Exploración y Curación de Datos

Ariel Wolfmann
Mayo 2024

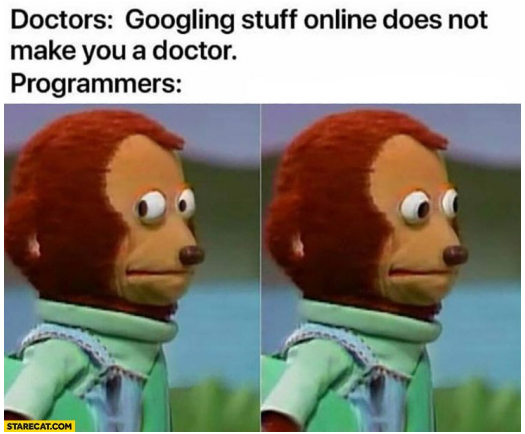
Sobre mi

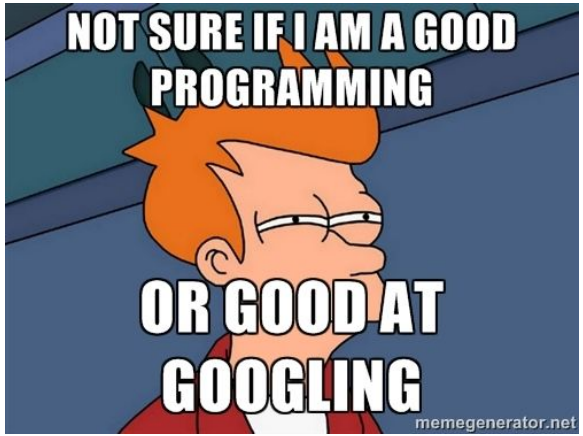
- Data & Software Engineering
- Ecosistema de startups (Rappi, Addi)
- Cs de la Computación, Famaf
- co-organizador PyData Córdoba

PyData

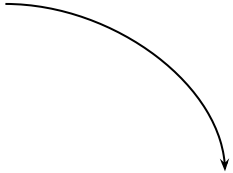
Programar vs Buscar/Googlear

- Aprender a Googlear con la abstracción suficiente
  - No reinventar la rueda
  - Prompt Engineering ChatGPT
  - LLMs te pueden resolver una gran parte pero no el 100% (aun).
- Stack Overflow
- Entender mensajes de error
- Leer documentacion, ej: https://pandas.pydata.org/docs/



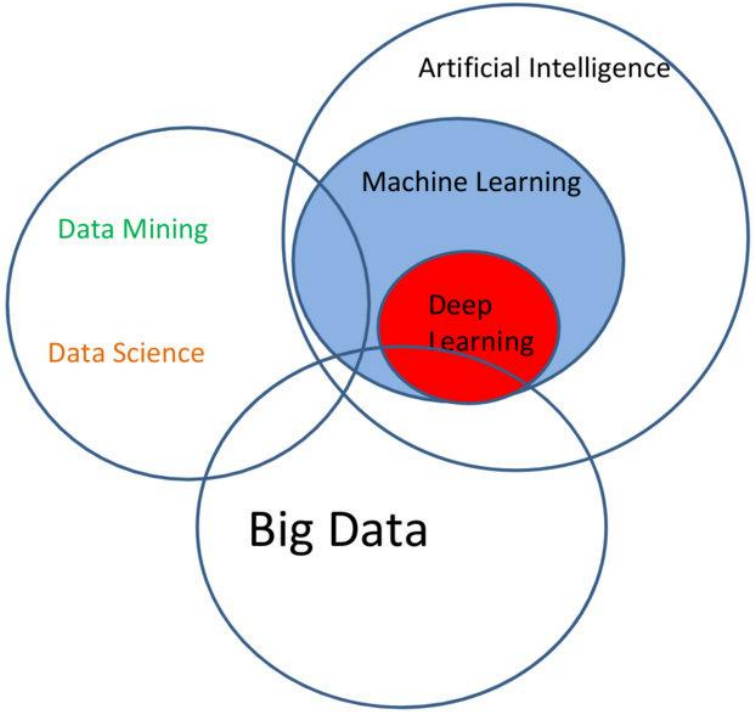


Analytics, Data Science,
Machine Learning,
Inteligencia Artificial



¿Cómo los definirían? ¿Son sinónimos?
¿Cuáles son las diferencias?

