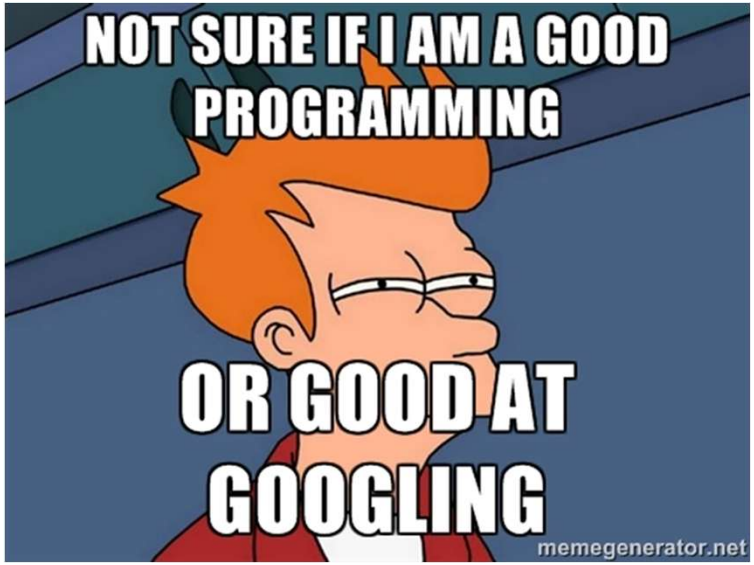
Ciencia de Datos y BigData

Exploración y Curación de Datos

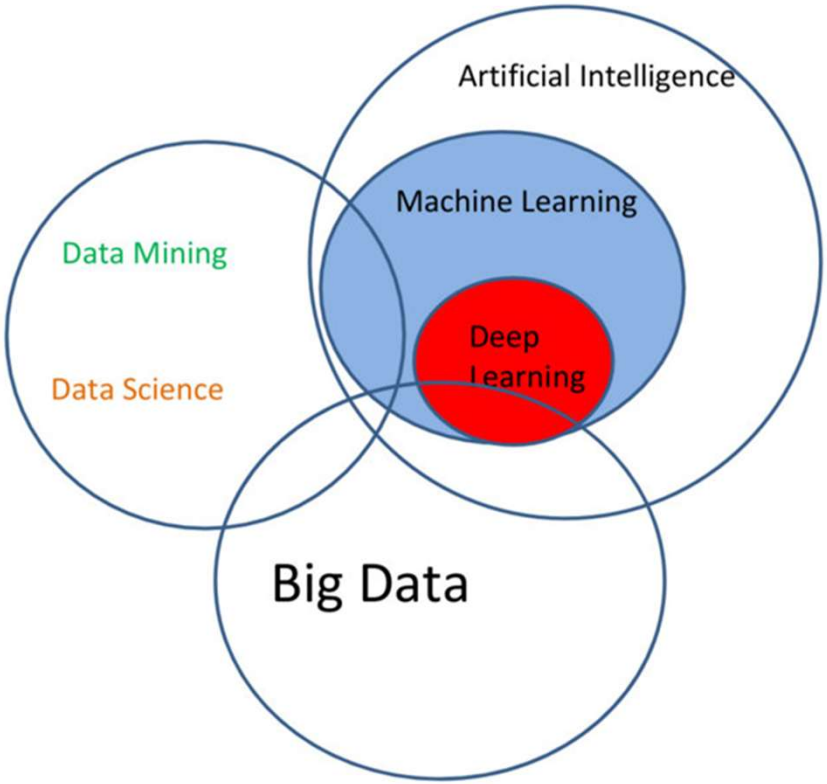
Dr. José Ramón Iglesias
DSP-ASIC BUILDER GROUP
Director Semillero TRIAC
Ingenieria Electronica
Universidad Popular del Cesar

Programar vs Googlear

- Aprender a Googlear con la abstracción suficiente
- Stack Overflow
- Entender mensajes de error
- Leer documentacion, ej: https://pandas.pydata.org/docs/



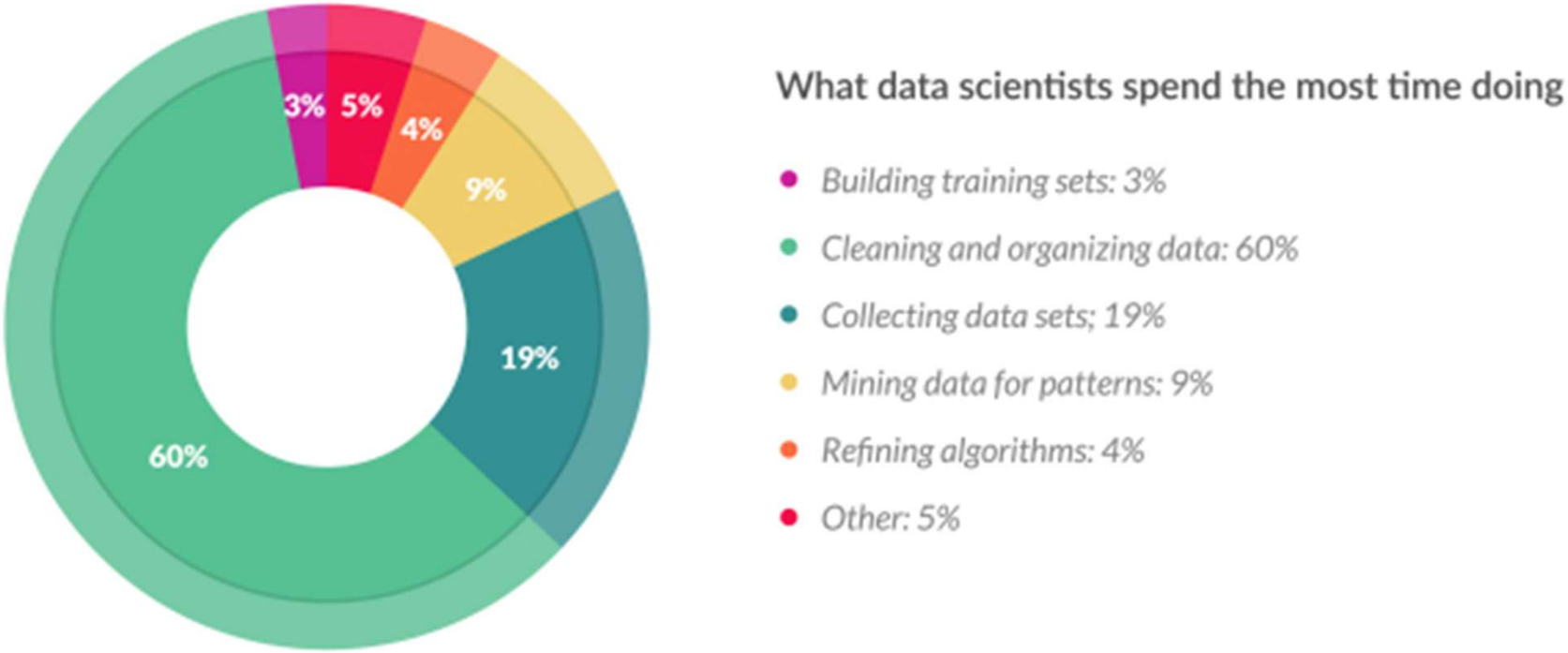
Conceptos relacionados



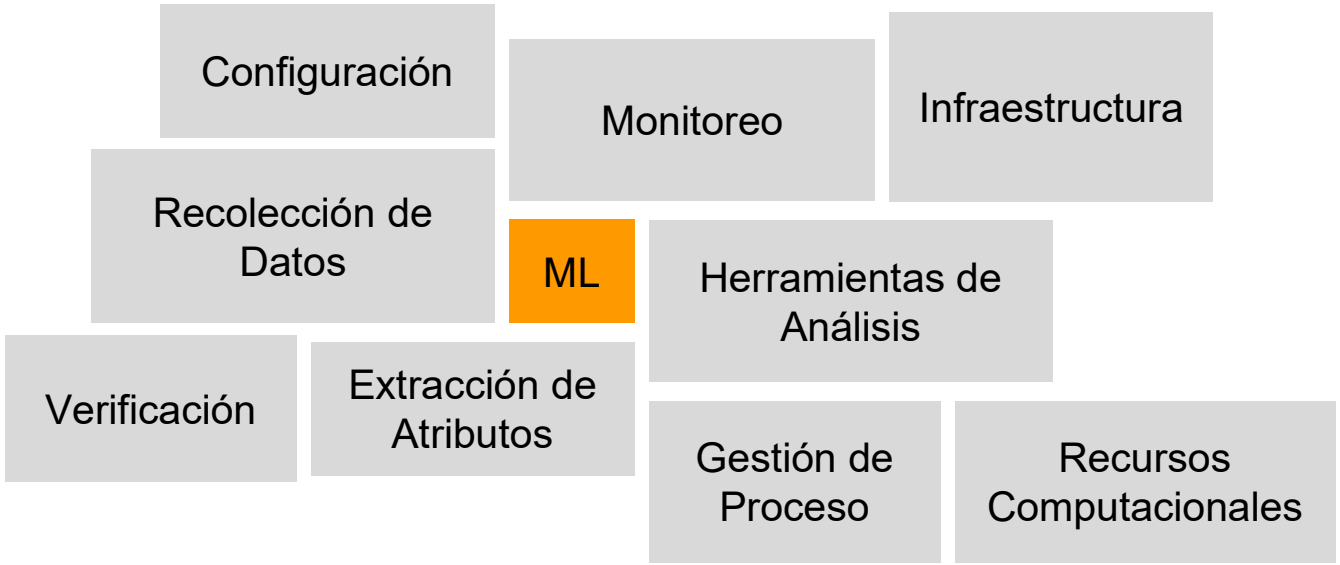
Conceptos Relacionados

- **Inteligencia Artificial**: Computadoras emulando comportamientos humanos específicos.
- **Big Data**: Disponibilidad de grandes volúmenes de data y la capacidad de procesarlos.
- **Machine Learning**: Predecir un comportamiento basado en métodos estadísticos, encontrando patrones.
- **Deep Learning**: Subconjunto de ML, sin necesidad de definir características pero requiere de mucho mayor volumen de datos.
- **Business Intelligence**: Orientado a medir y reportar eventos del pasado
- **Data Science**: Transformar datos e hipótesis en información para la toma de decisiones o para predecir acciones.

