado informe llevado a cabo por el Fraunhofer ISI y por OpenForum Europe para la Comisión Europea, «The impact of open source Software and hardware on technological independence, competitiveness and innovation in the EU economy», cuantifica la importancia económica del código abierto aplicado tanto al Software como al Hardware, su efecto en la contribución al producto interior bruto generado, la reducción en aspectos como el coste total de propiedad, dependencia del proveedor y autonomía digital; lanza una serie de recomendaciones específicas de políticas públicas destinadas a lograr un sector público digitalmente autónomo, una investigación y desarrollo abierto que fomente el crecimiento europeo y una industria más digitalizada y competitiva.

En las estimaciones del informe se apunta que las empresas europeas invirtieron alrededor de mil millones de euros en Software de código abierto en 2018, lo que resultó en un impacto en la economía europea de entre 65,000 y 95,000 millones de euros. El análisis estima una relación costo-beneficio superior a 1:4 y predice que un aumento del 10% de las contribuciones de a repositorios de código abierto sería susceptible de generar anualmente entre un 0.4% y un 0.6% adicional en el PIB, así como más de seiscientas nuevas empresas tecnológicas en la Unión Europea.

El análisis de las contribuciones a repositorios de Software de código abierto en la Unión Europea revela que el ecosistema tiene una naturaleza diferente frente al norteamericano, con un volumen de contribuciones que provienen sobre todo de empleados de compañías pequeñas o muy pequeñas, frente a un escenario en los Estados Unidos en el que predominan grandes compañías tecnológicas que se benefician en sus modelos de negocio de la gran cantidad y de la rápida mejora del Software disponible. En Europa, los contribuyentes individuales ascendieron a más de 260,000, lo que representa el 8% de los casi 3.1 millones de empleados de la UE en el sector del desarrollo de Software en 2018. En total, los más de 30 millones de desarrollos consolidados en repositorios en los estados miembros de la Unión Europea representan una inversión de personal equivalente a casi mil millones de euros, que han pasado a estar disponibles en el dominio público y que, por lo tanto, no tienen que ser desarrollados por otros actores.

Según el análisis, cuanto más pequeña es la empresa, mayor es la inversión relativa en Software de código abierto (las empresas con 50 empleados o menos asumieron casi la mitad de los desarrollos en la muestra de las com-

pañías más activas). Aunque más del 50% de los contribuyentes pertenecen a la industria tecnológica (el 8% del total de sus empleados participaron en estos desarrollos), también hubo participación significativa de empresas de consultoría, científicas, técnicas y, en menor medida, de distribuidores, minoristas y empresas del ámbito financiero.

¿Puede una filosofía de desarrollo como el código abierto, disponible para todo el mundo, llegar a convertirse en una fuente de ventajas diferenciales para el resto de los países, que se ha visto tradicionalmente muy superado en su relevancia en el entorno tecnológico por los gigantes tecnológicos de Estados Unidos o de China? El informe afirma que su uso puede llegar a incidir en gran medida en el desarrollo de una independencia tecnológica superior, de una mayor competitividad y de más innovación. Veremos si llegamos a ver en el resto del mundo políticas que incentiven el uso del código abierto como una variable estratégica clave para ello. La idea, capitalizar la tecnología de una forma más orientada al procomún y al desarrollo colaborativo, suena sin duda atractiva e interesante.

5.5.5 Software Libre en Empresas y Corporaciones

En esta sección exploraremos algunas de las claves por las cuales el Software libre está hoy en el punto de mira de todo tipo de empresas y grandes corporaciones (algunas de las cuales ayer eran sus acérrimos enemigos). Pero hay que empezar destacando que corporativos como Google, Amazon Web Services, Azure de Microsoft, Microsoft, IBM, entre otras, en los últimos años han ganado miles de millones de dólares ofreciendo a sus clientes servicios y/o productos basados en Software libre y con una queja recurrente por su pobre o nula contribución a la comunidad de código abierto que construyó y mantiene esos proyectos.

Si bien es imposible saber exactamente de cuánto dinero estamos hablando, pero es cierto que empresas y corporaciones se benefician diariamente del trabajo de los desarrolladores de código abierto que no emplean.

Grandes Equipos de Programadores GNU/Linux ha demostrado que los equipos de desarrolladores grandes, distribuidos, aunque desorganizados pueden crear Software viable.

Antes de la llegada de GNU/Linux, la mayoría del Software era desarrollado por pequeños equipos de programadores que trabajaban en estrecha coordinación entre sí. Ese era el enfoque recomendado por informáticos de hace

unos decenios, que advertían que añadir más programadores a un proyecto tendía a disminuir su eficiencia. Y estaban muy equivocados.

Desde el principio, el Kernel de Linux se desarrolló con un enfoque diferente, en el que programadores de todo el mundo, que en la mayoría de los casos no se conocían, escribieron e integraron el código de forma rápida y poco organizada. Gracias a la publicación temprana y frecuente, consiguieron que funcionara y hoy día es el Kernel más usado en informática en supercomputadoras y dispositivos móviles.

Pero actualmente hay un mito ampliamente instalado en el mundo de código abierto, que dice que los proyectos son impulsados por una comunidad de contribuyentes gratuitos, pero en realidad, los desarrolladores pagados contribuyen con la mayor parte del código en la mayoría de los proyectos de código abierto modernos de los cuales las corporaciones pueden sacar provecho. Claro ejemplo es el propio Kernel de Linux, en el cual una gran cantidad de desarrolladores actuales pertenecen o son subvencionados por empresas, fundaciones o corporaciones (actualmente cientos de ellas), como en el caso de Linus Torvalds que trabaja bajo los auspicios de la Fundación de Linux.

Reutilización de Software Parte de la razón por la que Linux se hizo muy popular entre los ingenieros de Software con relativa rapidez fue que Linux -y el Software libre en general- facilita la reutilización del código escrito por otras personas.

Hoy en día, la reutilización de Software de terceros es habitual, incluso entre los equipos de desarrollo cuyos productos no son de código libre. Es difícil imaginar la construcción de una aplicación hoy en día sin hacer uso de las bibliotecas de Software de origen, las API de terceros u otros recursos externos a su propio proyecto.

Es cierto que proyectos como GNU, que precedió al Kernel de Linux en siete años, promovían la reutilización de código antes de que apareciera el núcleo. Pero, podría decirse que Linux fue el proyecto que trajo las prácticas de codificación libre a la corriente que tanto parece interesar a ciertas grandes empresas, ayudando a crear el modelo de ingeniería de Software de componentes de Software modulares y reutilizables.

Gestión Actual del Código Fuente Linus Torvalds, que creó el núcleo de Linux cuando era estudiante en Helsinki, es el más famoso por ese trabajo.

Pero un hecho a menudo olvidado es que Torvalds es también el padre de Git, el masivamente popular gestor de código fuente libre.

Torvalds creó Git para ayudar a gestionar el código fuente de Linux. Si Linux no existiera, tampoco existiría Git. Tampoco existiría GitHub, ni GitLab, ni GitOps. Y, lo que es más importante, sin la idea de Software libre y colaborativo de Richard Stallman, tampoco existiría la cultura de intercambio y colaboración abierta que sostienen estas tecnologías.

Estrategias de Despliegue de Software "App Store" Apple puede atribuirse el mérito de haber lanzado la primera App Store, un lugar donde los desarrolladores pueden compartir aplicaciones y los usuarios pueden instalarlas fácilmente, utilizando un catálogo Online centralizado.

Pero al igual que con muchas cosas que ha hecho Apple, el concepto de App Store (que ahora es una estrategia de despliegue de Software apilado como servicio, especialmente pero no sólo en el ecosistema móvil) se parece mucho a lo que los desarrolladores de GNU/Linux estaban haciendo a través de los repositorios de Software mucho antes de que las tiendas de aplicaciones se convirtieran en algo común en el mundo del Software propietario, como también lo es la Tienda de Windows.

Los repositorios de Software en GNU/Linux hacen más o menos lo mismo que las tiendas de aplicaciones: Permiten a los usuarios seleccionar las aplicaciones que quieren de una lista centralizada y en línea, y luego instalarlas con unos pocos clics o bien órdenes de terminal (como el famoso *apt* de Debian).

Es cierto que empresas como Apple parecen tener el mérito de crear tiendas de aplicaciones muy fáciles de usar de hacer clic e ir, pero no es un invento de ellos. Y la historia del concepto de tienda de aplicaciones en general implica a más actores que sólo la comunidad de Apple. Aún así, creo que se podría argumentar con fuerza que, sin GNU/Linux y los repositorios de Software de GNU/Linux, las tiendas de aplicaciones tal y como las conocemos hoy no existirían.

Formatos Abiertos para Intercambio de Información Hay una gran variedad de tecnologías disponibles para producir y almacenar datos. Como son: hojas de cálculo, bases de datos, Software estadístico más específico y más. Esto genera una enorme diversidad de formatos, a veces esto es por decir lo menos caótico.

La ventaja de los archivos de formatos abiertos, es que permiten a los de-

sarrolladores producir varios paquetes de Software y servicios utilizando esos formatos. Esto entonces reduce al mínimo los obstáculos para la reutilización de la información que contienen.