Conceptos relacionados



Conceptos Relacionados

- **Analytics:** Análisis de datos para obtener insights (revelar info util).
- **Inteligencia Artificial**: Computadoras emulando comportamientos humanos específicos.
- **Big Data**: Disponibilidad de grandes volúmenes de datos y la capacidad de procesarlos.
- **Machine Learning**: Predecir un comportamiento basado en métodos estadísticos, encontrando y aprendiendo 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**: Recolección, limpieza, transformar datos e hipótesis en información para la toma de decisiones o para predecir acciones.
- **NLP**: Procesamiento de lenguaje natural

Conceptos - Ecommerce que quiere mejorar sus ventas

- **Analytics**: métricas KPIs, la cant de visitas diarias, conversión, ticket prom.
- **Data science:** recopilar datos comportamiento de usuarios (**big data**), analisis exploratorio para identificar patrones de consumo.Mejorar carrito abandonado
- **Machine Learning:** Predecir abandono de compra, entrenar modelo usando datos históricos -> activar estrategias como descuentos, recordatorios, etc.
- **Inteligencia artificial**: chatbot servicio al cliente, usando modelos **NLP**, recomendar productos similares.
- Business Intelligence: armar tablero de reporte de las métricas,
    - como mejoro el experimento de carrito abandonado?
    - interpretar resultados

Ingeniería de datos

- Recolección, almacenamiento y procesamiento de datos.
  - Infraestructura de datos
- Calidad de datos: datos precisos y confiables para ofrecer resultados útiles.
- Escalabilidad: manejar la carga de trabajo
- Seguridad y privacidad: manejar datos sensibles y confidenciales. Ej GDPR.

Data Product

Producto que para lograr su objetivo lo hace a través del uso de datos.

Sin datos no sería posible

Data Product

“Los datos son el nuevo petróleo”

- Recolectarlos a gran escala no es fácil.
- Necesitan ser **transformados** para tener valor.
- El petróleo se usa una sola vez, en cambio los datos son persistentes, se pueden **volver a utilizar.**
- Los datos son como el agua: son esenciales, necesitan ser **limpiados** y **accesibles** para todos
    - *Democratizar el acceso a datos, deben ser interpretables.*
    - *Calidad de datos.*

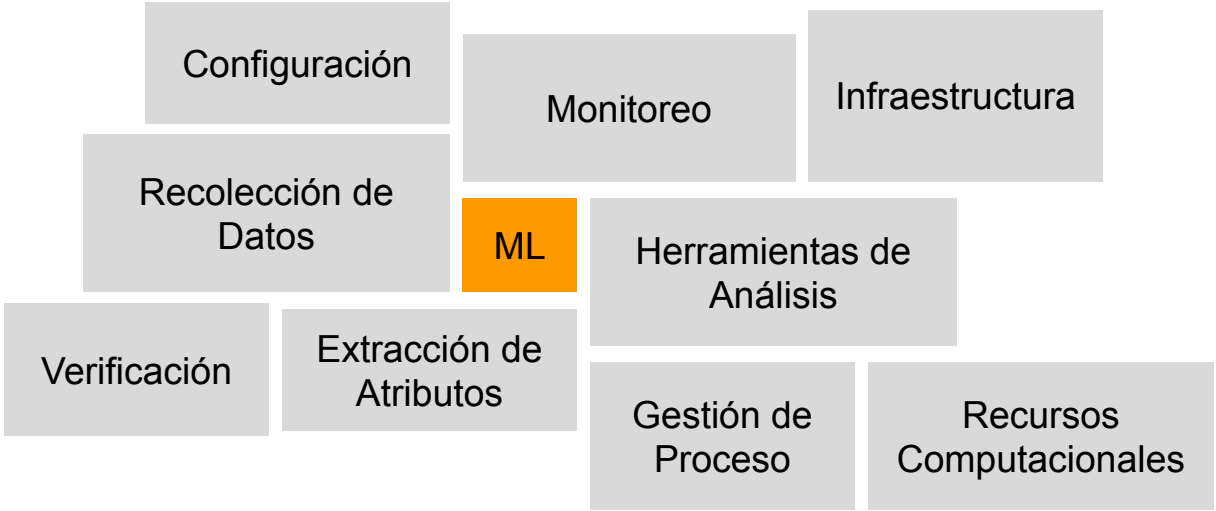
Data Product - *Desafíos*

Product - Market fit: Desafío de producto tecnológico tradicional

DATOS - Un producto basado en datos tiene a su vez otra fuente de **incertidumbre**:

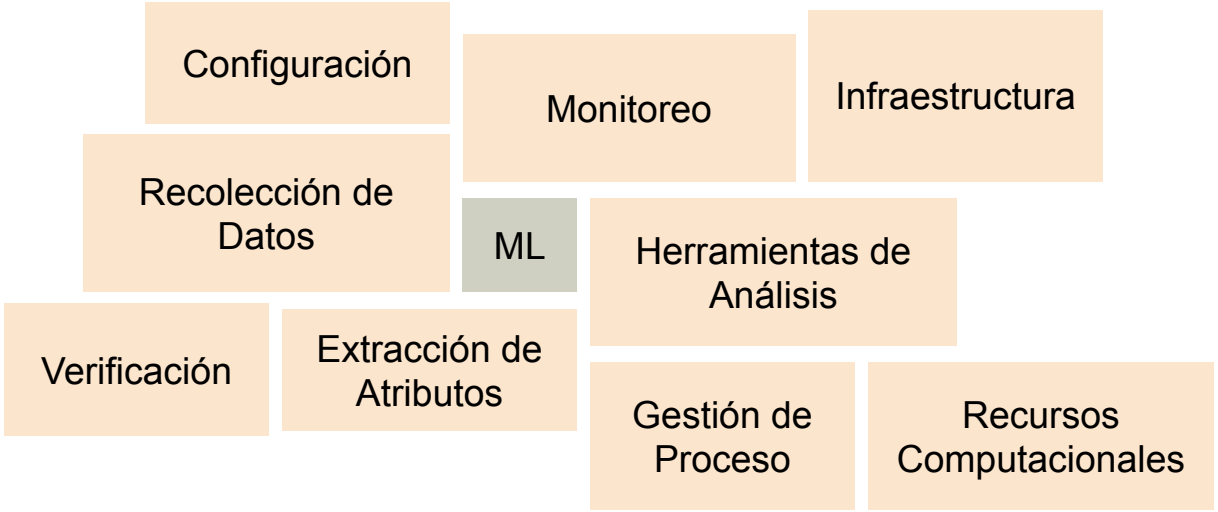
- La predicciones nunca van a ser 100% correctas
- Es muy difícil garantizar la performance de un modelo sin desarrollarlo.
  - Quizás no haya datos o señal suficiente.
- La performance de un modelo en desarrollo puede variar significativamente al ponerlo productivo.

