DigitalHouse > Coding School

DATA SCIENCE

UNIDAD 3
MÓDULO 5

Introducción a Bases de Datos, SQL y Clasificación

Octubre 2017

INTRODUCCIÓN

BASES DE DATOS Y SQL

OBJETIVOS DE LA CLASE

1. **Reconocer las Bases de Datos más comunes y sus aplicaciones.**

2. **Conocer las características de una Base de Datos Relacional.**

3. **Saber dónde residen y desde dónde se utilizan las Bases de Datos.**

4. **Describir lo que significa SQL y NoSQL con sus ventajas y desventajas.**

BASES DE DATOS

- Una **Base de Datos** es un conjunto de datos **relacionados** y **organizados** de tal manera que puedan responder a un **propósito específico**. Representan aspectos de la realidad (Dato != Información).

- Con el término “Base de Datos” también se hace referencia al software (**DBMS**) que permite administrarla, es decir, gestionar su tamaño, cargarle datos y consultarlos.

- Permiten **organizar** los datos con métodos eficientes para **obtener la información requerida** y crear reglas para asegurarse de que **los datos se almacenen de forma correcta y consistente**.

- El lenguaje que permite obtener datos de una Base de Datos se llama **SQL** (Structured Query Language).

DNI	NOMBRE	APELLIDO	EDAD	SEXO
12121212	Martín	Martinez	56	M
23232323	Carmen	Carter	40	F
34343434	Pablo	Ponce	23	M

- Los datos se estructuran en **tablas, filas** y **columnas**.
- En su definición más formal se utilizan los términos: relaciones, tuplas y atributos, respectivamente.
- Restricciones básicas que, mediante reglas, pueden aplicarse a una tabla:
- DNI debe tener un valor único (**Clave Primaria**).
- EDAD debe ser un número entero positivo.
- SEXO sólo puede contener M y F (**Clave Foránea** a tabla que indica M=Masculino, F=Femenino).

EJEMPLO DE APLICACIÓN

Base de Datos para un Banco

Un banco registra en una tabla como la siguiente, las transacciones bancarias que hacen sus clientes.

- ¿Qué pasaría si en vez de una Base de Datos fuese un archivo de texto? ¿Cómo sería el uso a través de un Homebanking?
- Problemas que puedan afectar la:
- **Consistencia**
- **Disponibilidad**
- **Tolerancia a la partición**
- **Escalabilidad**

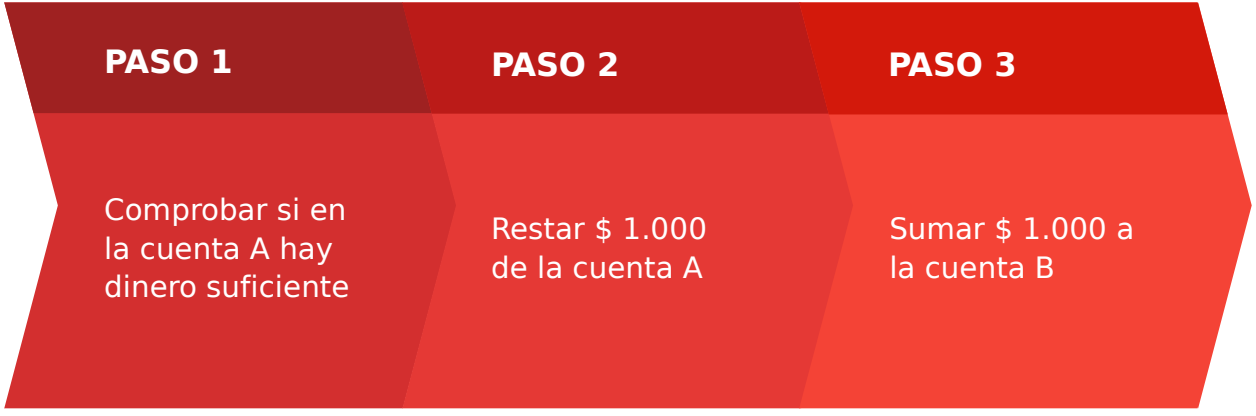
Nro. Cuenta	Saldo
123	$ 1.000
456	$ 1.550
789	$ 2.400
…	…

INTEGRIDAD TRANSACCIONAL

Ejemplo de una Transacción bancaria

- Una **transacción** es un conjunto de operaciones, pertenecientes a una misma tarea, que se realizan sobre una base de datos y representan un cambio en los datos.