El advenimiento del Software libre ha generado algunos de los formatos abiertos más usados para el intercambio de información, pero los entes generadores de información no siempre se adecuan a los niveles de apertura deseados, algunos de estos formatos abiertos son: XML, JSON, YAML, RDF, REBOL, PDF, CSV, ODF, OOXML, TXT, HTML, HDF.

Pero, incluso si la información se proporciona en formato electrónico, formato legible por máquina y en detalle, puede existir problemas relacionados con el formato del archivo en sí (principalmente el generado por los diversos sistemas operativos). Los formatos en los cuales la información es publicada -en otras palabras, la base digital en la cual la información es almacenada- puede ser "abierta" o "cerrada".

Un formato abierto es aquel donde las especificaciones del Software están disponibles para cualquier persona, de forma gratuita, así cualquiera puede usar dichas especificaciones en su propio Software sin ninguna limitación en su reutilización que fuere impuesta por derechos de propiedad intelectual.

Si el formato del archivo es "cerrado", esto puede ser debido a que el formato es propietario y sus especificaciones no están disponibles públicamente, o porque el formato es propietario y aunque las especificaciones se han hecho públicas, su reutilización es limitada. Si la información es liberada en un formato de archivo cerrado, esto puede causar grandes obstáculos para reutilizar la información codificada en él, forzando a aquellos que deseen usar la información a comprar Software innecesario.

El uso de formatos de archivo con propiedad, para el que la especificación no está disponible públicamente, puede crear dependencia de Software de terceros o de los titulares de licencias de los formatos de archivos. En el peor de los casos, esto puede significar que la información sólo se puede leer con cierto Software específico, que puede ser caro, o que puede quedar obsoleto.

La preferencia del término Gobierno de Datos Abiertos, es que la información se publicará en formatos de archivo abiertos, los cuales son de lectura mecánica y esto es una aportación más del Software libre.

Ciencia Abierta La ciencia abierta (Open Science) es el movimiento creciente para hacer que la ciencia sea abierta. La ciencia en sí misma se utilizó como un ejemplo principal de la eficacia del movimiento de código abierto, ci-

tando prácticas como la difusión abierta de información, métodos y revisión por pares de la literatura científica. Podría decirse que la ciencia abierta comenzó en el siglo XVII con el advenimiento de la revista científica y la práctica de repetir los experimentos presentados en los artículos académicos. Estas revistas se imprimirían y distribuirían en todo el mundo, a menudo supervisadas por sociedades científicas como la Royal Society.

¿Qué impulsó la necesidad de un movimiento de ciencia abierta? La Royal Society tenía el famoso lema "Nullius in verba", traducido de forma aproximada como "no tome la palabra de nadie". Esto encarnaba un principio general en la ciencia de que todas las teorías están abiertas a ser cuestionadas y los resultados declarados deben ser repetibles. De hecho, es una práctica generalizada que fue realizada por la sociedad en esos primeros años. En los últimos años esta práctica no ha sido tan común, con más y más ciencia confiando en elementos cerrados, lo que en última instancia conduce a errores que son más difíciles de detectar sin un intercambio completo de información: datos, métodos y publicaciones.

El movimiento de ciencia abierta afirma en términos generales que la ciencia debe realizarse de manera abierta y reproducible donde todos los componentes de la investigación estén abiertos. Muchas revistas permanecen estancadas en un formato en el que se imprimían físicamente, a pesar de que en la actualidad se distribuyen en gran medida en línea. A menudo, todavía utilizan archivos PDF como una forma de "papel electrónico" con publicaciones fijas, procesos cerrados de revisión por pares y poco o ningún acceso a los datos. Este fue sin duda el modo más eficiente de difundir el conocimiento científico antes de los albores de internet, pero ahora un número cada vez mayor lo considera lejos de ser el óptimo.

La ciencia abierta encarna una serie de aspectos, en el núcleo esto incluye acceso abierto, datos abiertos, código abierto y estándares abiertos que ofrecen una diseminación sin restricciones del discurso científico. Estas cosas permiten una ciencia reproducible al brindar acceso completo a los componentes principales de la investigación científica. Hay una serie de componentes adicionales que también se están explorando, como la revisión por pares abierta, donde los revisores de publicaciones científicas publican revisiones abiertamente con su nombre adjunto y la ciencia de libreta abierta donde las libretas (tradicionalmente cerradas) se publican abiertamente en línea a medida que se realiza la investigación.

¿Por qué la ciencia abierta es tan importante en la era digital? También existe una creciente comprensión de que, dado que la investigación científica

depende cada vez más del código informático para simulaciones, cálculos, análisis, visualización y procesamiento de datos en general, es importante tener acceso a este código tal como tradicionalmente ha sido importante mostrar (y derivar) cualquier nueva técnica matemática introducida para el análisis. Hay revistas como PLOS ONE y F1000 que exploran el significado de las publicaciones, ya sea que se deben congelar en el tiempo o se pueden actualizar. Los repositorios de datos también están ganando importancia a medida que las agencias de financiación requieren la publicación y preservación de los datos generados por la investigación financiada.

En esencia, la ciencia abierta se trata de volver a esos valores fundamentales inculcados por algunos de los primeros científicos de que no debemos confiar en la palabra de nadie, que es esencial que todos los elementos pertinentes a un descubrimiento pretendido se publiquen para que los resultados puedan repetirse y validarse. El movimiento de la ciencia abierta varía en el grado en que lo requiere, pero están surgiendo patrones. Se están estableciendo recomendaciones sobre licencias, como CC0 para datos, CC-BY para publicaciones, licencias compatibles con OSI para código fuente y formatos abiertos para datos. En última instancia, se trata de empoderar a todos para que participen en la ciencia, con internet como vehículo principal para la amplia difusión de este conocimiento.

Este movimiento está cambiando la forma en que se hace la ciencia, está recibiendo el respaldo de muchas agencias de financiamiento, ya que requieren planes de gestión de datos, planes de distribución de código fuente y una mayor validación de los resultados a través del acceso abierto a estos resultados para todos. Esto también mejora la transferencia de conocimientos de la academia a la industria, ya que se brinda acceso completo en el momento de la publicación o después de un período de embargo. El movimiento de la ciencia abierta se limita en gran medida a la investigación que está financiada por las agencias de financiación nacionales de todo el mundo y exige que todos los que financian la investigación tengan acceso total e igualitario a ella.

Open Hardware El concepto de Software libre también se permeó al Hardware. El término Open Hardware u Open Source Hardware, se refiere al Hardware cuyo diseño se hace públicamente disponible para que cualquiera pueda estudiarlo, modificarlo y distribuirlo, además de poder producir y vender Hardware basado en ese diseño. Tanto el Hardware como el Software

que lo habilita, siguen la filosofía del Software libre. Hoy en día, el término "hágalo usted mismo" (DIY por sus siglas en inglés) se está popularizando en el Hardware gracias a proyectos como Arduino que es una fuente abierta de prototipos electrónicos, una plataforma basada en Hardware flexible y fácil de utilizar que nació en Italia en el año 2005.

El movimiento de Hardware abierto o libre, busca crear una gran librería accesible para todo el mundo, lo que ayudaría a las compañías a reducir en millones de dólares en trabajos de diseño redundantes. Ya que es más fácil tener una lluvia de ideas propuesta por miles o millones de personas, que por solo una compañía propietaria del Hardware, tratando así de que la gente interesada entienda cómo funciona un dispositivo electrónico, pueda fabricarlo, programarlo y poner en práctica esas ideas en alianza con las empresas fa-bricantes, además se reduciría considerablemente la obsolescencia programada y en consecuencia evitaríamos tanta basura electrónica que contamina el medio ambiente. Al hablar de Open Hardware hay que especificar de qué tipo de Hardware se está hablando, ya que está clasificado en dos tipos:

- Hardware estático. Se refiere al conjunto de elementos materiales de los sistemas electrónicos (tarjetas de circuito impreso, resistencias, capacitores, LEDs, sensores, etcétera).

- Hardware reconfigurable. Es aquél que es descrito mediante un HDL (Hardware Description Language). Se desarrolla de manera similar a como se hace Software. Los diseños son archivos de texto que contienen el código fuente.

Para tener Hardware reconfigurable debemos usar algún lenguaje de programación con licencia GPL (General Public License). La licencia GPL, al ser un documento que cede ciertos derechos al usuario, asume la forma de un contrato, por lo que usualmente se le denomina contrato de licencia o acuerdo de licencia. La Organización Europea para la investigación Nuclear (CERN) publicó el 8 de julio de 2011 la versión 1.1 de la Licencia de Hardware Abierto.

Existen programas para diseñar circuitos electrónicos y aprender de la electrónica como EDA (Electronic Design Automation) y GEDA (GPL Electronic Design Automation), son aplicaciones de Software libre que permiten poner en práctica las ideas basadas en electrónica.

Es posible realizar el ciclo completo de diseño de Hardware reconfigurable desde una máquina con GNU/Linux, realizándose la compilación, simulación,

síntesis y descarga en una FPGA (Field Programmable Gate Arrays). Para la compilación y simulación se puede usar GHDL (https://ghdl.free.fr) junto con GTKWave (https://gtkwave.sourceforge.net) y para la síntesis el entorno ISE de Xilinx. Este último es Software comercial pero existe una versión gratuita con algunas restricciones.