Limpieza y Exploración de datos

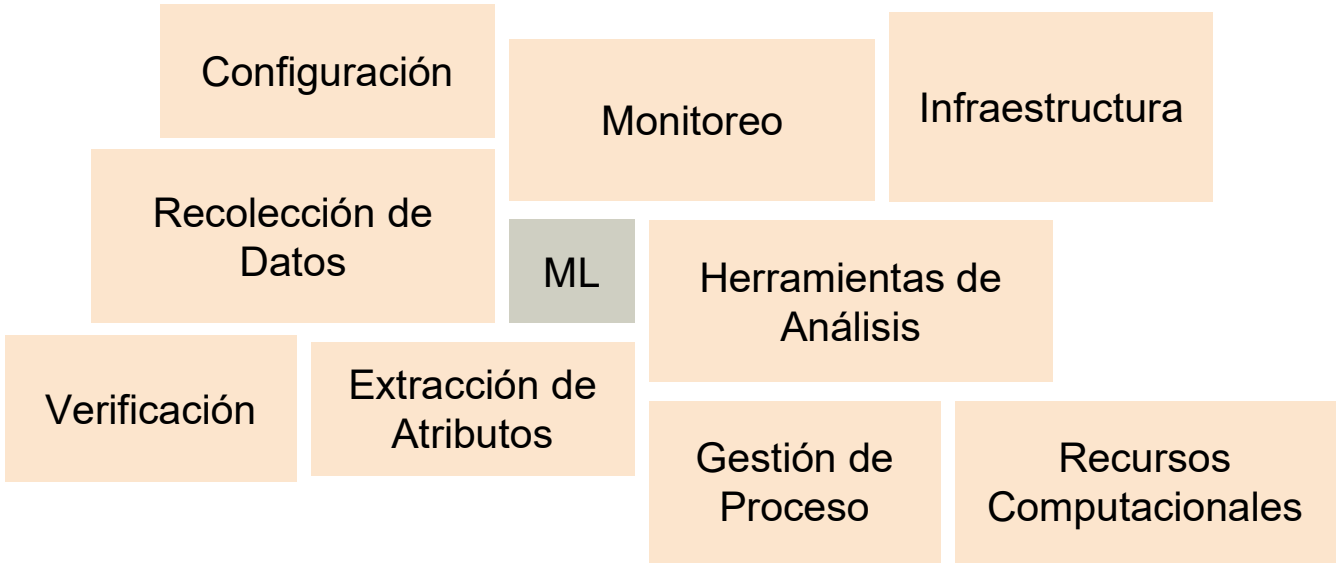


Producto de datos



https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf

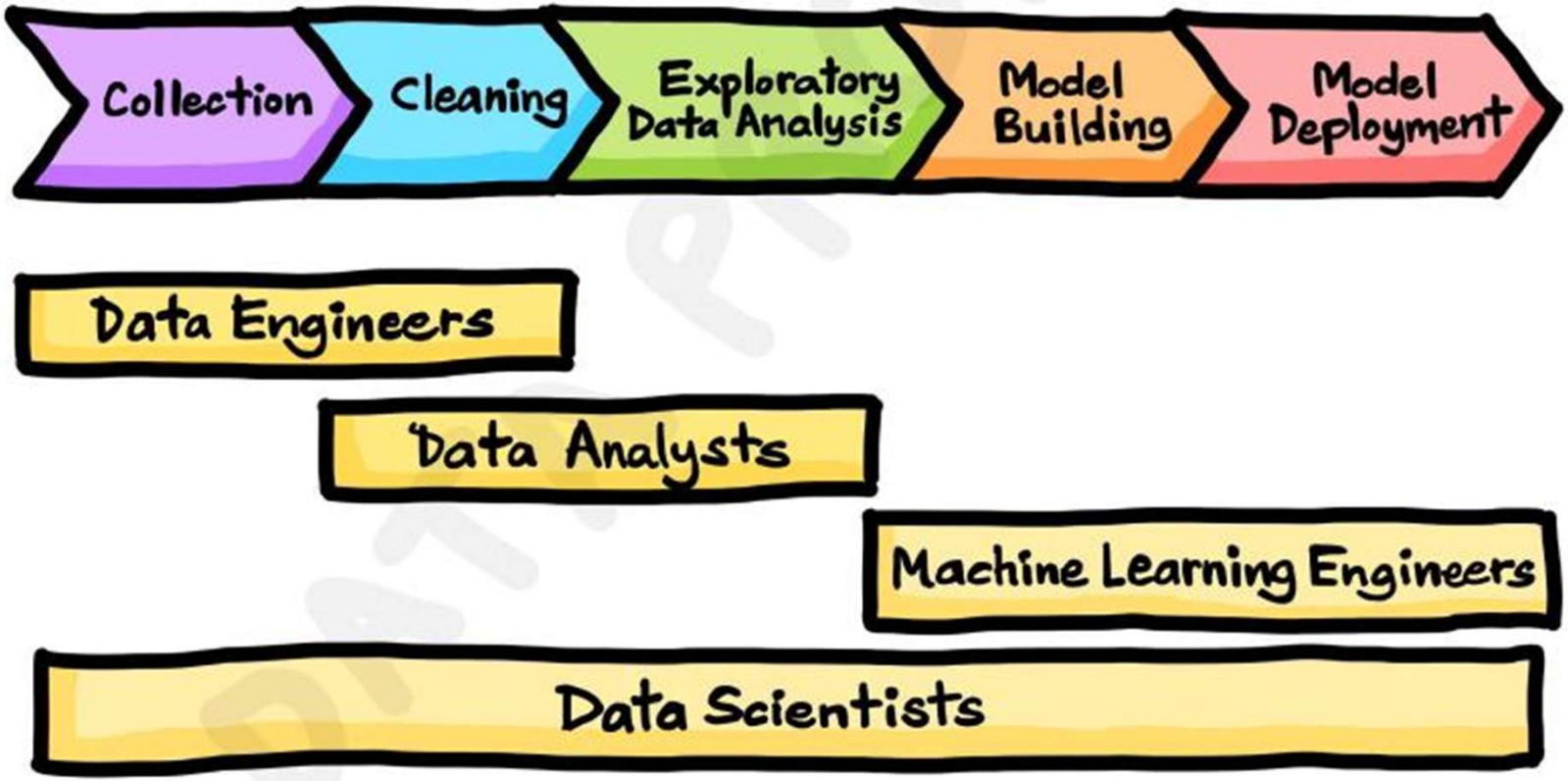
Producto de datos



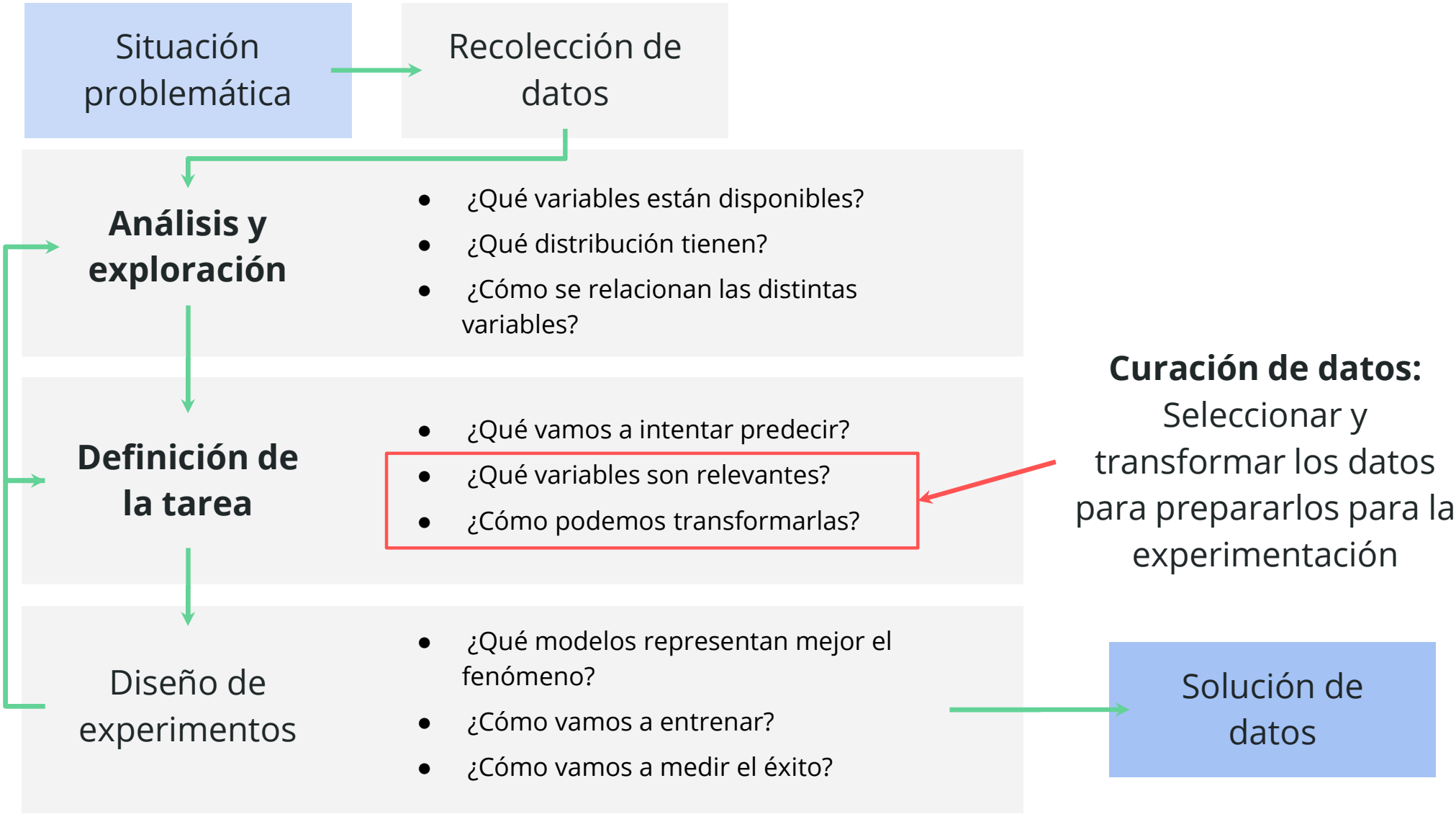
Tendencia:
Modelo de ML -> Commodity
Data pipelines y analisis -> ventaja competitiva

https://www.youtube.com/watch?v=Ndxpo4PeEms

Roles en ciencia de datos



https://www.youtube.com/dataprofessor


Situación problemática
Recolección de datos
Análisis y exploración
¿Qué variables están disponibles?
¿Qué distribución tienen?
¿Cómo se relacionan las distintas variables?
Definición de la tarea
¿Qué vamos a intentar predecir?
¿Qué variables son relevantes?
¿Cómo podemos transformarlas?
Diseño de experimentos
¿Qué modelos representan mejor el fenómeno?
¿Cómo vamos a entrenar?
¿Cómo vamos a medir el éxito?
Curación de datos: Seleccionar y transformar los datos para prepararlos para la experimentación
Solución de datos

Identificar el problema

- Cómo resolverlo con datos?
- Hablar con los especialistas y entender su relación con los datos
- Entender qué se sabe y qué no se sabe sobre el problema
- Si el problema todavía no fue abordado, ¿por qué?
- ¿Cuáles son los beneficios de trabajar en este problema y no en otro?
- ¿Quién se beneficia de que trabajemos en este problema y no en otro?
- ¿Existe realmente un problema?