Ejemplo de una Transacción bancaria



- ¿Qué pasa si el proceso falla entre el paso 2 y el 3 (o sólo se ejecutan los pasos 1 y 2)?
- ¿Qué obtiene alguien que quiere consultar su Saldo entre los pasos mencionados?

- Por definición, las Transacciones deben ser **atómicas, consistentes, aisladas y durables**.
- Estas propiedades se las conocen como ACID, por sus siglas en inglés:
- **A**tomicity: Todas las modificaciones de una transacción deben ser ejecutadas o bien, ninguna de ella. Se ejecuta “todo o nada”.
- **C**onsistency: La transacción comienza a ejecutarse en un estado válido (consistente) y finaliza en otro estado válido. Para eso, deben cumplirse todas las reglas de integridad que hayan sido definidas.
- **I**solation: Dos o más transacciones *concurrentes* se ejecutan sin afectarse entre sí. Si ambas transacciones deben trabajar sobre el mismo dato una de ellas deberá esperar a que la otra termine.
- **D**urability: Asegura que una vez realizada la operación va a persistir (quedarán almacenados los cambios) por más que falle el sistema. Uso del Transaction Log del DBMS.

BASE DE DATOS RELACIONAL

Tablas

- Compuestas de filas y columnas donde se almacenan los datos, que están relacionadas de acuerdo a las entidades o conceptos que representan.

- Se le asocian reglas que son aplicadas cuando se intenta cargarles algún dato:
  - Cuántos campos tiene y cómo se llaman
  - De qué tipo son (textos, números enteros, fechas, etc.)

- Son más consistentes que un archivo CSV.

- Al menos una columna debe funcionar como **Clave Primaria**.

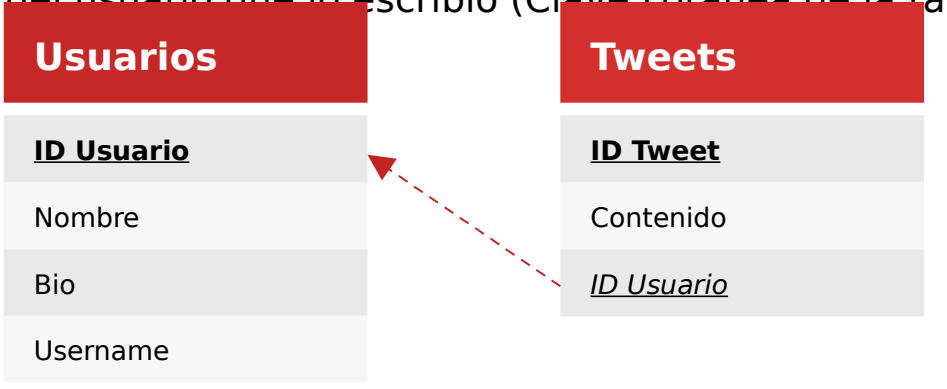
- Puede contener una **Clave Foránea** que haga referencia a la Clave Primaria de otra tabla.

- Algunas implementaciones de Bases de Datos Relacionales:
  - MySQL y PostgreSQL: implementaciones open source, gratuitas y ampliamente utilizadas por sitios web.
  - Oracle y Microsoft SQL Server: utilizadas principalmente por grandes empresas.
  - Todas utilizan SQL para consultas, aunque poseen algunas diferencias en características para gestionarla.

EJEMPLO DE APLICACIÓN

Twitter

- Las principales **entidades** o conceptos: Usuarios y Tweets. Cada una representada por una **tabla**.
- La tabla Usuarios tendrá el ID del usuario (Clave Primaria), su nombre y otros datos personales.
- La tabla Tweets tendrá el ID del Tweet (Clave Primaria), el texto o contenido del Tweet y el ID del usuario que lo escribió (Clave Foránea de la tabla de Usuarios).