Producto de datos



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

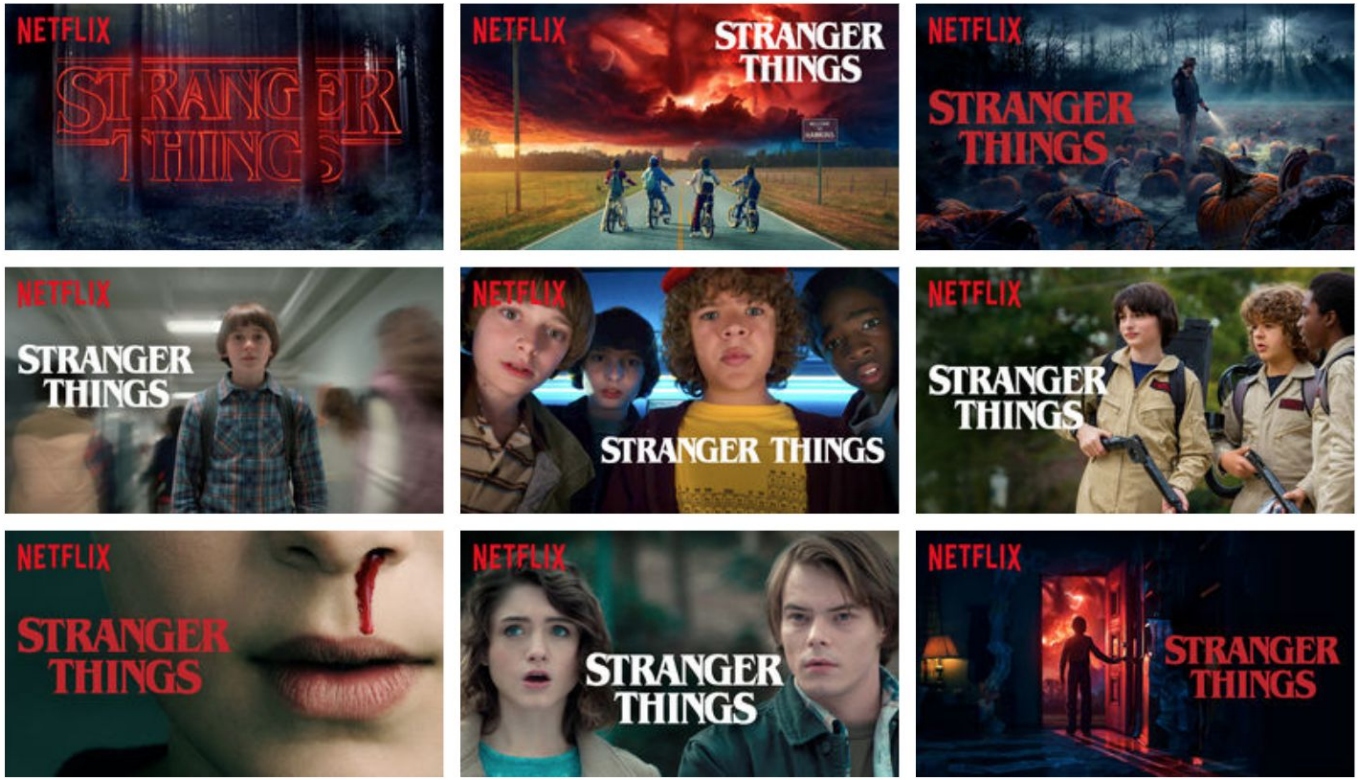
Producto de datos



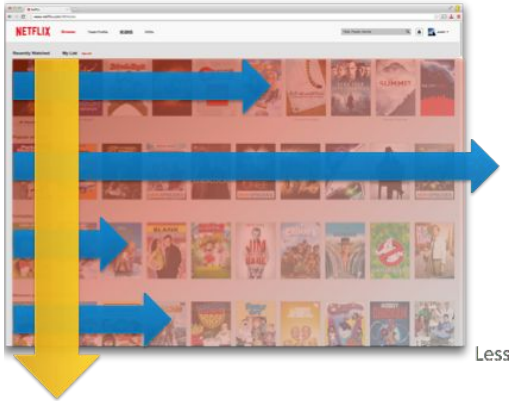
Tendencia:
Modelo de ML -> Commodity
Data pipelines y analisis -> ventaja competitiva

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

Netflix - *Recomendaciones*



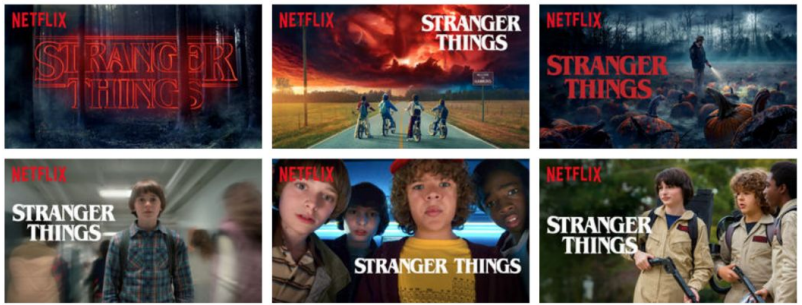
Netflix - *Recomendaciones*



Actividad - Netflix

Contesten a las siguientes preguntas:

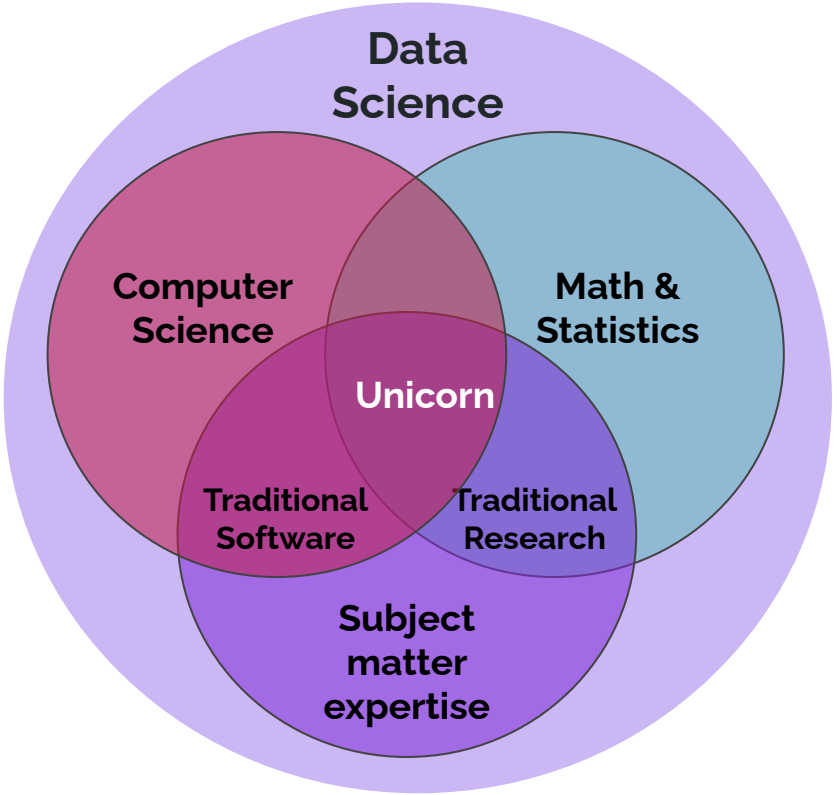
- ¿Cómo hace Netflix para recomendarte películas?
- ¿Por qué a algunos muestra letras rojas y a otros blancas, con distintas letras de fondo?
- ¿Cómo los ordena / rankea?
- ¿Qué datos utiliza para recomendarte?
- ¿Cuáles son los componentes principales?