Sabemos que tanto en el caso del Software como el Hardware, libre no es lo mismo que gratis. Específicamente, en el caso del Hardware, como estamos hablando de componentes físicos que son fabricados, la adquisición de componentes electrónicos puede ser costosa. Aun así, es un campo que no solo es apasionante sino que también tiene mucho futuro y representa grandes oportunidades.

Entusiasmo de la Comunidad Por último, pero no menos importante, probablemente el mayor impacto duradero de GNU/Linux en el modelo de ingeniería de Software se reduce a lo que podría llamarse entusiasmo de la comunidad. Me refiero a la forma en que GNU/Linux en particular, y el Software libre en general, ha animado a los desarrolladores de todo tipo a considerar las contribuciones a la comunidad como uno de sus objetivos finales y esto ahora es notorio no solo en Software, sino en Hardware abierto, obras literarias, escritos técnicos (como este trabajo), imágenes, vídeo, música y un largo etc.

En un mundo de código libre en el que las contribuciones a los proyectos de código pueden ser aceleradores de carrera y el código de licencia libre se reutiliza ampliamente, los desarrolladores entienden que hay un valor real en la construcción de Software que puede beneficiar a tantos usuarios como sea posible.

Tal vez los desarrolladores valorarían a la comunidad en su conjunto si GNU/Linux y el código libre nunca hubieran aparecido. Pero me cuesta imaginar un mundo en el que corporaciones como Microsoft y Google trabajarán juntos en la construcción de Software para GNU/Linux si GNU/Linux no hubiera popularizado el concepto de proyectos de Software impulsados por la comunidad que nadie posee realmente, pero que todos pueden utilizar.

Si bien, es innegable que todo lo anterior puso en la mira de las empresas de todos los tamaños y de las grandes corporaciones el Software libre, la principal razón es el poder utilizar una gran cantidad Software funcional, depurado y ampliamente usado para ofrecer servicios y/o productos basados en Software libre y así beneficiarse económicamente de ello.

Por otro lado, se ha visto a través de múltiples estudios, el impacto y la cuantificación de la importancia económica del código abierto aplicado tanto al Software como al Hardware, su efecto en la contribución al producto interior bruto generado, la reducción en aspectos como el coste total de propiedad, dependencia del proveedor y autonomía digital. Además de generar políticas públicas destinadas a lograr un sector público digitalmente autónomo, una investigación y desarrollo abierto que fomente el crecimiento de los países y una industria más digitalizada y competitiva.

Retomando la frase «el Software de código abierto nunca fue pensado para que las empresas de servicios lo tomasen y lo vendieran». ¿Para qué fue pensado el código abierto, entonces? No hay ninguna licencia de código abierto o Software libre reconocida por la Free Software Foundation o la Open Source Initiative que prohíba hacer negocio con el Software. Lo que prohíben es la discriminación en la capacidad y alcance de su uso en función de la parte, se trate de un individuo o de la mayor multinacional imaginable.

Pese a ello, esta situación está alterando el paradigma actual, en el que el modelo de desarrollo del código abierto se ha impuesto como impulsor de la innovación en el sector empresarial, a la postre el mayor estímulo que haya tenido el código abierto hasta la fecha.

¿Puede una filosofía de desarrollo como el código abierto, disponible para todo el mundo, llegar a convertirse en una fuente de ventajas diferenciales para el resto de los países, que se ha visto tradicionalmente muy superado en su relevancia en el entorno tecnológico por los gigantes tecnológicos de Estados Unidos o de China? Se afirma que su uso puede llegar a incidir en gran medida en el desarrollo de una independencia tecnológica superior, de una mayor competitividad y de más innovación. La idea, capitalizar la tecnología de una forma más orientada al procomún y al desarrollo colaborativo, suena sin duda atractiva e interesante pero no libre de inconvenientes para algunos sectores de desarrolladores de Software libre.

5.6 Código Abierto y las Organizaciones Internacionales

Aunque la Organización de las Naciones Unidas (ONU) ha hablado previamente bien del desarrollo del código abierto, varios eventos recientes muestran que la ONU está tomando medidas definitivas para presentar al mundo entero el camino del código abierto. En julio del 2021, el Consejo Económico y Social de la ONU (ECOSOC) adoptó un proyecto de resolución presentado por el representante de Pakistán titulado: Tecnologías de fuente abierta para

el desarrollo sostenible.

5.6.1 Las Naciones Unidas y el Código Abierto

El ECOSOC destacó la disponibilidad de tecnologías de código abierto que pueden contribuir a los Objetivos de Desarrollo Sostenible (ODS). El consejo invitó al Secretario General a "desarrollar propuestas específicas sobre formas de aprovechar mejor las tecnologías de código abierto para el desarrollo sostenible basadas en las aportaciones de los Estados Miembros interesados y otras partes interesadas".

El desarrollo de tecnología de código abierto puede ser una herramienta rápida y eficaz para la innovación. Aplicarlo a tecnologías apropiadas para ayudar a alcanzar los ODS es extremadamente prometedor. Las "tecnologías apropiadas" abarcan opciones y aplicaciones tecnológicas que son a pequeña escala, económicamente asequibles, descentralizadas, energéticamente eficientes, ambientalmente racionales y fácilmente utilizadas por las comunidades locales para satisfacer sus necesidades.

Existe un caso particularmente fuerte para las tecnologías apropiadas de código abierto OSAT (Outsourced Semiconductor Assembly and Test). OSAT podría ayudar a todos a salir de la pobreza y alcanzar un estado sostenible aprovechando el mismo tipo de desarrollo que hace que el Software de código abierto sea un éxito rotundo.

La Declaración Ministerial del Foro político de alto nivel sobre desarrollo sostenible también destacó la importancia de "tecnologías no patentadas que pueden contribuir a los Objetivos de Desarrollo Sostenible, a través de diversas fuentes de acceso abierto". Pidió "el desarrollo y la puesta en funcionamiento de una plataforma en línea en el marco del Mecanismo de facilitación de la tecnología para establecer un mapeo integral y servir como puerta de entrada a la información sobre iniciativas, mecanismos y programas de ciencia, tecnología e innovación existentes, dentro y fuera de las Naciones Unidas".

Es un pequeño paso, pero muy emocionante, porque las Naciones Unidas no se demoran una vez que ven formas de ayudar a sus Estados Miembros y a las personas que los integran. Ahora, el Departamento de Asuntos Económicos y Sociales de las Naciones Unidas (DESA) está trabajando para que esto suceda. DESA está utilizando una Nota sobre una base de datos centralizada propuesta de las Naciones Unidas de tecnologías apropiadas de código abierto publicada por la Conferencia de las Naciones Unidas sobre Comercio

y Desarrollo (UNCTAD) para hacerlo.

La Nota de la UNCTAD aboga por una base de datos centralizada de OSAT para acelerar el descubrimiento y la innovación en todos los sectores asociados con los ODS al tiempo que se minimizan los obstáculos legales o financieros. Esto es importante para la difusión del acervo mundial de conocimientos, especialmente en los países en desarrollo.

Actualmente, no existe un repositorio completo o una base de datos central de OSAT y Appropedia.org, quizás sea el mejor ejemplo. Sin embargo, la Nota de la UNCTAD dice: "Muchas organizaciones, organizaciones sin fines de lucro y empresas con fines de lucro están desarrollando OSAT y manteniendo bases de datos existentes a pequeña escala. Si bien hay muchos OSAT disponibles, se encuentran dispersos en varias bases de datos para tecnologías particulares. Mientras tanto, sigue existiendo una clara necesidad de aumentar la tasa de uso de OSAT.

Por lo tanto, existe una necesidad urgente de una base de datos de código abierto centralizada global (COSD) confiable. Al tener un alcance global, un repositorio de COSD proporcionaría una ventanilla única a la que todos pueden acceder para resolver los desafíos locales".

Concluye: "La ONU está bien posicionada para liderar el establecimiento de un COSD dado su papel bien establecido en la promoción de la tecnología de código abierto a través de varios foros y publicaciones intergubernamentales. En particular, 2030 Connect es una plataforma tecnológica en línea de la ONU que se desarrolló como parte del trabajo del Equipo de Trabajo Interinstitucional de la ONU. El COSD podría mejorarlo".

Con el liderazgo de la ONU, quizás no estemos demasiado lejos de cuándo, sí tiene un problema local (sin importar en qué parte del mundo se encuentre), pueda descargar una solución de código abierto examinada y probada. Quizás, esta es la potencia de fuego que necesitamos para alcanzar los ambiciosos Objetivos de Desarrollo Sostenible.

5.6.2 La Comisión Europea se Compromete a Liberar Todo el Software que Pueda Beneficiar a la Sociedad

A finales del 2021, la Unión Europea (UE) y su órgano legislativo, la Comisión Europea siguen avanzando en su estrategia digital con el Software de código abierto como uno de los pilares fundamentales. En esta ocasión ha sido esta última la que anuncia novedades para con la distribución del Software desarrollado para cubrir necesidades internas de la organización.

De acuerdo a la información publicada, la Comisión Europea ha aprobado una nueva regulación que favorece el libre acceso al Software que producen siempre y cuando existan beneficios potenciales para «los ciudadanos, las empresas u otros servicios públicos», lo que de la teoría a la práctica bien puede abarcar todo lo que se desarrolle bajo su tejado.