DISEÑAR UNA BASE DE DATOS RELACIONAL

Uber

Ejercicio para Practicar

- Considerar el sistema de transporte Uber en la cual existen Usuarios que contactan a Conductores para realizar un Viaje, en determinado Momento. En el mismo se registra cuándo inicia el viaje, el lugar inicial y el destino final, la duración, distancia, tarifa y la forma de pago.

DISEÑO
1. ¿Cómo diseñaría una Base de Datos Relacional para soportar estos datos?

TABLAS
2. Nombrar las tablas que tendría la Base de Datos.

CAMPOS
3. Indicar los campos contenidos en cada una de las tablas.

RELACIONES
4. ¿Cómo se relacionarían entre ellas?

BASES DE DATOS ALTERNATIVAS

- Difieren de las Bases de Datos Relacionales principalmente en su modelo de datos.
- No necesariamente se utiliza SQL para consultarlas (Bases de Datos NoSQL, “Not Only SQL”)
- Open-source, Distribuidas y de Escalabilidad Horizontal
- Priorizan la rapidez (para escrituras y lecturas simples) y flexibilidad antes que la consistencia (no son ACID) → Consistency, Availability & Partition tolerance trade-off
- Soportan grandes volúmenes de datos (“Big Data”) que pueden ser semi o no estructurados (Text, Log Files, Click Streams, Blogs, Tweets, etc.)

- En vez de utilizar tablas relacionadas, con filas y columnas de similares características, permiten diseñar su Modelo de Datos mediante otros objetos.

- Los más comunes son:

  Key-Value
  Document
  Time Series
  Graph

¡Hay decenas de modelos de datos y más de 200 sistemas de Bases de Datos NoSQL!

http://nosql-database.org/

Key-Value

- Los datos consisten en una clave indexada (y única) y un valor asociado (diccionarios o *hash-maps*).
- Algoritmos de caché inteligentes permiten lecturas/escrituras muy rápidas, si se cuenta con la Key.
- Trabaja igual que los diccionarios de Python pero en vez de almacenarse en memoria RAM (mientras se ejecuta el programa) lo hacen en disco, lo que permite que tengan un tamaño mucho mayor.
- Se utilizan como sistemas de **almacenamiento caché**, cuando se requieren almacenar la fecha y hora del último acceso de un usuario a determinado sitio web, contabilizar ciertas acciones que realizan algunos usuarios/clientes, etc.

Document

- Se organizan los datos a partir de una Entidad, que es más flexible que una Tabla Relacional.
- Suelen tener estructuras de datos *desnormalizados* y anidados (similares a JSON o XML), permitiendo relacionarlos entre sí en vez de referenciar otras tablas o entidades.
- Fácil de evolucionar agregando nuevos campos a un mismo Document.
- Se puede consultar un Document mediante su key o cualquier otro campo

```
{
    FirstName: "Bob",
    Address: "5 Oak St.",
    Hobby: "sailing"
}
```

```
<contact>
  <firstname>Bob</firstname>
  <lastname>Smith</lastname>
  <phone type="Cell">(123) 555-0178</phone>
  <phone type="Work">(890) 555-0133</phone>
  <address>
    <type>Home</type>
    <street1>123 Back St.</street1>
    <city>Boys</city>
    <state>AR</state>
    <zip>32225</zip>
    <country>US</country>
  </address>
</contact>
```

Algunos ejemplos:

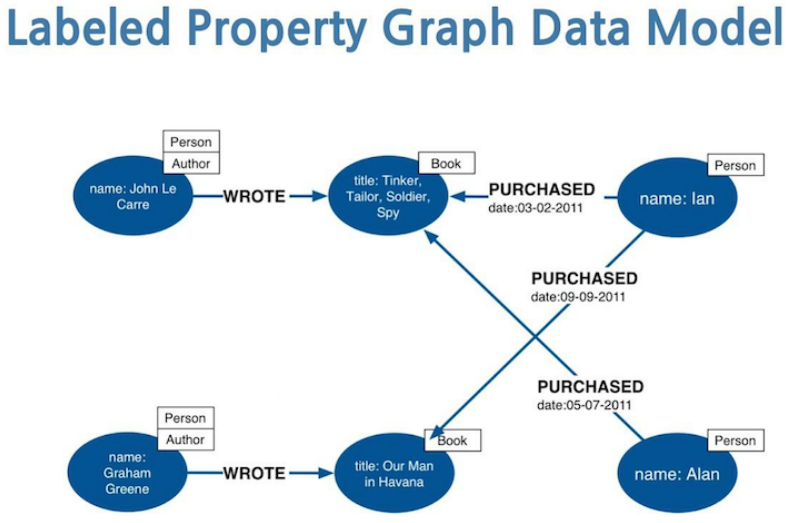
Time series

- Están optimizadas para manejar series de datos de tiempo, es decir que los datos están indexados por el tiempo (una fecha/hora o un rango de tiempo).
- Posibles aplicaciones: el valor de una acción de la bolsa, consumo energético, métricas de un servidor, historial de compras/ventas, métricas de websites (ads y clicks), datos de sensores de dispositivos IoT o de un Smartphone, etc.

Algunos ejemplos:

Graph

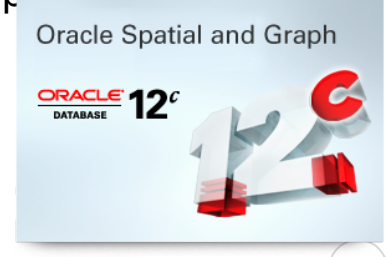
- Optimizadas para almacenar datos acerca de redes.
- Tienen naturaleza NoSQL con almacenamiento key-value o document.
- Cuentan con el concepto de Relaciones que pueden ser de distintas maneras, permitiendo atravesar jerarquías de datos complejas.
- Algunas utilizan el concepto de propiedad o etiquetado como para clasificar los Document



Fuente: https://neo4j.com/developer/graph-database/

Algunos ejemplos:

neo4j



AllegroGraph

BASES DE DATOS ALTERNATIVAS

Ejemplos de Aplicación

Considerar un gran sitio web de e-commerce.

- Catálogo con muchos productos, que precisan almacenarse en una Base de Datos.
- Diferentes productos pueden tener distintas propiedades.
- Se precisa hacer control del inventario de productos, el cual se ve afectado por los carritos de compras de los usuarios.

¿Qué características de las Bases de Datos relacionales no serían adecuadas para una implementación como esta?

Bases de Datos Time Series (TSDB)

- Las series de tiempo presentan desafíos que no suelen ser resueltos con un modelo de Base de Datos relacional.

¿Qué problemas podrían presentarse al intentar modelar series de datos con un modelo de tablas relacionales?

Bases de Datos Graph

- Una empresa cuenta con información de los llamados telefónicos, donde cada entidad tiene los datos: ID_emisor, ID_receptor, fecha_hora_llamada, duración_llamada.

- Un usuario puede realizar varias llamadas y algunos usuarios pueden estar más conectados que otros.

- A la empresa le interesa saber quiénes son los usuarios principales, los más conectados, para darles alguna promoción.

¿Cómo se podría hacer, conceptualmente, para obtener esa información en una Graph DB y en una Base de Datos Relacional?

BASES DE DATOS ALTERNATIVAS

Propiedades BASE

- Una alternativa a las propiedades ACID, características de las Base de Datos Relacionales, son las **BASE**, típicamente asociadas a las base de datos NoSQL.
- **B**asically **A**vailable: el sistema no garantiza disponibilidad.
- **S**oft state: el estado del sistema puede variar en el tiempo incluso sin inputs.
- **E**ventual consistency: el sistema será consistente a lo largo del tiempo si no está recibiendo inputs durante ese tiempo.
- **Principales objetivos**: deshacerse de los locks, permitir escribir a todos los usuarios, preocuparse por la consistencia después. No garantiza consistencia inmediata, a diferencia de las ACID.