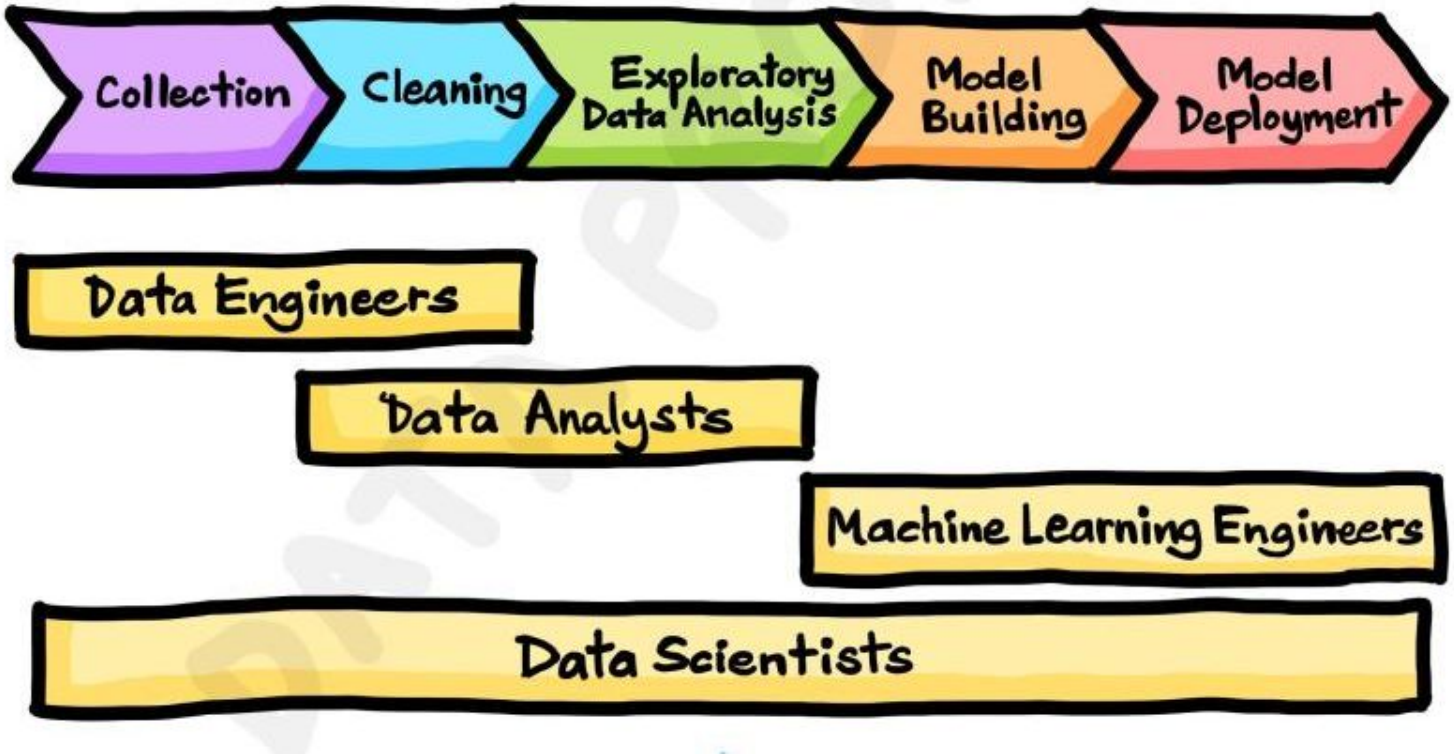
Ejecución de un proyecto

Equipos interdisciplinarios:

- Data Scientists
- Data Engineers
- Data Analysts
- Analytics Engineers



Roles en ciencia de datos



https://www.youtube.com/dataprofessor

Pasos en un proyecto de Datos

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?

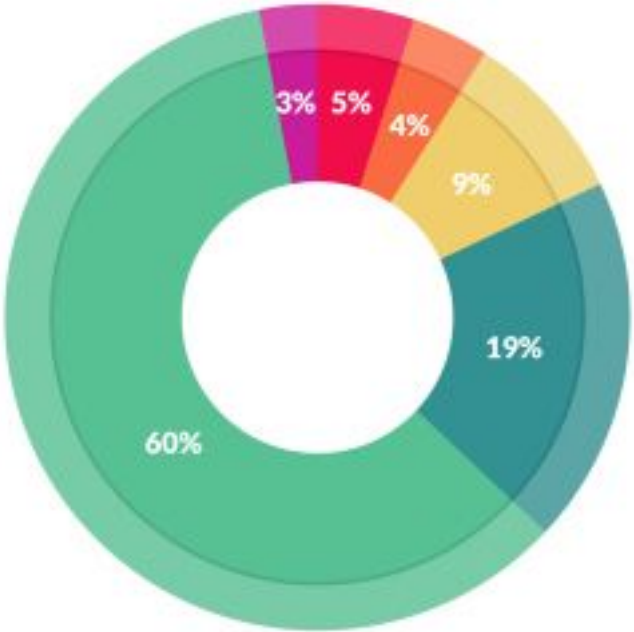
Diseño de experimentos

- ¿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

Limpieza y Exploración de datos



What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

Identificar el problema

- Hablar con los especialistas/stakeholders y entender objetivos
- Definir métricas de éxito del proyecto.
- 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?
    a. ¿Quién es el propietario de los datos? ¿Qué quiere a cambio?
- Evaluar la calidad de los datos
- Definir pipeline e infraestructura de datos
- 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

- Identificar patrones y relaciones significativas
- Estadística descriptiva
- Para decidir los procesos de curación, tenemos que entender nuestros datos **en conjunto**. Incluye:
    a. Técnicas más complejas para el análisis de datos que permiten relacionar múltiples variables.
    b. 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.

Humanos detrás del ML

EL PAÍS | ECONOMÍA | SOCIEDAD | CULTURA Y ESPECTÁCULOS | EL MUNDO | DEPORTES | PSICOLOGÍA | CONTRATAPA SECCIONES

Página12

Edición Impresa | 07 de junio de 2018
Hoy: UNIVERSIDAD NO

CULTURA Y ESPECTÁCULOS

10 de abril de 2018

Un trabajo soñado para los fanáticos del streaming

Netflix paga por ver y etiquetar series y películas

Arts and Entertainment

Netflix tagging: Yes, it's a real job

By Jhaan Elker June 11, 2015 Email the author

Josh Garrell is a Netflix tagger. His job? He is paid to watch television shows and movies for hours a day. (Jhaan Elker/The Washington Post)

Starting in the wee hours of June 12, hundreds of thousands of fans are

Desarrollar y entrenar modelo

- Entrenar modelos predictivos, seleccionar algoritmos y variables con las que entrenar, analizar su rendimiento
- ¿Tenemos una hipótesis de experimentación?
- ¿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?