Esta nueva disposición se apoya a su vez en un reciente estudio realizado también por la Comisión sobre el impacto del Software de código abierto en áreas como la independencia tecnológica, la competitividad y la innovación en la economía de la Unión Europea. El objetivo, hallar evidencias sólidas con las que conformar las políticas europeas de código abierto para los próximos años.

En términos económicos, de hecho, los cálculos son de lo más optimistas y apuntan un impacto económico contundente, de miles de millones de euros de ahorro al año -a modo de ejemplo, se estimó entre 65 y 95 mil millones de euros solo en 2018- y con un incremento mínimo en la apuesta, se podría dar un crecimiento del PIB de la UE de en torno a los 100,000 millones de euros.

Con semejante escenario, no es de extrañar que la misma Comisión Europea esté interesada en promover las soluciones de código abierto dentro y fuera de las instituciones y no solo se basan en el beneficio económico directo: son muchas otras las ventajas del modelo también recogidas en el informe, tal y como se ha mencionado: independencia, competitividad, innovación... y en el caso de las administraciones públicas, colaboración, reutilización y transparencia.

En palabras de Johannes Hahn, comisario de Presupuesto y Administración: «El código abierto ofrece grandes ventajas en un ámbito en el que la UE puede desempeñar un papel de liderazgo. Las nuevas normas aumentarán la transparencia y ayudarán a la Comisión, así como a los ciudadanos, las empresas y los servicios públicos de toda Europa, a beneficiarse del desarrollo de Software de código abierto. Poner en común los esfuerzos para mejorar el Software y la creación conjunta de nuevas funciones reduce los costes para la sociedad, ya que también nos beneficiamos de las mejoras realizadas por otros desarrolladores. Esto también puede mejorar la seguridad, ya que especialistas externos e independientes comprueban los fallos y las deficiencias de seguridad de los programas informáticos».

La comisaría de Innovación, Investigación, Cultura, Educación y Juventud, Mariya Gabriel, ha declarado: «La Comisión pretende, con su ejemplo, estar al frente de la transición digital en Europa. Con las nuevas normas, la

Comisión aportará un valor significativo a las empresas, también las emergentes, a los innovadores, a los ciudadanos y las administraciones públicas, poniendo a su disposición el código abierto de sus soluciones informáticas. Esta decisión también ayudará a estimular la innovación, gracias al código de la Comisión disponible públicamente».

Como muestra del Software desarrollado bajo el amparo de la Comisión Europea que va a ser liberado se incluyen proyectos como eSignature, «un conjunto de normas, herramientas y servicios gratuitos que ayudan a las administraciones públicas y a las empresas a acelerar la creación y verificación de firmas electrónicas jurídicamente válidas en todos los Estados miembros de la UE»; o LEOS (Legislation Editing Open Software), «el Software utilizado en toda la Comisión para elaborar textos jurídicos. LEOS, escrito originalmente para la Comisión, se está desarrollando en estrecha colaboración con Alemania, España y Grecia».

Esta nueva iniciativa de la Comisión Europa contempla asimismo la creación de un repositorio centralizado para facilitar el descubrimiento, el acceso y la reutilización del Software incluido, el cual se sumará a todos los proyectos realizados por las diferentes administraciones públicas comunitarias en base al mismo modelo de desarrollo. Y viene de lejos este impulso, aun cuando comienza a unificarse ahora.

Sin ir más lejos, hace años que la propia Comisión Europea puso en marcha el programa Interoperable Delivery of European eGovernment Services to Public Administrations, Businesses and Citizens que dio origen al observatorio JoinUp (https://joinup.ec.europa.eu/), en cuyas páginas se recogen casi 3,000 soluciones de Software abierto, 133 colecciones de recursos y cuantiosa información relacionada.

Más tarde, de 2014 a 2017 se inició la «primera fase» en la estrategia de código abierto de la Unión Europea, especialmente dentro de la propia Comisión, estableciendo determinados requisitos en materia de Software de código abierto; actualmente se está desarrollando la nueva «estrategia de código abierto 2020-2023», con la que la Comisión Europea pretende ampliar y afianzar los objetivos de la estrategia digital y la contribución al programa Europa Digital.

6 Apéndice B: POO Java vs Python

Muchos programadores de Java en últimas fechas se trasladan a Python, a menudo luchan con el enfoque de Python para la programación orientada a objetos (POO). El enfoque para trabajar con objetos, tipos de variables y otras capacidades de lenguaje tomadas por Python vs Java es bastante diferente. Puede hacer que cambiar entre ambos lenguajes sea muy confuso.

Este apartado se compara y contrasta el soporte de programación orientada a objetos en Python vs Java. Al final, podremos aplicar nuestros conocimientos de programación orientada a objetos a Python al comprender cómo reinterpretar nuestra comprensión de los objetos de Java en Python y utilizar los objetos de una manera Pythonica.

A lo largo de este apartado, podremos:

- Construir una clase básica tanto en Java como en Python
- Explorar cómo funcionan los atributos de objeto en Python vs Java
- Comparar y contrastar los métodos de Java y las funciones de Python
- Descubrir los mecanismos de herencia y polimorfismo en ambos lenguajes
- Investigar la reflexión en Python vs Java
- Aplicar todo en una implementación de clase completa en ambos lenguajes

Este apartado no es una introducción a la programación orientada a objetos. Más bien, compara características y principios orientados a objetos de Python vs Java. Es recomendable que el lector tenga conocimiento de Java y también este familiarizado con la codificación de Python.

6.1 Ejemplo de Clases Java vs Python

Para comenzar, implementaremos la misma clase pequeña tanto en Java como en Python para ilustrar las diferencias entre ellos (se harán modificaciones a ellas a medida que avancemos).

Primero, supongamos que se tiene la siguiente definición de clase Carro en Java:

```
public class Carro {
   private String color;
   private String modelo;
   private int anio;
   public Car(String color, String modelo, int anio) {
      this.color = color;
      this.modelo = modelo;
      this.anio = anio;
   }
   public String getColor() {
      return color;
   }
   public String getmodelo() {
      return modelo;
   }
   public int getAnio() {
      return anio;
   }
}
```

Las clases de Java se definen en archivos con el mismo nombre que la clase. Entonces, se debe guardar esta clase en un archivo llamado *Carro.java*. Solo se puede definir una clase en cada archivo.

Una clase de Carro pequeña similar está escrita en Python de la siguiente manera:

```
class Carro:
   def __init__(self, color, modelo, anio):
      self.color = color
      self.modelo = modelo
      self.anio = anio
```

En Python se puede declarar una clase en cualquier lugar, en cualquier archivo, en cualquier momento. Por ejemplo, guardamos esta clase en el archivo carro.py.

Usando estas clases como base, podremos explorar los componentes básicos de clases y objetos.

6.2 Atributos de Objetos

Todos los lenguajes orientados a objetos tienen alguna forma de almacenar datos sobre el objeto. En Java y Python, los datos se almacenan en atributos, que son variables asociadas con objetos específicos.

Una de las diferencias más significativas entre Python y Java es cómo se definen y administran los atributos de clase y objeto. Algunas de estas diferencias provienen de las limitaciones impuestas por los lenguajes, mientras que otras provienen de las mejores prácticas de programación.

Declaración e Inicialización En Java, declaramos atributos en el cuerpo de la clase, fuera de cualquier método, con un tipo definido. Se debe definir los atributos de la clase antes de que se utilicen:

```
public class Carro {
   private String color;
   private String modelo;
   private int anio;
...
```

En Python, ambos se declaran y definen atributos dentro de la clase .__init__(), que es el equivalente al constructor de Java:

```
def __init__(self, color, modelo, anio):
   self.color = color
   self.modelo = modelo
   self.anio = anio
```

Al prefijar los nombres de las variables con *self*, le dice a Python que estos son atributos. Cada instancia de la clase tendrá una copia. Todas las variables en Python están escritas de forma flexible, y estos atributos no son una excepción.

También se puede crear variables de instancia fuera de .__init__(), pero no es una práctica recomendada ya que su alcance suele ser confuso. Si no se utilizan correctamente, las variables de instancia creadas fuera de .__init__() pueden provocar errores sutiles que son difíciles de encontrar. Por ejemplo, puede agregar un nuevo atributo .ruedas a un objeto *carro* como este:

```
1  >>> import carro
2  >>> mi_carro = carro.Carro("amarillo", "beetle", 1967)
3  >>> print(f"mi carro es {mi_carro.color}")
4  mi carro es amarillo
5
6  >>> mi_carro.ruedas = 5
7  >>> print(f"ruedas: {mi_carro.ruedas}")
8  ruedas: 5
```

Sin embargo, si olvidamos mi_carro.ruedas = 5 en la línea 6, Python muestra un error:

```
1  >>> import Carro
2  >>> mi_carro = Carro.Carro("amarillo", "beetle", 1967)
3  >>> print(f"Mi carro es {mi_carro.color}")
4  Mi carro es amarillo
5
6  >>> print(f"ruedas: {mi_carro.ruedas}")
7  Traceback (most recent call last):
8  File "<stdin>", line 1, in <module>
9  AttributeError: 'Carro' object has no attribute 'ruedas'
```

En Python, cuando declaras una variable fuera de un método, se trata como una variable de clase. Si actualizamos la clase Carro de la siguiente manera:

```
class Carro:
  ruedas = 0
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
```

Esto cambia la forma en que usa la variable .ruedas. En lugar de referirse a él usando un objeto, se refiere a él usando el nombre de la clase:

```
1  >>> import carro
2  >>> mi_carro = carro.Carro("amarillo", "beetle", 1967)
3  >>> print(f"Mi carro es {mi_carro.color}")
4  Mi carro es amarillo
```

```
5
6  >>> print(f"Este tiene {carro.Carro.ruedas} ruedas")
7  Este tiene 0 ruedas
8
9  >>> print(f"Este tiene {mi_carro.ruedas} ruedas")
10  Este tiene 0 ruedas
```

Nota: En Python, se hace referencia a una variable de clase con la siguiente sintaxis:

El nombre del archivo que contiene la clase, sin la extensión .py
Un punto
el nombre de la clase
Un punto
El nombre de la variable

Como guardó la clase Carro en el archivo carro.py, se refiere a la variable de clase ruedas en la línea 6 como carro.Carro.ruedas.