BASE DE DATOS LOCAL

- Una Base de Datos puede ser local o remota, contenida en un único servidor o distribuido en varias máquinas con los datos replicados. La configuración final que tendrá una BD una vez particionada o distribuida se denomina *sharding*.

Ejemplo de una Base de Datos local

- SQLite (https://sqlite.org/) es un software que permite utilizar una Base de Datos mediante SQL.

- Está basada en un archivo (*no posee una arquitectura cliente-servidor*).

- Resulta sencillo y útil para pequeños proyectos. Para ambientes productivos conviene migrar hacia otras Bases de Datos SQL más robustas.

- SQLite v3 está integrado en la mayoría de las distribuciones de **Python** (*inclusive Anaconda*).

- Herramientas con GUI para administrar Bases de Datos SQLite:
  Add-on de Firefox SQLite Manager
  DB Browser for SQLite (http://sqlitebrowser.org/)

— Existen distintas maneras para interactuar con una Base de Datos SQLite, por ejemplo:

1. Mediante la línea de comandos ejecutando sqlite3.exe.

2. Paquete sqlite3 de Python.

3. Interfaz SQL **pandas** (http://pandas.pydata.org/).

4. Lenguajes “Object-relational mapping” de alto nivel (*de uso similar a los lenguajes de OOP*). Ejemplos para usar en Python:
   Peewee: http://peewee.readthedocs.io/en/latest/index.html
   Object: https://github.com/emilianobilli/nonsqlite

*En todos estos casos se utilizan **sentencias SQL** “disfrazadas” en funciones específicas que las encapsulan.*

SQL (Structured Query Language) es un lenguaje que permite administrar Bases de Datos relacionales.

- DDL (Data Definition Language): Permiten definir la estructura (e.g.: tablas) que almacene los datos:

  CREATE TABLE MyTable (Campo1 Tipo1, Campo2 Tipo2);
  ALTER TABLE MyTable ADD COLUMN Campo3 Tipo3;
  DROP TABLE MyTable;

- DML (Data Manipulation Language): Permiten manipular los datos:

  INSERT INTO MyTable (Campo1, Campo2) VALUES (Valor1, Valor2);
  UPDATE MyTable SET Campo1 = Valor1, Campo2 = Valor2;
  DELETE FROM MyTable;
  SELECT * FRO

Estructura / DDL

Datos / DML

DNI	NOMBRE	APELLIDO	EDAD	SEXO
12121212	Martín	Martinez	56	M
23232323	Carmen	Carter	40	F
34343434	Pablo	Ponce	23	M

- Abrir la terminal (Shell, línea de comandos, etc.) y ejecutar **sqlite3** seguido del nombre del archivo que conforma la Base de Datos. Si el archivo no existe, lo crea.

```
> sqlite3 mi_bd.sqlite
SQLite version 3.15.1 2016-11-04 12:08:49
Enter ".help" for usage hints.
sqlite>
```

Al ejecutar **.help** se pueden ver los distintos comandos propios de SQLite.
Algunos que serán útiles después:

.databases
.tables
.schema
.dump
.exit

— Escribir los siguientes comandos SQL sobre mi_bd.sqlite:

1. Crear una tabla con un campo que sea Primary Key (PK).
2. Mediante ALTER TABLE agregarle más campos.
3. Agregarle datos a la tabla.
4. Intentar insertar un registro con un valor de PK existente. ¿Qué sucede? ¿Por qué?
5. Ejecutar una sentencia INSERT y al campo que es PK ponerle valor NULL. ¿Qué sucede? ¿Por qué?
6. Realizar modificaciones y eliminaciones de datos
7. Salir con el comando .exit

Tipos de datos admitidos en SQLite: www.sqlite.org/datatype3.html

BASE DE DATOS REMOTA

Ejemplo de una Base de Datos remota

- PostgreSQL (https://www.postgresql.org/) es un software que permite gestionar una Base de Datos Relacional, en una arquitectura cliente-servidor.