Recolectar datos

- ¿Qué fuentes de datos conocen los especialistas?
- ¿Hay datos en Internet que puedan servir para resolver el problema?
- Evaluar la calidad de los datos
- ¿Quién es el propietario de los datos? ¿Qué quiere a cambio?
- Si los datos no existen, ¿cómo podemos conseguirlos?
- ¿Cuántos datos necesitamos? ¿Cuáles son los posibles sesgos de las fuentes de datos? Los datos, ¿son independientes entre sí? ¿cuál es el espacio muestral?

Exploración de datos

- Para decidir los procesos de curación, tenemos que entender nuestros datos **en conjunto**. Incluye:
    a. Todas las herramientas de análisis que hemos visto.
    b. Técnicas más complejas para el análisis de datos que permiten relacionar múltiples variables.
    c. Técnicas de visualización de datos no estructurados

Curación de datos

Problema	Datos	Decisiones de curación
Predecir los salarios de los programadores en Argentina en 2020	Encuesta voluntaria con columnas edad, género, años de experiencia y salario	• Eliminar edades menores que 18 y mayores que 99 • Eliminar salarios mayores que 1 millón de pesos • Estandarizar los años de experiencia de tal forma que la media sea 0. • Re-escalar las edades en un rango de 1 a 0, tal que 18 años o menos corresponda a 0 y 70 años o más corresponda a 1. • Eliminar la columna género.
Predecir el precio de una propiedad	Base de datos gubernamental con registros de transacciones inmobiliarias. Tiene precio, fecha y ubicación.	• Eliminar día y mes de la transacción. • *Escrapear* sitios de compra/venta para extraer información adicional sobre cada propiedad. • Imputar los valores faltantes utilizando estimaciones en base a ejemplos parecidos.

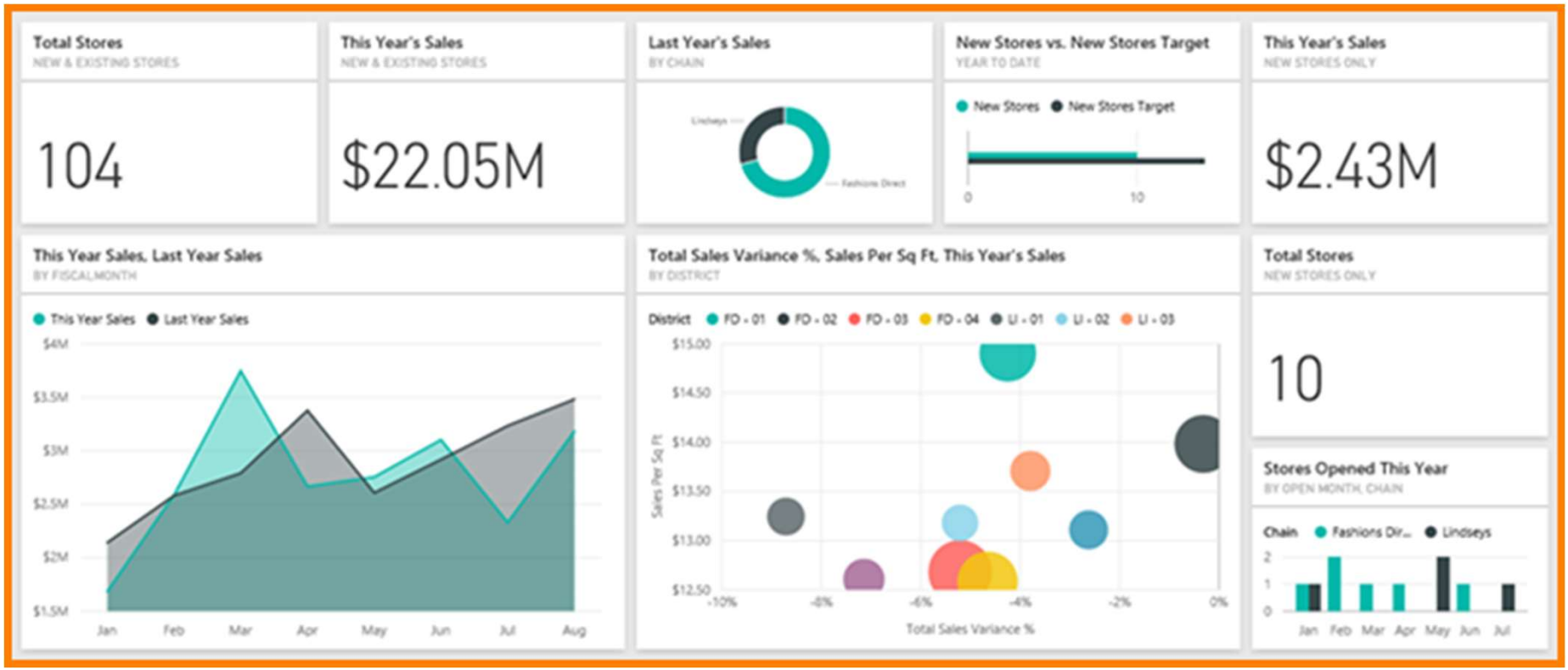
Desarrollar modelo

- ¿Tenemos una hipótesis? ¿O estamos explorando a ver qué encontramos?
- ¿Qué variables importantes podríamos no estar teniendo en cuenta?
- ¿Cuáles son nuestros propios sesgos a la hora de analizar los datos?
- Si el objetivo es elaborar un informe cuali- o cuantitativo, ¿están las conclusiones justificadas por nuestro análisis? ¿Estamos influenciados por las expectativas de quién va a leer el informe? ¿ocultamos resultados que contradicen las conclusiones?
- Si desarrollamos un modelo predictivo o de machine learning, ¿es el modelo trivial? ¿se basa el modelo únicamente en datos anteriores a lo que queremos predecir? ¿entendemos cómo funciona el modelo? ¿es ético?

Solución de datos: Visualizar y comunicar resultados

- ¿Entendemos a quién le estamos comunicando, cuáles son sus conocimientos e intereses y cómo difieren de los nuestros?
- Implementar un mínimo sentido de la estética y la economía para presentar los resultados
- ¿Somos aburridos y cómo podemos dejar de serlo?
- Estamos intentando convencer al otro de algo y por qué? ¿Estamos siendo honestos? ¿Estamos diciendo lo que el otro quiere escuchar?
- ¿Estamos distribuyendo correctamente el crédito por las contribuciones?
- ¿Entendemos el alcance de nuestro reporte?
- ¿Entendimos cómo respondió nuestra audiencia a lo que les dijimos?

Visualización de datos: Dashboards



Superset: Open Source https://superset.apache.org/

Perfil Analista de Datos

- Bases de datos consultas SQL
- Python, Pandas para análisis exploratorio, estadística descriptiva
- Dashboards y reportes: PowerBI, Tableau, Microstrategy, Superset
- Eventos Analytics: Google Analytics
- Inglés
- Git (hacer pull request)
- CURIOSIDAD!

Fuentes de datos

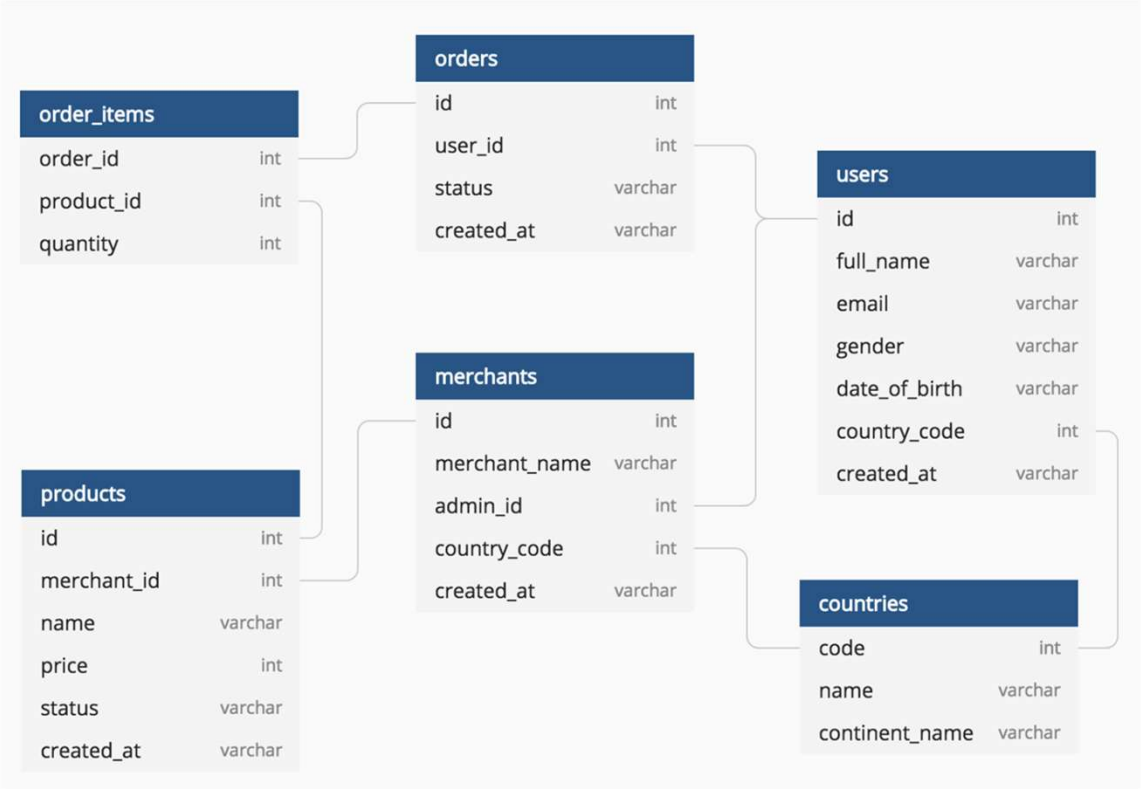
- Transaccionales: Compras, ventas, cotización de acciones en la bolsa, etc.
- Conversacionales: chats, mails, grabaciones telefónicas.
- Fotos y videos: subir a las redes sociales, etc.
- Sensores: posicion del gps, acelerómetro, etc.
- IoT: smart tv, smart watch, alexa, etc.
- registro de actividades: escuchar musica, leer un libro, buscar en google, comprar un artículo en un ecommerce (metadatos).