Puedo consultar *mi_carro.ruedas* o *carro.Carro.ruedas*, pero hay que tener cuidado. Cambiar el valor de la variable de instancia *mi_carro.ruedas* no cambiará el valor de la variable de clase *carro.Carro.ruedas*:

```
1  >>> from carro import *
2  >>> mi_carro = carro.Carro("amarillo", "Beetle", "1966")
3  >>> mi_otro_carro = carro.Carro("rojo", "corvette",
"1999")
4
5  >>> print(f"Mi carro es {mi_carro.color}")
6  Mi carro es amarillo
7  >>> print(f"Este tiene {mi_carro.ruedas} ruedas")
8  Este tiene 0 ruedas
9
10  >>> print(f"Mi otro carro es {mi_otro_carro.color}")
11  Mi otro carro es rojo
12  >>> print(f"Este tiene {mi_otro_carro.ruedas} ruedas")
13  Este tiene 0 ruedas
```

```
14
15  >>> # cambiar el valor de la variable de clase
16  ... carro.Carro.ruedas = 4
17
18  >>> print(f"Este tiene {mi_carro.ruedas} ruedas")
19  Este tiene 4 ruedas
20  >>> print(f"Mi otro carro tiene {mi_otro_carro.ruedas}
ruedas")
21  Mi otro carro tiene 4 ruedas
22
23  >>> # Cambiar el valor de la variable de instancia para
mi_carro
24  ... mi_carro.ruedas = 5
25
26  >>> print(f"Mi carro tiene {mi_carro.ruedas} ruedas")
27  Mi carro tiene 5 ruedas
28  >>> print(f"Mi otro carro tiene {m_otro_carro.ruedas}
ruedas")
29  Mi otro carro tiene 4 ruedas
```

Se han definido dos objetos Coche en las líneas 2 y 3:

1. mi_carro
2. mi_otro_carro

Al principio, ambos tienen cero ruedas. Cuando establece la variable de clase usando *carro.Carro.ruedas = 4* en la línea 16, ambos objetos ahora tienen cuatro ruedas. Sin embargo, cuando configura la variable de instancia usando *mi_ carro.ruedas = 5* en la línea 24, solo ese objeto se ve afectado.

Esto significa que ahora hay dos copias diferentes del atributo de *ruedas*:

1. Una variable de clase que se aplica a todos los objetos *Carro*
2. Una variable de instancia específica aplicable solo al objeto *mi_ carro*

No es difícil referirse accidentalmente al incorrecto e introducir errores sutiles.

El equivalente de Java a un atributo de clase es un atributo estático:

```
public class Carro {
  private String color;
  private String modelo;
  private int anio;
  private static int ruedas;
  public Car(String color, String modelo, int anio) {
    this.color = color;
    this.modelo = modelo;
    this.anio = anio;
  }
  public static int getRuedas() {
    return ruedas;
  }
  public static void setRuedas(int count) {
    ruedas = count;
  }
}
```

Normalmente, se refiere a variables estáticas utilizando el nombre de clase de Java. Puede hacer referencia a una variable estática a través de una instancia de clase como Python, pero no es una práctica recomendada.

La clase de Java se está alargando. Una de las razones por las que Java es más detallado que Python es la noción de métodos y atributos públicos y privados.

Pública y Privada Java controla el acceso a métodos y atributos diferenciando entre datos públicos y datos privados.

En Java, se espera que los atributos se declaren como privados o protegidos si las subclases necesitan acceso directo a ellos. Esto limita el acceso a estos atributos desde el código fuera de la clase. Para proporcionar acceso a atributos privados, se declaran métodos públicos que establecen (*setters*) y recuperan (*getters)* datos de manera controlada.

Recordando de la clase Java anterior que la variable de color se declaró como privada. Por lo tanto, este código Java mostrará un error de compilación en la última línea:

```
Carro miCarro = new Carro("azul", "Ford", 1972);
// Pintando el carro
```

```
miCarro.color = "Rojo";
```

Si no especifica un nivel de acceso, el atributo predeterminado es paquete protegido, lo que limita el acceso a las clases en el mismo paquete. Debe marcar el atributo como público si desea que este código funcione.

Sin embargo, declarar atributos públicos no se considera una práctica recomendada en Java. Se espera que declare los atributos como privados y utilice métodos de acceso público, como *.getColor()* y *.getModelo()* que se muestran en el código.

Python no tiene la misma noción de datos privados o protegidos que tiene Java. Todo en Python es público. Este código funciona bien con su clase Python existente:

```
>>> mi_carro = Carro.Carro("azul", "Ford", 1972)
>>> # Pintando el carro
... mi_carro.color = "rojo"
```

En lugar de privada, Python tiene la noción de una variable de instancia no pública. Cualquier variable que comience con un carácter de subrayado se define como no pública. Esta convención de nomenclatura dificulta el acceso a una variable, pero es solo una convención de nomenclatura y aún puede acceder a la variable directamente.

Si Agregamos la siguiente línea a la clase Python Carro:

```
class Carro:
  ruedas = 0
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
    self._portavasos = 6
```

Puede acceder a la variable ._portavasos directamente:

```
1  >>> import carro
2  >>> mi_carro = carro.Carro("amarillo", "Beetle", "1969")
3  >>> print(f"Fue construido en {mi_carro.anio}")
4  Fue construido en 1969
5  >>> mi_carro.anio = 1966
```

```
>>> print(f"Fue construido en {mi_carro.anio}")
Fue construido en 1966
>>> print(f"Tiene {mi_carro._portavasos} portavasos.")
Tiene 6 portavasos.
```

Python reconoce además el uso de caracteres de subrayado doble delante de una variable para ocultar un atributo en Python. Cuando Python ve una variable de subrayado doble, cambia el nombre de la variable internamente para dificultar el acceso directo. Este mecanismo evita accidentes pero aún así no imposibilita el acceso a los datos.

Para mostrar este mecanismo en acción, volvemos a cambiar la clase Python Carro:

```
class Carro:
  ruedas = 0
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
    self.__portavasos = 6
```

Ahora, cuando se intenta acceder a la variable .__portavasos se ve el siguiente error:

```
>>> import Carro
>>> mi_carro = Carro.Carro("amarillo", "Beetle", "1969")
>>> print(f"Fue contruido en {mi_carro.anio}")
Fue construido en 1969
>>> mi_carro.anio = 1966
>>> print(f"Fue construido en {my_car.anio}")
Fue construido en 1966
>>> print(f"Tiene {mi_carro.__portavasos} portavasos.")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Carro' object has no attribute '__portava-
sos'
```

Entonces, ¿por qué no existe el atributo de .__portavasos?

Cuando Python ve un atributo con guiones bajos dobles, cambia el atributo anteponiendo el nombre original del atributo con un guión bajo, seguido del nombre de la clase. Para usar el atributo directamente, también debe cambiar el nombre que usa:

```
>>> print(f"Tiene {mi_carro._Carro__portavasos} portavasos")
Tiene 6 portavasos
```

Cuando usa guiones bajos dobles para ocultar un atributo al usuario, Python cambia el nombre de una manera bien documentada. Esto significa que un desarrollador determinado todavía puede acceder al atributo directamente.

Entonces, si sus atributos de Java se declaran privados y sus atributos de Python están precedidos por guiones bajos dobles, ¿cómo proporciona y controla el acceso a los datos que almacenan?.

Control de acceso En Java, se accede a atributos privados usando *getters* y se establecen usando *setters* de manera controlada. Para permitir que los usuarios pinten sus autos, agregando el siguiente código a la clase de Java:

```
public String getColor() {
  return color;
}
public void setColor(String color) {
  this.color = color;
}
```

Dado que *.getColor* () y *.setColor* () son públicos, cualquiera puede llamarlos para cambiar o recuperar el color del automóvil. Las mejores prácticas de Java de usar atributos privados a los que se accede con captadores *get* y definidores *set* públicos es una de las razones por las que el código Java tiende a ser más detallado que Python.