- Puede correr en diferentes plataformas, incluyendo Linux, UNIX y Windows.

- Soporta la mayoría de los tipos de datos SQL (http://www.w3schools.com/sql/sql_datatypes.asp), incluyendo BLOb, Imágenes, Sonidos, etc.

- Posee interfaces de programación nativa para C/C++, Java, .Net, Perl, Python, Ruby, ODBC y otros.

- Herramienta con GUI para administrar Bases de Datos PostgreSQL:
  pgAdmin (https://www.pgadmin.org/)
  Muchas otras herramientas (algunas open-source y otras comerciales) en:
https://www.postgresql.org/download/products/1-administ elopment-tools/

- Para conectarse por línea de comandos, a un server instalado en la máquina local, utilizando el port default, y autenticarse a la misma con el usuario *postgres* se debe tipear "`psql -U postgres`" y luego la contraseña para dicho usuario.
- Para conectarse a un servidor remoto, acceder de la siguiente manera:

```
> psql -h dsi.c20gkj5cvu3l.us-east-1.rds.amazonaws.com -p 5432 -U
dsi_student titanic
Contraseña para usuario dsi_student:
psql (9.6.1, servidor 9.4.7)
ADVERTENCIA: El código de página de la consola (850) difiere del código
             de página de Windows (1252).
             Los caracteres de 8 bits pueden funcionar incorrectamente.
             Vea la página de referencia de psql «Notes for Windows
users»
             para obtener más detalles.
conexión SSL (protocolo: TLSv1.2, cifrado: ECDHE-RSA-AES256-GCM-SHA384,
bits: 256, compresión: desactivado)
Digite «help» para obtener ayuda.

titanic=>
```

— Algunos comandos interesantes (*¡si se conectaron con –E se podrán ver las querys SQL internas!*):

\q: Salir
\c __database__: Conectarse a una Base de Datos
\d __table__: Mostrar la definición/estructura de una tabla
\dt *.*: Listar los nombres de las tablas de todos los schemas
\l: Listar las bases de datos
Admite sentencias SQL completas, al igual que SQLite.

Comparación de comandos básicos en distintas Bases de Datos

Tarea	MySQL	PostgreSQL	SQLite
Conectarse a una BD	mysql <dbname>	psql <dbname>	sqlite3 <filename>
Ayuda del cliente	help contents	\?	.help
Ayuda de SQL	help contents	\h	n/a
Listar Bases de Datos	SHOW DATABASES;	\l	.databases
Usar Base de Datos	USE <dbname>	\c <dbname>	n/a
Listar tablas	SHOW TABLES;	\dt	.tables
Mostrar definición de una tabla	info DESCRIBE <tablename>	\d <tablename>	.schema <tablename>

Opciones y parámetros de la línea de comandos, así como comandos y operadores dentro del entorno:
https://www.postgresql.org/docs/9.2/static/app-psql.html

PRÁCTICA LIBRE DE COMANDOS SQL Y DEL CLIENTE

- Individualmente, optar por el sistema de Bases de Datos a utilizar (SQLite, PostgreSQL u otro).

- Pensar en un uso sencillo de una Base de Datos.
- Ejecutar los comandos necesarios para crearla/conectarse a la misma, diseñar sus tablas, crearlas y agregarle datos de ejemplo.

- Luego listar las tablas, ver sus estructuras/definiciones, modificar/eliminar tablas.

- Consultar la ayuda para poder realizar las operaciones requeridas.

CONCLUSIONES

- Las Bases de Datos relacionales son las más utilizadas.
- Organizan sus datos en tablas, con filas y columnas.
- Existen otros tipos de bases de datos (key-value, document, graph, etc.)
- Conocimos SQLite, que es la Base de Datos local más sencillas.
- Aprendimos a agregarle datos y luego a visualizarlos.
- También nos conectamos a una Base de Datos remota (PostgreSQL)
- Y aprendimos algunos comandos SQL básicos!

INTRODUCCIÓN

BASES DE DATOS Y SQL