Estructura de los datos

- Llamamos “datos” a un conjuntos de registros.
- Cada registro tiene asociado un conjunto de características, y las características pueden relacionarse de manera compleja.
- Distintas estructuras suelen almacenarse con formatos de archivo particulares
    - La estructura de los datos no es lo mismo que el tipo de base de datos o archivos en los que se almacena

Datos estructurados

- Todos los registro tienen las mismas características con el mismo tipo
- Las características de algunos registros pueden ser registros en otra tabla



- Archivos en formato CSV, parquet, etc.
- Bases de datos relacionales como MySQL, Postgres

https://www.holistics.io/blog/top-5-free-database-diagram-design-tools/

Datos semi-estructurados

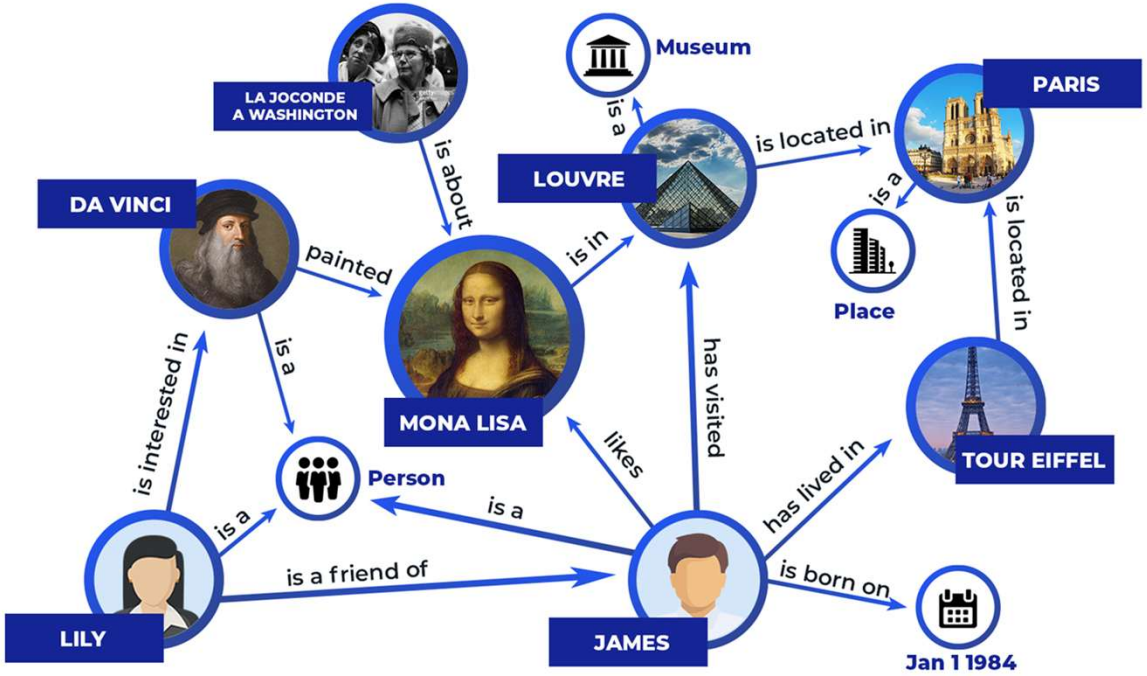
- Cada registro tiene un conjunto distinto de características
- Los registros pueden estar anidados


```
{"orders": [
  {
    "client_id": 1458,
    "items": [
     {"description": "Empanadas", "amount": 12},
     {"description": "Salsa picante", "amount": 1}
    ],
    "total": 950,
    "payment_method": "cash"
  },
  {
    "client_id": 985,
    "items": [
     {"description": "Lomito Completo", "amount": 2,
      "observations": "Uno sin huevo"}
    ],
    "total": 1400,
    "payment_method": "debit",
    "debit_card": "Maestro"
  }
 ]}
```


- Archivos en formato JSON
- Bases de datos no relacionales como MongoDB

Datos semi-estructurados

- Los registros pueden tener relaciones complejas
  - Jerarquías
  - Estructura de grafo (Twitter)



- Triplas RDF
- Graph-oriented databases

https://www.komunicando.es/wp-content/uploads/2017/12/knowledge-graph.jpg

Datos no estructurados

- Colecciones de distintos tipos:
  - Documentos de texto
  - Imágenes
  - Audio
- Pueden o no tener metadatos asociados

Calidad de datos

- **Completitud** — Tenemos toda la información necesaria
- **Validez** — Es el tipo de datos correcto
- **Precisión** — Los datos reflejan precisamente los objetos que representan
- **Integridad** — Las relaciones entre las entidades representadas son consistentes
- **Consistencia** — Hay duplicados o inconsistencias dentro del sistema?
- **Temporalidad** — Los datos están disponibles en el momento en el que se los requiere?

Formatos de Datos

- Tabulares: como una planilla, con filas y columnas.
  - Formatos de Archivos: CSV, TSV, XLS
  - Estructura de Datos: Dataframe
- Jerárquicos: con valores anidados dentro de otros valores.
  - Formatos de Archivos: JSON, XML
  - Estructura de Datos: Lista de Objetos
- Crudos: sin estructura específica
  - Formato de Archivos: TXT
  - Estructura de Datos: String

Formatos: Tabulares vs Jerárquicos vs Crudos

	Sepal.Length	Sepal.Width
1	5.1	3.5
2	4.9	3.0
3	4.7	3.2

Tabular Data

↳Name: Robin
  ↳ Species: Hedgehog
  ↳ Owner: Justice Smith
    ↳ Address: 1234 Main St.
    ↳ Phone #: 123-4567
↳Name: Bunny
  ↳ Species: Rabbit
    ↳ Breed: Holland Lop
  ↳ Color: Brown and white

Hierarchical Data

石室诗士施氏，嗜狮，誓食十狮。氏时时适市视狮。十时，适十狮适市。是时，适施氏适市。氏视是十狮，恃矢势，使是十狮逝世。氏拾是十狮尸，适石室。石室湿，氏使侍拭石室。石室拭，氏始试食是十狮尸。食时，始识是十狮，实十石狮尸。试释是事。

Raw Text

CSV - Comma Separated Values

- Archivos de texto delimitado que usa coma para separar valores.
- Cada línea es un registro con uno o más campos.
- No está formalmente especificado!

```
latitud,longitud,Nombre
-54.832543,-68.3712885,SAN SEBASTIAN  ( USHUAIA )
-54.8249379,-68.3258626,AERO PUBLICO DE USHUAIA
-54.8096728,-68.3114748,PUERTO USHUAIA (PREFECTURA)
-54.8019121,-68.3029511,PUERTO USHUAIA
-51.6896359,-72.2993574,PASO LAURITA CASAS VIEJAS
-51.5866042,-72.3649779,PASO DOROTEA
-51.2544488,-72.2652242,PASO RIO DON GUILLERMO
-53.3229179,-68.6063227,PASO SAN SEBASTIAN
-53.78438,-67.7173342,TERMINAL RIO GRANDE
```

https://tools.ietf.org/html/rfc4180

Lectura de CSV

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html

```
In [1]: data = 'col1,col2,col3\na,b,1\na,b,2\nc,d,3'
In [2]: pd.read_csv(StringIO(data))
Out[2]:
```

	col1	col2	col3
0	a	b	1
1	a	b	2
2	c	d	3

Lectura de JSON

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_json.html

```
In: data = [{'state': 'Florida',
             'shortname': 'FL',
             'info': {
                 'governor': 'Rick Scott'
             },
             'counties': [{'name': 'Dade', 'population': 12345},
                          {'name': 'Broward', 'population': 40000},
                          {'name': 'Palm Beach', 'population': 60000}]},
            {'state': 'Ohio',
             'shortname': 'OH',
             'info': {
                 'governor': 'John Kasich'
             },
             'counties': [{'name': 'Summit', 'population': 1234},
                          {'name': 'Cuyahoga', 'population': 1337}]}]
```

```
In: from pandas.io.json import json_normalize

In: json_normalize(data, 'counties', ['state', 'shortname',
['info', 'governor']])
```

Out:

	name	population	state	shortname	info.governor
0	Dade	12345	Florida	FL	Rick Scott
1	Broward	40000	Florida	FL	Rick Scott
2	Palm Beach	60000	Florida	FL	Rick Scott
3	Summit	1234	Ohio	OH	John Kasich
4	Cuyahoga	1337	Ohio	OH	John Kasich