Como se vio anteriormente, acceder a los atributos directamente en Python. Dado que todo es público, se puede acceder a cualquier cosa en cualquier momento y desde cualquier lugar. Establece y obtiene valores de atributo directamente haciendo referencia a sus nombres. Incluso puede eliminar atributos en Python, lo que no es posible en Java:

```
>>> mi_carro = Carro("amarillo", "beetle", 1969)
>>> print(f"Mi carro fue construido en {mi_carro.anio}")
Mi carro fue construido en 1969
>>> mi_carro.anio = 1966
>>> print(f"Mi carro fue construido en {mi_carro.anio}")
Mi carro fue construido en 1966
>>> del mi_carro.anio
>>> print(f"Mi carro fue construido en {mi_carro.anio}")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Carro' object has no attribute 'anio'
```

Sin embargo, es posible que en ocasiones desee controlar el acceso a un atributo. En ese caso, puede usar las propiedades de Python.

En Python, las propiedades proporcionan acceso controlable a los atributos de la clase mediante la sintaxis del decorador de Python. Las propiedades permiten que se declaren funciones en clases de Python que son análogas a los métodos getter y setter de Java, con la ventaja adicional de permitirle eliminar atributos también.

Podemos ver cómo funcionan las propiedades agregando una a la clase de Carro:

```
class Carro:
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
    self._voltaje = 12
  @property
    def voltaje(self):
    return self._voltaje
  @voltaje.setter
  def voltaje(self, volts):
    print("Aviso: esto puede causar problemas!")
    self._voltaje = volts
  @voltaje.deleter
  def voltaje(self):
    print("Aviso: La radio dejara de funcionar!")
    del self._voltaje
```

Aquí, expandimos la noción de *Carro* para incluir vehículos eléctricos. Al Declarar el atributo *._voltaje* para mantener el voltaje de la batería.

Para proporcionar acceso controlado, definimos una función llamada *voltaje()* para devolver el valor privado. Al usar la decoración *@property*, lo marca como un captador al que cualquiera puede acceder directamente.

De manera similar, se define una función de establecimiento, también llamada *voltaje()*. Sin embargo, decoramos esta función con *@voltaje.setter*. Por último, usamos *@voltaje.deleter* para decorar un tercer *voltaje()*, lo que permite la eliminación controlada del atributo.

Los nombres de las funciones decoradas son todos iguales, lo que indica que controlan el acceso al mismo atributo. Los nombres de las funciones también se convierten en el nombre del atributo que se usa para acceder al valor. Así es como funcionan estas propiedades en la práctica:

```
1  >>> from carro import *
2  >>> mi_carro = Carro("amarillo", "beetle", 1969)
3
4  >>> print(f"Mi carro usa {mi_carro.voltaje} volts")
5  Mi carro usa 12 volts
6
7  >>> mi_carro.voltaje = 6
8  Aviso: esto puede causar problemas!
9
10  >>> print(f"Mi carro ahora usa {mi_carro.voltaje} volts")
11  Mi carro ahora usa 6 volts
12
13  >>> del mi_carro.voltaje
14  Aviso: la radio dejara de trabajar!
```

Tengamos en cuenta que usamos *.voltaje*, no *._voltaje*. Esto le dice a Python que use las funciones de propiedad que se definió:

- Cuando imprime el valor de mi_carro.voltaje en la línea 4, Python llama a *.voltaje()* decorado con *@property*.
- Cuando asigna un valor a mi_carro.voltaje en la línea 7, Python llama a *.voltaje()* decorado con *@voltaje.setter*.

- Cuando elimina *mi_ carro.voltaje* en la línea 13, Python llama a *.voltaje()* decorado con *@voltaje.deleter*.

Las decoraciones *@property*, *@.setter* y *@.delete*r permiten controlar el acceso a los atributos sin requerir que los usuarios utilicen métodos diferentes. Incluso puede hacer que los atributos parezcan propiedades de solo lectura omitiendo las funciones decoradas *@.setter* y *@.deleter*.

self and this En Java, una clase se refiere a sí misma con la referencia *this*:

```
public void setColor(String color) {
  this.color = color;
}
```

this está implícito en el código Java: normalmente no es necesario escribirlo, a menos que pueda haber confusión entre dos variables con el mismo nombre.

Podemos escribir el mismo código de esta manera:

```
public void setColor(String newColor) {
  color = newColor;
}
```

Dado que *Carro* tiene un atributo llamado *.color*, y no hay otra variable en el alcance con el mismo nombre, una referencia a ese nombre funciona. Usamos esto en el primer ejemplo para diferenciar entre el atributo y el parámetro, ambos con nombre color.

En Python, la palabra clave *self* tiene un propósito similar. Así es como se refiere a las variables de miembro, pero a diferencia de *this* de Java, es necesario si desea crear o hacer referencia a un atributo de miembro:

```
class Carro:
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
    self._voltaje = 12
  @property
  def voltaje(self):
    return self._voltaje
```

Python requiere cada uno de ellos en el código anterior. Cada uno crea o hace referencia a los atributos. Si los omite, Python creará una variable local en lugar de un atributo.

La diferencia entre cómo se usa *self* y *this* en Python y Java se debe a diferencias subyacentes entre los dos lenguajes y cómo nombran variables y atributos.

6.3 Métodos y Funciones

Esta diferencia entre Python y Java es, en pocas palabras, que Python tiene funciones y métodos, mientras que Java sólo métodos.

En Python, el siguiente código está perfectamente bien (y es muy común):

```
>>> def di_hola():
... print("Hola!")
...
>>> di_hola()
Hola!
```

Se puede llamar a *di_hola()* desde cualquier lugar que sea visible. Esta función no tiene referencia a *self*, lo que indica que es una función global, no una función de clase. No puede alterar ni almacenar ningún dato en ninguna clase, pero puede usar variables locales y globales.

Por el contrario, cada línea de código Java que escribe pertenece a una clase. Las funciones no pueden existir fuera de una clase y, por definición, todas las funciones de Java son métodos. En Java, lo más cerca que puede llegar a una función pura es mediante el uso de un método estático:

```
public class Utils {
   static void DiHola() {
      System.out.println("Hola!");
   }
}
```

Utils.DiHola() se puede llamar desde cualquier lugar sin crear primero una instancia de *Utils*. Como puede llamar a *DiHola()* sin crear un objeto primero, esta referencia no existe. Sin embargo, esta todavía no es una función en el sentido de *di_hola()* de Python.

6.4 Herencia y Polimorfismo

La herencia y el polimorfismo son dos conceptos fundamentales en la programación orientada a objetos.

La herencia permite a los objetos derivar atributos y funcionalidad de otros objetos, creando una jerarquía que se mueve de objetos más generales a más específicos. Por ejemplo, un automóvil y un bote son tipos específicos de vehículos. Los objetos pueden heredar su comportamiento de un único objeto principal o de varios objetos principales, y se denominan objetos secundarios cuando lo hacen.

El polimorfismo permite que dos o más objetos se comporten entre sí, lo que permite que se usen indistintamente. Por ejemplo, si un método o función sabe cómo pintar un objeto Vehículo, también puede pintar un objeto Coche o Barco, ya que heredan sus datos y comportamiento del Vehículo.

Estos conceptos fundamentales de POO se implementan de manera bastante diferente en Python vs Java.

Herencia Python admite la herencia múltiple o la creación de clases que heredan el comportamiento de más de una clase principal.

Para ver cómo funciona esto, actualicemos la clase de Carro dividiéndola en dos categorías, una para vehículos y otra para dispositivos que usan electricidad:

```
class Vehiculo:
  def __init__(self, color, modelo):
    self.color = color
    self.modelo = modelo
class Dispositivo:
  def __init__(self):
    self._voltaje = 12
class Carro(Vehiculo, Dispositivo):
  def __init__(self, color, modelo, anio):
    Vehiculo.__init__(self, color, modelo)
    Dispositivo.__init__(self)
    self.anio = anio
  @property
  def voltaje(self):
    return self._voltaje
```

```
@voltaje.setter
def voltaje(self, volts):
  print("Aviso: esto puede causar problemas!")
  self._voltaje = volts
@voltaje.deleter
def voltaje(self):
  print("Aviso: el radio dejara de trabajar!")
  del self._voltaje
```

Un vehículo se define por tener atributos *.color* y *.modelo*. Entonces, un dispositivo se define para tener un atributo *._voltaje*. Dado que el objeto Carro original tenía estos tres atributos, se puede redefinir para heredar las clases Vehiculo y Dispositivo. Los atributos de color, modelo y _voltaje serán parte de la nueva clase Carro.

En *.__init__()* para Carro, llama a los métodos *.__init__()* para ambas clases principales para asegurarse de que todo se inicialice correctamente. Una vez hecho esto, puede agregar cualquier otra funcionalidad que desee a su Carro. En este caso, agrega un atributo *.anio* específico para los objetos Carro y los métodos *getter* y *setter* para *.voltaje*.

Funcionalmente, la nueva clase Carro se comporta como siempre. Creas y usa los objetos Carro como antes:

```
>>> from carro import *
>>> mi_carro = Carro("amarillo", "beetle", 1969)
>>> print(f"Mi carro es {mi_carro.color}")
Mi carro es amarillo
>>> print(f"Mi carro usa {mi_carro.voltaje} volts")
Mi carro usa 12 volts
>>> mi_carro.voltaje = 6
Aviso: esto puede causar problemas!
>>> print(f"Mi carro ahora usa {mi_carro.voltaje} volts")
Mi carro ahora usa 6 volts
```

Java, por otro lado, solo admite herencia única, lo que significa que las clases en Java pueden heredar datos y comportamiento de una sola clase principal. Sin embargo, los objetos Java pueden heredar el comportamiento de muchas interfaces diferentes. Las interfaces proporcionan un grupo de métodos relacionados que un objeto debe implementar y permiten que varias clases secundarias se comporten de manera similar.