Bases de Datos

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

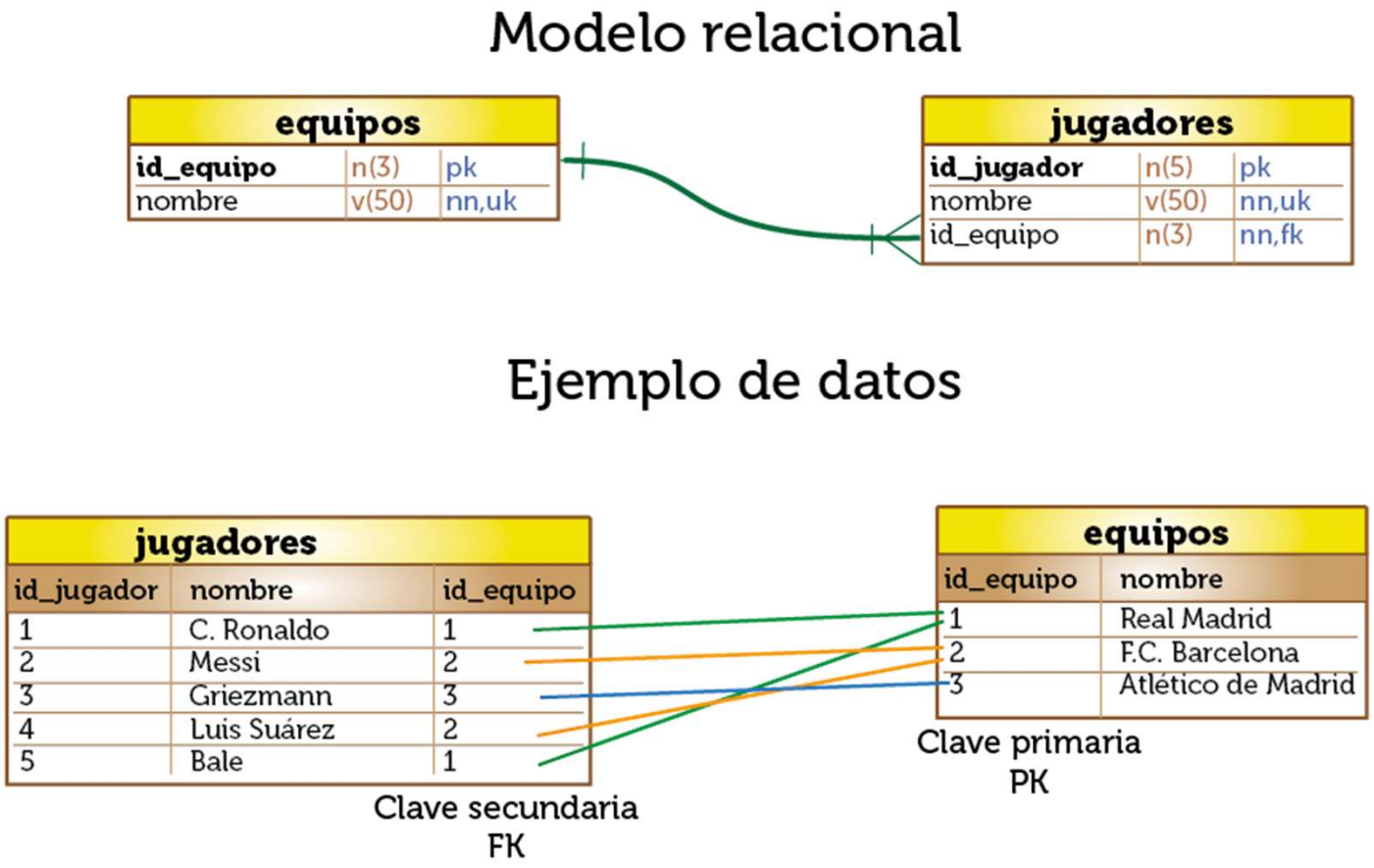
Clasificadas según

- Variabilidad de los datos: estáticas o dinámicas
- Contenidos: bibliográficas, texto completo, directorios, etc
- Modelo de administración: [no] transaccionales, [no] relacionales, distribuidas, etc

Bases de Datos Relacionales

- Basada en tuplas (listas ordenadas de elementos).
- Se compone de varias tablas (relaciones)
- Cada tabla es un conjunto de campos (columnas) y registros (filas)
- Se establecen relaciones entre tablas usando claves
- Permiten combinar columnas de una o más tablas (JOIN)

Modelo Relacional: Ejemplo



ACID

ACID - **A**tomicity, **C**onsistency, **I**solation, **D**urability

Atomicidad: Serie de operaciones indivisible e irreductible que ocurren todas juntas o ninguna.

Consistencia: Cualquier transacción que comienza en el futuro verá los efectos de las transacciones que ocurrieron en el pasado.

Aislamiento: Visibilidad de la integridad de una transacción para otros usuarios y sistemas (manejo de concurrencia).

Durabilidad: transacciones completadas sobrevivirán de manera permanente.

Bases de Datos No Relacionales

- No permiten JOIN
- No intentan garantizar ACID
- Escalan horizontalmente (agregación de 2 o más instancias de DB)

Bases de Datos Documentales

- Almacenan, Recuperan y Gestionan “documentos” (datos estructurados, pero no necesariamente todos iguales)
- Son más fáciles de extender: se pueden agregar o quitar campos a los documentos. No tienen la idea de “campos vacíos”.

```
{
    _id: 1234,
        Nombre:"Pepe",
    Dirección:"Plaza Mayor 5",
    Profesión:"Panadero"
}
```

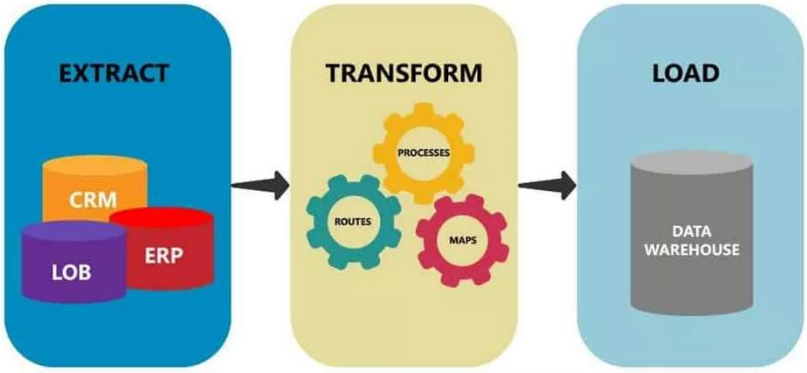
```
{
    _id: 4567,
        Nombre:"Juan",
    Dirección:"Vía Azul 13",
    Hijos:[
        {Nombre: “Miguel”, Edad: 3},
        {Nombre: “Sara”, Edad: 5},
    ]
}
```

Demo notebook

EL460_01 Exploracion.ipynb

ETL

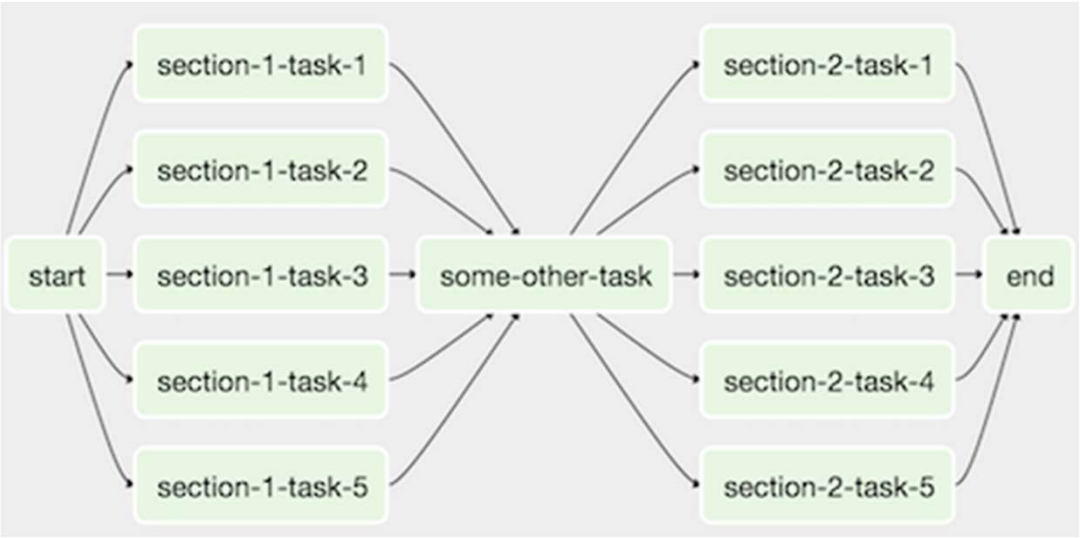
- **Extracción**:
  - Extraer datos heterogéneos de las distintas fuentes provistas.
- **Transformación**:
  - Normalizar
  - Elegir solo las partes relevantes
  - Inter relacionar los datos de las distintas fuentes
- **Load** (carga):
  - Agregar toda estos datos al Data Warehouse
- Actualmente se está usando en otro orden: **ELT**



ETL - Extract, Transform, Load

Orquestar ETLs: Airflow

- Herramienta Open Source en Python
- Muy útil para cuando tenemos que manejar muchos ETLs en paralelo
- https://airflow.apache.org/docs/apache-airflow/stable/tutorial.html

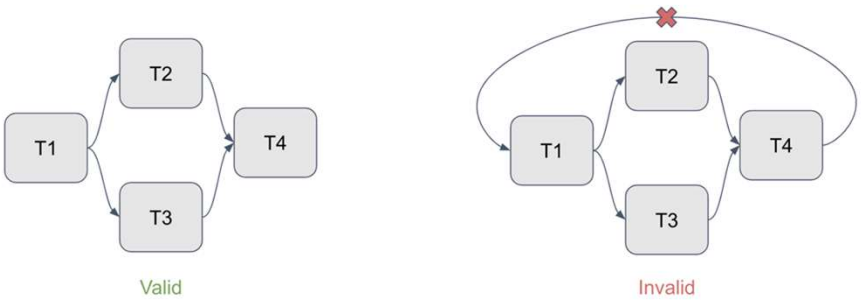


Apache Airflow

Airflow Conceptos principales

DAG
Un grafo acíclico dirigido, o DAG, es un data pipeline definido en código Python.
Cada DAG representa una colección de tareas que desea ejecutar y está organizado para mostrar las relaciones entre las tareas en la interfaz de usuario de Airflow.

Propiedades:
Dirigida: si existen varias tareas con dependencias, cada una debe tener al menos una tarea definida en sentido ascendente o descendente.
Acíclico: las tareas no pueden crear datos que vayan a la autorreferencia. Esto es para evitar crear bucles infinitos.
Grafo: todas las tareas se presentan en una estructura clara con procesos que ocurren en puntos claros con relaciones establecidas con otras tareas.

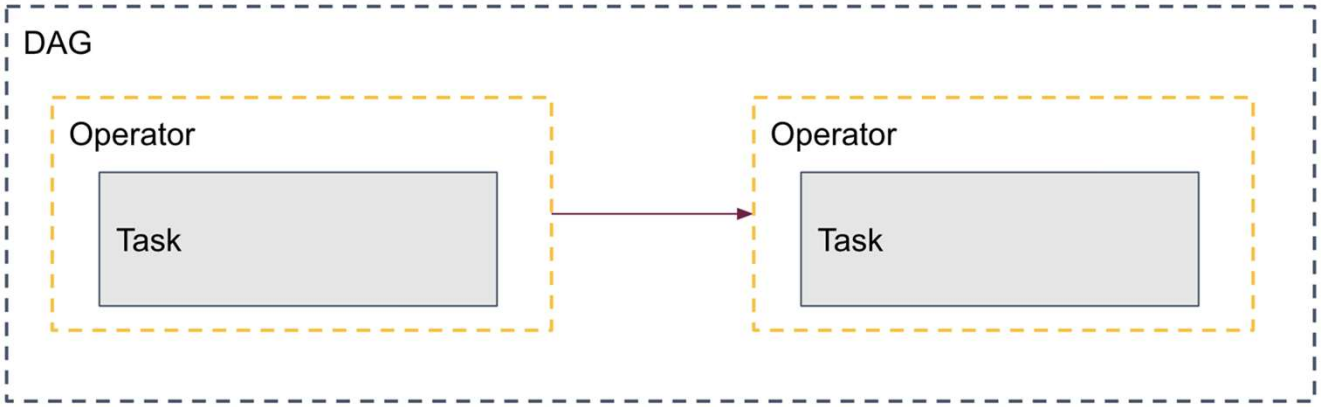


Airflow Conceptos principales

Task
Representa cada nodo de un DAG definido. Son representaciones visuales del trabajo que se realiza en cada paso del flujo de trabajo, y los operadores definen el trabajo real que representan.

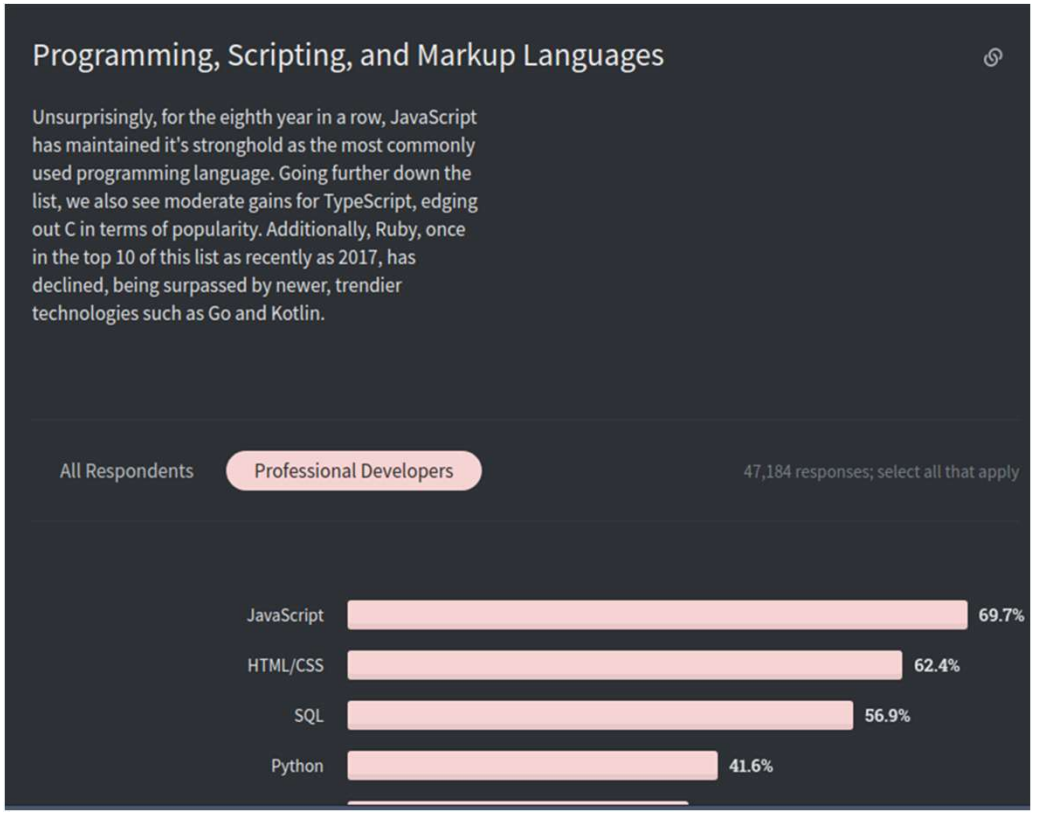
Operator
son los componentes básicos de Airflow y determinan el trabajo real que se realiza. Se pueden considerar como un envoltorio de una sola tarea, o nodo de un DAG, que define cómo se ejecutará esa tarea. Los DAG se aseguran de que los operadores se programen y ejecuten en un orden determinado, mientras que los operadores definen el trabajo que se debe realizar en cada paso del proceso.

https://www.astronomer.io/guides/intro-to-airflow/



Por qué SQL?

- Es el lenguaje más común para acceder a bases de datos relacionales.
- Es la interfaz de los warehouse, data lakes y sistemas de almacenamiento distribuido.
- Es una habilidad que deben tener para ser *data engineers*.
- 3ro en encuesta de popularidad de Stack Overflow 2020



Why You Can't Do All of Your Data Engineering with SQL
A Beginner's Guide to Data Engineering — Part I
https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-professional-developers

Por qué SQLite?

- Es muy pequeña y su instalación es sencilla
- Es ideal para desarrollo: no tiene dependencias externas
- Es ideal para aprender SQL
- Es la DB más desplegada (celulares, Skype, iTunes, TVs, autos, etc)

SQLite Home Page

Consultas - Sintaxis General

SELECT DISTINCT *column_list* -- selecciona una lista de columnas opcionalmente, filas únicas

FROM *table_list* -- de una lista de tablas (*table_list*)

JOIN *table* **ON** *join_condition* -- combinada con la tabla (*table*) si se cumple la condición (*join_condition*)

WHERE *row_filter* -- filtrando solo las que cumplen un criterio (*row_filter*)

ORDER BY *column* -- ordenadas por los valores de cierta columna (*column*)

LIMIT *count* **OFFSET** *offset* -- limitada a un número de registros (*count*) a partir del registro (*offset*)

GROUP BY *column* -- agrupadas por el valor de la columna (*column*)

HAVING *group_filter;* -- solo aquellos grupos que cumplen la condición (*group_filter*)

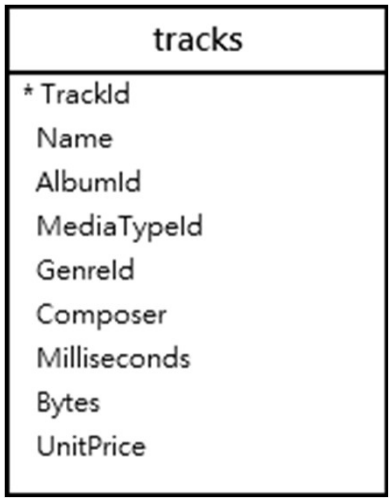
Consultas - SELECT / FROM

Para realizar una consulta, usamos SELECT y debemos indicar

- la tabla de la cual obtener los registros usando la cláusula FROM
- la lista de columnas separada por comas

Ejemplo: Canciones y su compositor

```
SELECT name, composer
FROM tracks
```



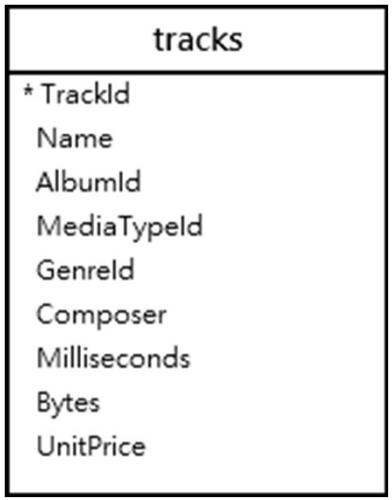
Consultas - ORDER BY ordenamiento

Para ordenar los resultados de una consulta, agregamos ORDER BY

- ASC para ordenamiento ascendente
- DESC para ordenamiento descendente

Ejemplo: Canciones y su compositor

```
SELECT name, composer
FROM tracks
ORDER BY
  composer ASC
  name DESC;
```



Consultas - DISTINCT remoción de duplicados

Para filtrar los resultados de una consulta, agregamos la palabra clave DISTINCT

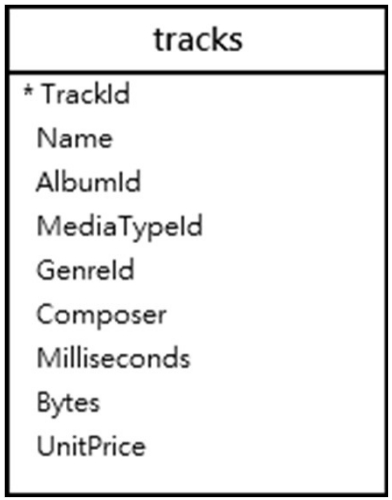
Ejemplo: Ciudades y países de los clientes

```
SELECT DISTINCT city, country
FROM customers
ORDER BY
  country ASC,
  city ASC;
```

Consultas - WHERE filtrado de resultados

Para filtrar los resultados que cumplen un determinado criterio se usa la cláusula WHERE con diferentes operadores

- =, <>, <, >, <=, >=
- IN (....), BETWEEN x AND y
- LIKE
- ....



Ejemplo: Canciones con compositor de nombre SMITH

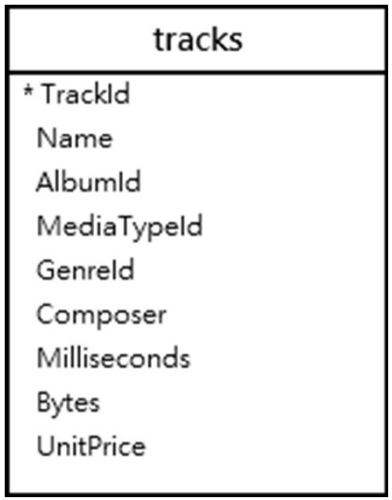
```
SELECT name, composer
FROM tracks
WHERE composer LIKE '%SMITH%'
```

Consultas - LIMIT/OFFSET limitar cantidad

Para restringir la cantidad de resultados en una consulta, se usa la cláusula LIMIT con un número. Adicionalmente, se puede incluir la cláusula OFFSET para indicar a partir de qué registro se obtienen los resultados.