Para ver esto en acción, divida la clase Java Carro en una clase principal y una interfaz:

```
public class Vehiculo {
  private String color;
  private String modelo;
  public Vehiculo(String color, String modelo) {
    this.color = color;
    this.modelo = modelo;
  }
  public String getColor() {
    return color;
  }
  public String getModelo() {
    return modelo;
  }
}
public interface Dispositivo {
  int getVoltaje();
}
public class Carro extends Vehiculo implements Dispositivo {
  private int voltaje;
  private int anio;
  public Car(String color, String modelo, int anio) {
    super(color, modelo);
    this.anio = anio;
    this.voltaje = 12;
  }
  @Override
  public int getVoltaje() {
    return voltaje;
  }
  public int getAnio() {
    return anio;
  }
}
```

Recordemos que cada clase e interfaz debe vivir en su propio archivo.

Como hicimos con Python, creamos una nueva clase llamada Vehículo para contener los datos y la funcionalidades más generales relacionados con el vehículo. Sin embargo, para agregar la funcionalidad del dispositivo, debe crear una interfaz en su lugar. Esta interfaz define un método único para devolver el voltaje del dispositivo.

La redefinición de la clase *Carro* requiere que heredes de *Vehiculo* usando extender e implemente la interfaz de *Dispositivo* usando implements. En el constructor, llama al constructor de la clase padre usando el *super()* incorporado. Dado que solo hay una clase principal, solo puede hacer referencia al constructor del vehículo. Para implementar la interfaz, escribe *getVoltaje()* usando la anotación *@Override*.

En lugar de obtener la reutilización del código de Dispositivo como lo hizo Python, Java requiere que implemente la misma funcionalidad en cada clase que implementa la interfaz. Las interfaces solo definen los métodos, no pueden definir datos de instancia o detalles de implementación.

Entonces, ¿por qué es este el caso de Java? Todo se reduce a tipos.

Tipos y Polimorfismo La estricta verificación de tipos de Java es lo que impulsa el diseño de su interfaz.

Cada clase e interfaz en Java es un tipo. Por lo tanto, si dos objetos Java implementan la misma interfaz, se considera que son del mismo tipo con respecto a esa interfaz. Este mecanismo permite usar indistintamente diferentes clases, que es la definición de polimorfismo.

Puede implementar la carga del dispositivo para sus objetos Java creando un .cargar() que necesita un dispositivo para cargarse. Cualquier objeto que implemente la interfaz del dispositivo se puede pasar a .cargar(). Esto también significa que las clases que no implementan Dispositivo generarán un error de compilación.

Cree la siguiente clase en un archivo llamado Rinoceronte.java:

```
public class Rinoceronte {
}
```

Ahora puede crear un nuevo *Main.java* para implementar *.cargar()* y explorar en qué se diferencian los objetos Carro y Rinoceronte:

```
public class Main{
```

```
    public static void cargar(Dispositivo dispositivo) {
      dispositivo.getVoltaje();
    }
    public static void main(String[] args) throws Exception {
      Carro carro = new Carro("amarillo", "beetle", 1969);
      Rinoceronte rinoceronte = new Rinoceronte();
      cargar(carro);
      cargar(rinoceronte);
    }
  }
```

Esto es lo que debería ver cuando intente compilar este código:

```
Information:2021-02-02 15:20 - Compilation completed with
1 error and 0 warnings in 4 s 395 ms
Main.java
Error:(43, 11) java: incompatible types: Rinoceronte cannot
be converted to Dispositivo
```

Dado que la clase Rinoceronte no implementa la interfaz del dispositivo, no se puede pasar a *.cargar()*.

En contraste con la escritura de variables estricta de Java, Python usa un concepto llamado escritura de pato, que en términos básicos significa que si una variable "camina como un pato y grazna como un pato, entonces es un pato". En lugar de identificar objetos por tipo, Python examina su comportamiento.

Podemos explorar la escritura de pato implementando capacidades de carga de dispositivo similares para la clase de Dispositivo Python:

```
>>> def cargar(dispositivo):
... if hasattr(Dispositivo, '_voltaje'):
... print(f"Cargando a {dispositivo._voltaje} voltaje disposi-
tivo ")
... else:
... print(f"No se puede cargar {dispositivo.__class__.__name__}")
...
>>> class Telefono(Dispositivo):
... pass
```

```
...
>>> class Rinoceronte:
... pass
...
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> mi_telefono = Telefono()
>>> mi_Rinoceronte = Rinoceronte()
>>> cargar(mi_carro)
Cargando 12 volts dispositivo
>>> cargar(mi_telefono)
Cargando a 12 volts Dispositivo
>>> cargar(mi_Rinoceronte)
No se puede cargar Rinoceronte
```

cargar() debe verificar la existencia del atributo *._voltaje* en el objeto al que se le pasa. Dado que la clase Dispositivo define este atributo, cualquier clase que herede de él (como Carro y Telefono) tendrá este atributo y, por lo tanto, mostrará que se está cargando correctamente. Las clases que no heredan de Dispositivo (como Rinoceronte) pueden no tener este atributo y no podrán cargar (lo cual es bueno, ya que cargar rinocerontes puede ser peligroso).

Métodos Predeterminados Todas las clases de Java descienden de la clase Object, que contiene un conjunto de métodos que heredan todas las demás clases. Las subclases pueden anularlas o mantener los valores predeterminados. La clase Object define los siguientes métodos:

```
class Object {
   boolean equals(Object obj) { ... }
   int hashCode() { ... }
   String toString() { ... }
}
```

De forma predeterminada, *equals()* compara las direcciones del objeto actual con un segundo objeto pasado, y *hashcode()* calcula un identificador único que también usa la dirección del objeto actual. Estos métodos se utilizan en muchos contextos diferentes en Java. Por ejemplo, las clases de

servicios públicos, como las colecciones que clasifican objetos en función del valor, necesitan ambas.

toString() devuelve una representación de cadena del objeto. De forma predeterminada, este es el nombre de la clase y la dirección. Este método se llama automáticamente cuando se pasa un objeto a un método que requiere un argumento de cadena, como *System.out.println()*:

```
Carro carro = new Carro("amarillo", "Beetle", 1969);
System.out.println(carro);
```

Al ejecutar este código, se usará el *.toString()* predeterminado para mostrar el objeto de Carro:

```
Carro@61bbe9ba
```

No es muy útil, ¿verdad? Puede mejorar esto anulando el *.toString()* predeterminado. Agregue este método a la clase Java Carro:

```
public String toString() {
   return "Carro: " + getColor() + " : " + getmodelo() + "
: " + getanio();
}
```

Ahora, cuando ejecutemos el mismo código de muestra, veremos lo siguiente:

```
Carro: amarillo : Beetle : 1969
```

Python proporciona una funcionalidad similar con un conjunto común de *dunder* (abreviatura de "doble subrayado") de métodos. Cada clase de Python hereda estos métodos y puede anularlos para modificar su comportamiento.

Para las representaciones de cadena de un objeto, Python proporciona __repr__() y *__str__()*, que puede conocer en Pythonic OOP String Conversion: *__repr__* vs *__str__*. La representación inequívoca de un objeto es devuelta por *__repr__()*, mientras que *__str__()* devuelve una representación legible por humanos. Estos son aproximadamente análogos a *.hashcode()* y *.toString()* en Java.

Al igual que Java, Python proporciona implementaciones predeterminadas de estos métodos *dunder*:

```
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> print(repr(mi_carro))
<Carro.Carro object at 0x7fe4ca154f98>
>>> print(str(mi_carro))
<Carro.Carro object at 0x7fe4ca154f98>
```

Puede mejorar esta salida anulando *.__str__()*, agregando esto a la clase Python Carro:

```
def __str__(self):
  return f'Carro {self.color} : {self.modelo} : {self.anio}'
```

Esto le da un resultado mucho mejor:

```
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> print(repr(mi_carro))
<Carro.Carro object at 0x7f09e9a7b630>
>>> print(str(mi_carro))
Carro amarillo : Beetle : 1966
```

Anular el método *dunder* nos dio una representación más legible de su Carro. Es posible que también desee anular el *.__repr__()*, ya que a menudo es útil para depurar.

Python ofrece muchos más métodos *dunder*. Usando métodos *dunder*, puede definir el comportamiento de su objeto durante la iteración, comparación, adición o hacer que un objeto sea invocable directamente, entre otras cosas.

Sobrecarga del Operador La sobrecarga de operadores se refiere a redefinir cómo funcionan los operadores de Python cuando operan en objetos definidos por el usuario. Los métodos *dunder* de Python le permiten implementar la sobrecarga de operadores, algo que Java no ofrece en absoluto.