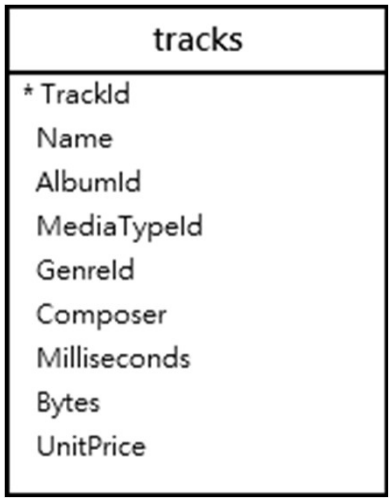
Ejemplo: Canciones 31 a 40 ordenadas por título.

```
SELECT name
FROM tracks
ORDER BY name
LIMIT 10 OFFSET 30
```



Consultas - GROUP BY agrupar resultados

Para agrupar los resultados, usamos la cláusula GROUP BY. Retorna un resumen de las filas agrupadas por el valor de una o más columnas. En cada grupo podemos aplicar una función de agregación: MAX, MIN, SUM, COUNT, AVG.



Ejemplo: Los 10 compositores más prolíficos y cantidad de canciones

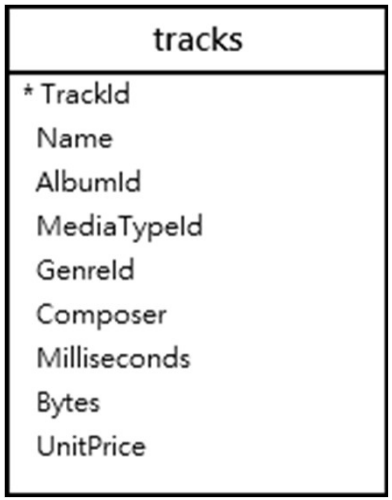
```
SELECT composer, count(trackid)
FROM tracks
WHERE composer <> '' -- filtrar las que no tienen compositor
GROUP BY composer
ORDER BY count(trackid) desc
LIMIT 10
```

Consultas - HAVING agrupar con condición

También podemos hacer que GROUP BY restrinja los grupos a aquellos que cumplen una condición utilizando la cláusula HAVING.

Ejemplo: Los compositores que escribieron más de 30 canciones

```
SELECT composer, count(trackid) AS cant -- un alias
FROM tracks
WHERE composer <> '' -- filtrar las que no tienen compositor
GROUP BY composer
HAVING cant > 30
ORDER BY cant desc
LIMIT 10
```



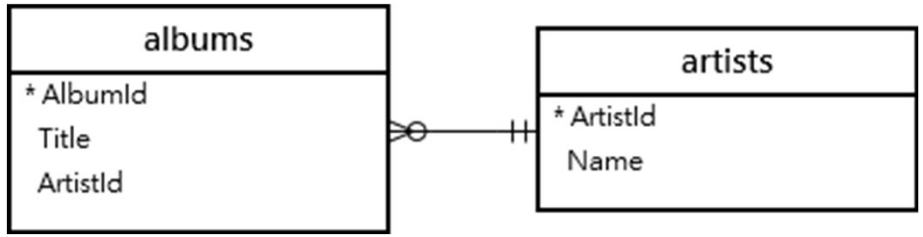
Consultas - JOIN

Cuando tenemos una relación entre tablas, podemos combinar sus campos con la cláusula JOIN.

JOIN busca las coincidencias entre los campos de las tablas tal como se indica con la palabra ON.

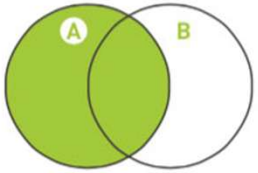
Ejemplo: Álbums y sus artistas

```
SELECT title, name
FROM albums
INNER JOIN artists
  ON artists.artistId = albums.artistId
```

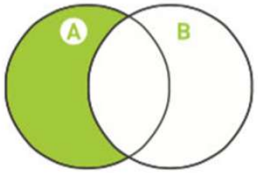


SQL JOINS

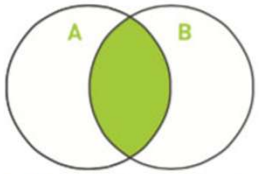
A CHEATSHEET BY WEBDEZIGN.CO.UK



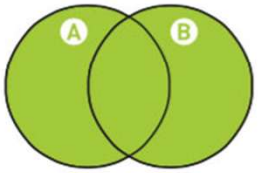

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



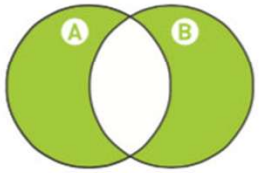

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



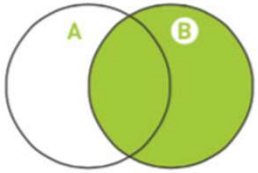

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



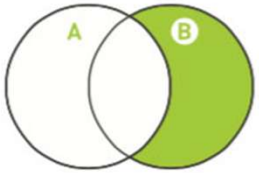

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



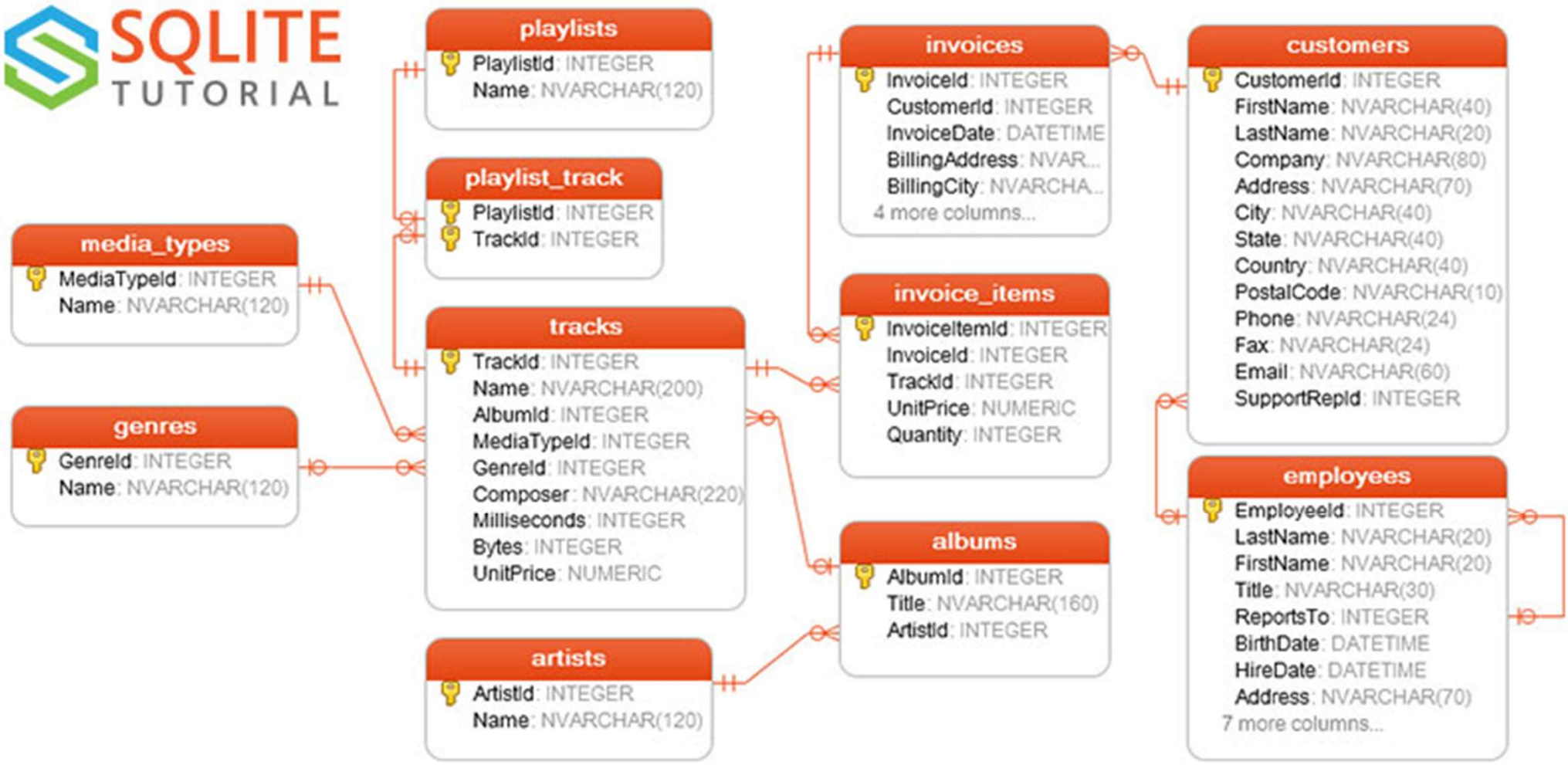

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```


SQLITE
TUTORIAL
playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)
playlist_track
PlaylistId: INTEGER
TrackId: INTEGER
media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)
genres
GenreId: INTEGER
Name: NVARCHAR(120)
tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC
artists
ArtistId: INTEGER
Name: NVARCHAR(120)
albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER
invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...
invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER
customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER
employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

Notebook SQL

EL460_02 SQL

Agrupamiento y agregación

1. groupby:
   - Tomar una serie de columnas A, B, C
   - Por cada combinación de valores de las columnas (a1, b1, c1), agrupar las filas que tengan esos valores.
2. agg:
   - Toma una función f
   - Por cada grupo de filas, aplicar la función f a cada columna.