Modificando la clase Python Carro con los siguientes métodos adicionales:

```
class Carro:
  def __init__(self, color, modelo, anio):
    self.color = color
    self.modelo = modelo
    self.anio = anio
```

```
def __str__(self):
  return f'Carro {self.color} : {self.modelo} : {self.anio}'
def __eq__(self, other):
  return self.anio == other.anio
def __lt__(self, other):
  return self.anio < other.anio
  def __add__(self, other):
  return Carro(self.color + other.color,
    self.modelo + other.modelo,
    int(self.anio) + int(other.anio))
```

La siguiente tabla muestra la relación entre estos métodos *dunder* y los operadores de Python que representan:

Método Dunder	Operator	Propósito
__eq__	==	Revisa si son del mismo año
__lt__	<	Cual carro es un modelo más reciente
__add__	+	Concatena dos objetos Carro de una manera absurda

Cuando Python ve una expresión que contiene objetos, llama a los métodos *dunder* definidos que correspondan a los operadores de la expresión. El siguiente código utiliza estos nuevos operadores aritméticos sobrecargados en un par de objetos Carro:

```
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> tu_carro = Carro("rojo", "Corvette", "1967")
>>> print (mi_carro < tu_carro)
True
>>> print (mi_carro > tu_corro)
False
>>> print (mi_carro == tu_carro)
False
>>> print (mi_carro + tu_Carro)
Car amarillorojo : BeetleCorvette : 3933
```

Hay muchos más operadores que puede sobrecargar usando métodos *dunder*. Ofrecen una forma de enriquecer el comportamiento de su objeto de una manera que los métodos predeterminados de la clase base común de Java no lo hacen.

6.5 Reflexión

La reflexión se refiere a examinar un objeto o clase desde dentro del objeto o clase. Tanto Java como Python ofrecen formas de explorar y examinar los atributos y métodos de una clase.

Examinar el Tipo de un Objeto Ambos lenguajes tienen formas de probar o verificar el tipo de un objeto.

En Python, usa *type()* para mostrar el tipo de una variable, e *isinstance()* para determinar si una variable dada es una instancia o hija de una clase específica:

```
>>> mi_carro = Carro("amarillo", "Beetle", "1966")
>>> print(type(mi_carro))
<class 'Carro.Carro'>
>>> print(isinstance(mi_carro, Carro))
True
>>> print(isinstance(mi_carro, Dispositivo))
True
```

En Java, consulta el objeto por su tipo usando *.getClass()*, y usa el operador *instanceof* para verificar una clase específica:

```
Carro carro = new Carro("amarillo", "beetle", 1969);
System.out.println(carro.getClass());
System.out.println(carro instanceof Carro);
```

Este código genera la siguiente salida:

```
class com.realpython.Carro
true
```

Examinar los Atributos de un Objeto En Python, puede ver todos los atributos y funciones contenidos en cualquier objeto (incluidos todos los métodos *dunder*) usando *dir()*. Para obtener los detalles específicos de un atributo o función determinados, use *getattr()*:

```
>>> print(dir(mi_carro))
['_Carro__portavasos', '__add__', '__class__', '__de-
lattr__', '__dict__',
'__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__',
'__gt__', '__hash__', '__init__', '__init_subclass__',
'__le__', '__lt__',
'__module__', '__ne__', '__new__', '__reduce__', '__re-
duce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__',
'__weakref__',
'_voltaje', 'color', 'modelo', 'voltaje', 'ruedas', 'anio']
>>> print(getattr(mi_carro, "__format__"))
<built-in method __format__ of Carro object at 0x7fb4c10f5438>
```

Java tiene capacidades similares, pero el control de acceso del lenguaje y la seguridad de tipos lo hacen más complicado de recuperar.

.getFields() recupera una lista de todos los atributos de acceso público. Sin embargo, dado que ninguno de los atributos de Carr es público, este código devuelve una matriz vacía:

```
Field[] fields = carro.getClass().getFields();
```

Java trata los atributos y métodos como entidades independientes, por lo que los métodos públicos se recuperan mediante *.getDeclaredMethods()*. Dado que los atributos públicos tendrán un método *.get* correspondiente, una forma de descubrir si una clase contiene una propiedad específica podría verse así:

> Utilice *.getDeclaredMethods()* para generar una matriz de todos los métodos.
>
> Recorra todos los métodos devueltos:
>
> Para cada método descubierto, devuelve verdadero si el método:
>
> Comienza con la palabra get OR acepta cero argumentos
>
> Y no vuelve vacío
>
> Y incluye el nombre de la propiedad
>
> De lo contrario, devuelve falso.

Aquí hay un ejemplo rápido y sucio:

```
    public static boolean getProperty(String name, Object ob-
ject) throws Exception {
      Method[] declaredMethods = object.getClass().getDeclaredMethods();
      for (Method method : declaredMethods) {
        if (isGetter(method) &&
          method.getName().toUpperCase().contains(name.toUpperCase()))
{
            return true;
        }
      }
      return false;
    }
    // Función auxiliar para obtener si el método es un método
getter
    public static boolean isGetter(Method method) {
      if ((method.getName().startsWith("get") ||
        method.getParameterCount() == 0 ) &&
        !method.getReturnType().equals(void.class)) {
          return true;
      }
      return false;
    }
```

getProperty() es su punto de entrada. Llamandolo con el nombre de un atributo y un objeto. Devuelve verdadero si se encuentra la propiedad y falso si no.

Métodos de Llamada a Través de la Reflexión Tanto Java como Python proporcionan mecanismos para llamar a métodos mediante la reflexión.

En el ejemplo de Java anterior, en lugar de simplemente devolver verdadero si se encuentra la propiedad, puede llamar al método directamente. Recuerde que *getDeclaredMethods()* devuelve una matriz de objetos *Method*. El objeto *Method* en sí tiene un método llamado *.invoke()*, que llamará al método. En lugar de devolver true cuando el método correcto se encuentra en la línea 7 anterior, puede devolver *method.invoke(object)* en su lugar.

Esta capacidad también existe en Python. Sin embargo, dado que Python no distingue entre funciones y atributos, debe buscar específicamente entradas que sean invocables:

```
>>> for method_name in dir(mi_carro):
... if callable(getattr(mi_carro, method_name)):
... print(method_name)
...
__add__
__class__
__delattr__
__dir__
__eq__
__format__
__ge__
__getattribute__
__gt__
__init__
__init_subclass__
__le__
__lt__
__ne__
__new__
__reduce__
__reduce_ex__
__repr__
__setattr__
__sizeof__
__str__
__subclasshook__
```

Los métodos de Python son más simples de administrar y llamar que en Java. Agregar el operador () (y cualquier argumento requerido) es todo lo que necesita hacer.

El siguiente código encontrará el *.__str__ ()* de un objeto y lo llamará a través de la reflexión:

```
>>> for method_name in dir(mi_carro):
... attr = getattr(mi_carro, method_name)
... if callable(attr):
... if method_name == '__str__':
... print(attr())
...
Carro amarillo : Beetle : 1966
```

Aquí, se comprueba cada atributo devuelto por *dir()*. Obtiene el objeto de atributo real usando *getattr()* y verifica si es una función invocable usando *callable()*. Si es así, compruebe si su nombre es *__str__()* y luego llámelo.

7 Bibliografía

Este texto es una recopilación de múltiples fuentes, nuestra aportación -si es que podemos llamarla así- es plasmarlo en este documento, en el que tratamos de dar coherencia a nuestra visión de los temas desarrollados.

En la realización de este texto se han revisado -en la mayoría de los casos indicamos la referencia, pero pudimos omitir varias de ellas, por lo cual pedimos una disculpa- múltiples páginas Web, artículos técnicos, libros, entre otros materiales bibliográficos, los más representativos y de libre acceso los ponemos a su disposición en la siguiente liga:

Herramientas
http://132.248.181.216/Herramientas/

Referencias

[1] https://www.gnu.org/philosophy/free-sw.es.html 147

[2] https://es.wikipedia.org/wiki/Software_libre 147

[3] https://www.hispaLinux.es/SoftwareLibre 147

[4] https://es.wikipedia.org/wiki/Software_propietario 145

[5] Diferentes Tipos de Licencias para el Software, https://www.gnu.org/licenses/license-list.html 147, 157

[6] FSF, Free Software Fundation, http://www.fsf.org/ 147, 157

[7] GNU Operating System, http://www.gnu.org/ 147, 157

[8] GCC, the GNU Compiler Collection, http://gcc.gnu.org/

[9] The Linux Kernel Archives, http://www.Kernel.org/

[10] El economista, https://eleconomista.com.mx/tecnociencia/2013/01/22/clausuraran-negocios-mexico-uso-ilegal-Software

[11] PCworld, http://www.pcworld.com.mx/UNAM-y-BSA-promueven-el-uso-de-Software-legal/

[12] https://es.wikipedia.org/wiki/SQL 11

[13] https://es.wikipedia.org/wiki/NoSQL

11

Declaramos terminado este trabajo sufrido, ideado y llevado a cabo entre los años 2020 al 2025, aún y a pesar de impedimentos tales como: la mala suerte, la desventura, el infortunio, la incomprensión, la gripe, el COVID-19, la migraña, las horas de frío y calor, la tristeza, la desesperanza, el cansancio, el presente, el pasado y nuestro futuro, el que dirán, la vergüenza, nuestras propias incapacidades y limitaciones, nuestras aversiones, nuestros temores, nuestras dudas y en fin, todo aquello que pudiera ser tomado por nosotros, o por cualquiera, como obstáculo en este tiempo de mentiras, verdades, de incredulidad e ignorancia o negación de la existencia real y física de la mala fe.

Atentamente

Antonio Carrillo Ledesma
Karla Ivonne González Rosas