Agrupamiento y agregación

df.groupby('species').agg('sum')

	species	sepal_length	sepal_width	petal_length	petal_width
0	setosa	5.1	3.5	1.4	0.2
1	setosa	4.9	3.0	1.4	0.2
2	setosa	4.7	3.2	1.3	0.2
3	setosa	4.6	3.1	1.5	0.2
4	setosa	5.0	3.6	1.4	0.2
50	versicolor	7.0	3.2	4.7	1.4
51	versicolor	6.4	3.2	4.5	1.5
52	versicolor	6.9	3.1	4.9	1.5
53	versicolor	5.5	2.3	4.0	1.3
54	versicolor	6.5	2.8	4.6	1.5
100	virginica	6.3	3.3	6.0	2.5
101	virginica	5.8	2.7	5.1	1.9
102	virginica	7.1	3.0	5.9	2.1
103	virginica	6.3	2.9	5.6	1.8
104	virginica	6.5	3.0	5.8	2.2

SUM

SUM

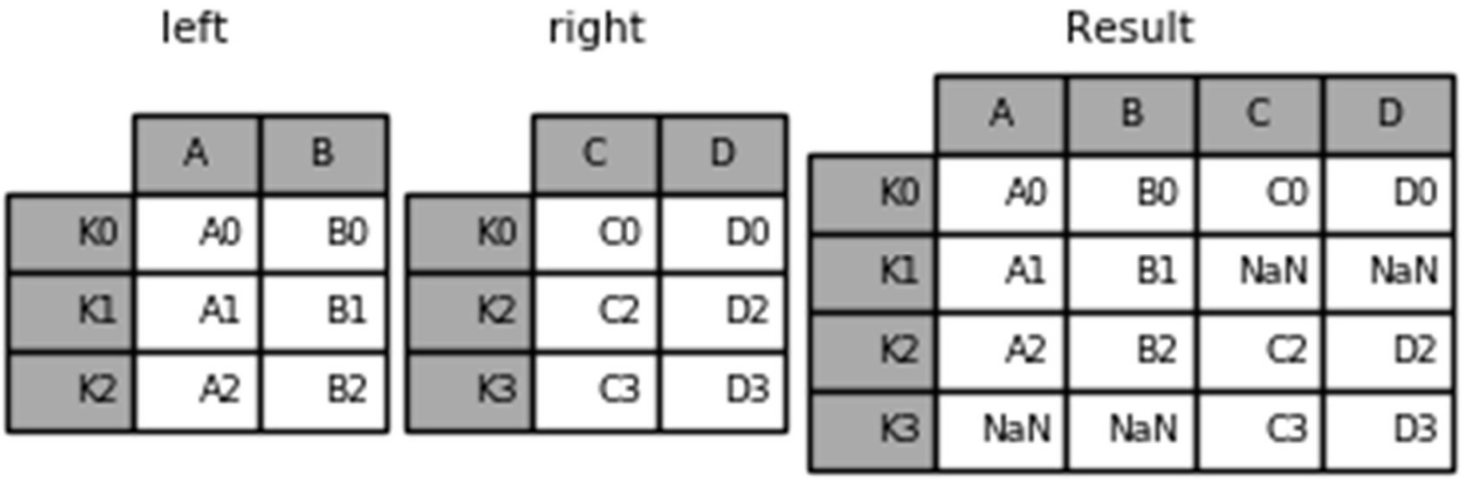
SUM

species	sepal_length	sepal_width	petal_length	petal_width
setosa	24.3	16.4	7.0	1.0
versicolor	32.3	14.6	22.7	7.2
virginica	32.0	14.9	28.4	10.5

https://towardsdatascience.com/how-to-use-the-split-apply-combine-strategy-in-pandas-groupby-29e0eb44b62e

Join y merge

1. df1.join(df2, how='outer')
    - Une horizontalmente los DataFrames y hace coincidir las filas donde el valor del índice es el mismo



	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

	C	D
K0	C0	D0
K2	C2	D2
K3	C3	D3

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2
K3	NaN	NaN	C3	D3

Join y merge

1. df1.merge(df2, on='key')
    - Igual al join, pero en lugar de comparar los índices, compara un conjunto de columnas.

left

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

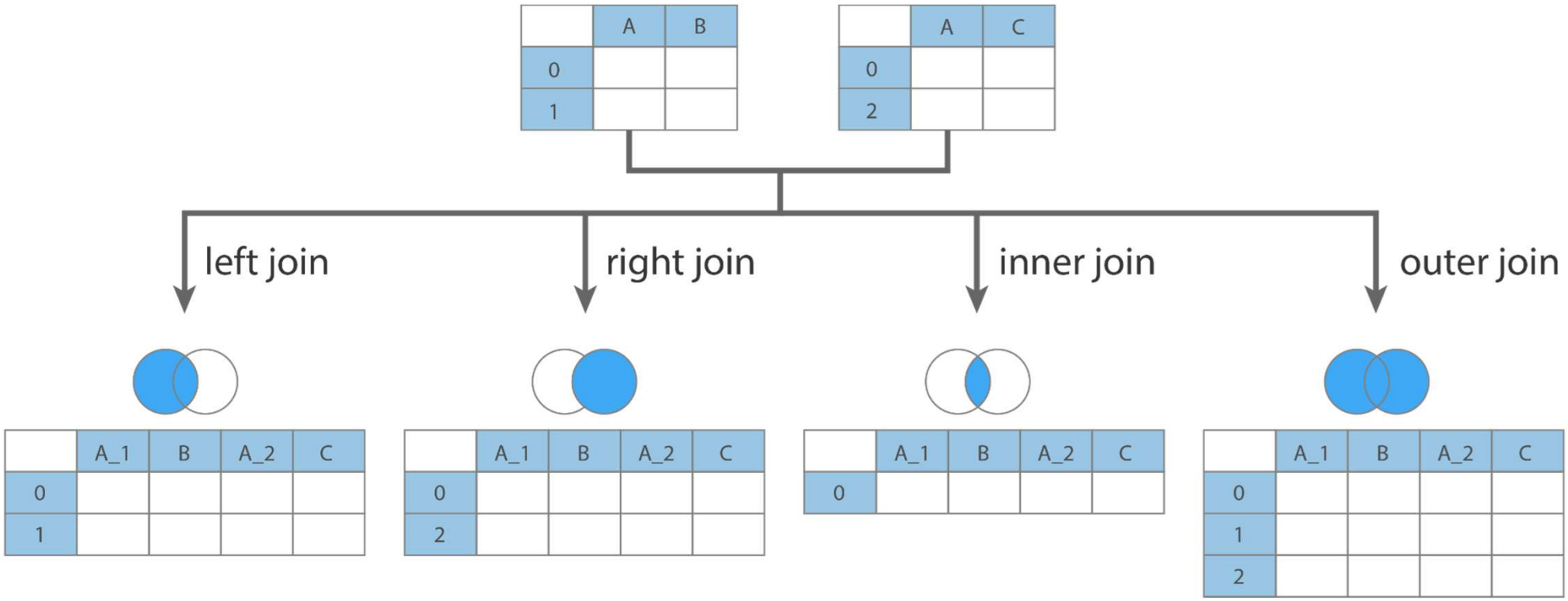
right

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

Result

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

Join y merge



https://towardsdatascience.com/python-pandas-dataframe-join-merge-and-concatenate-84985c29ef78

Notebook Combinación de datasets

Pandas Vs SQL

https://pandas.pydata.org/docs/getting_started/comparison/comparison_with_sql.html

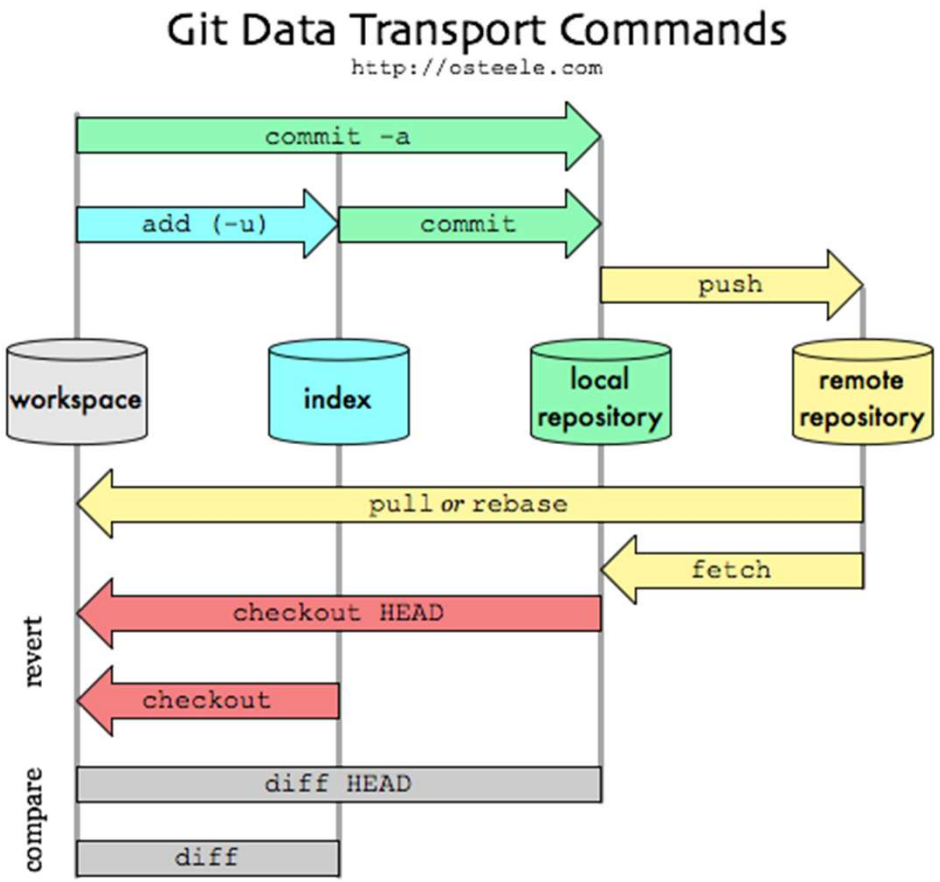
https://towardsdatascience.com/pandas-vs-sql-compared-with-examples-3f14db65c06f

Gestión de Código: git

Sistema de control de versiones para seguimiento de cambios en archivos de computadora y coordinación de trabajo entre múltiples personas.

Es distribuido y tiene como objetivos la velocidad, la integridad de datos y flujos de trabajo distribuidos y no lineales.

Git Resumen de Comandos



Material Complementario

- Tutorial de SQLite (incluye consola para aprender online)
  - https://www.sqlitetutorial.net/tryit/
- Soporte de SQL en pandas
  - Equivalencia entre métodos de pandas y sintaxis de SQL
    https://pandas.pydata.org/docs/getting_started/comparison/comparison_with_sql.html
  - Cargar y guardar datos en SQL
    https://pandas.pydata.org/docs/user_guide/io.html?highlight=read_sql#sql-queries
- SQL
  - https://www.sqlhabit.com/
- Airflow
  - https://www.youtube.com/watch?v=IH1-0hwFZRQ
  - https://www.udemy.com/course/the-ultimate-hands-on-course-to-master-apache-airflow/

Entregable parte 1