Puesta en producción - Despliegue

- Modelo validado debe implementarse en un entorno productivo para uso en el mundo real
- Integrar el modelo en un sistema existente
- Implementar interfaces de usuario para ser consumido (API).

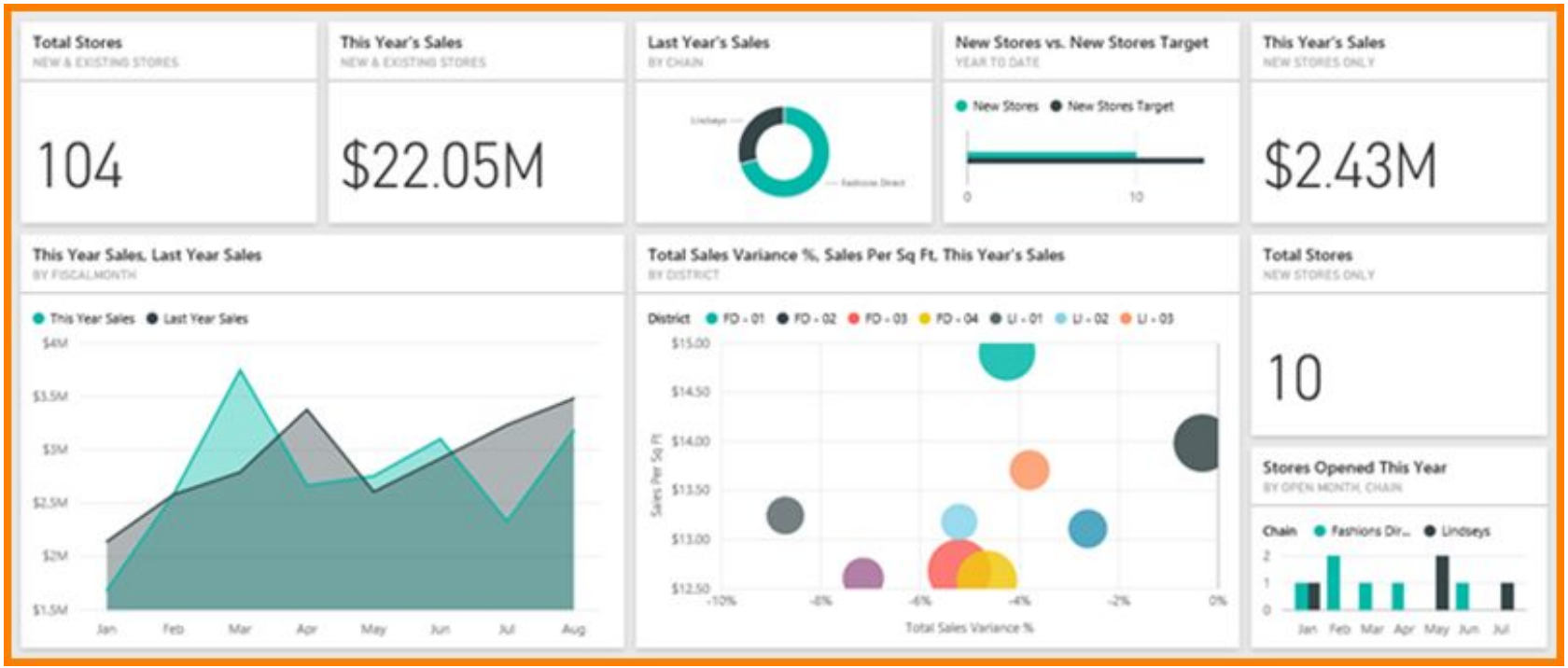
Netflix - *Competencia*

- Premio U$S 1 Millón.
- Sistema de recomendaciones.
- Predecir películas que le podría interesar a cada usuario.
- Netflix nunca pudo utilizar la solución ganadora debido a los costos de ingeniería requeridos.

Solución de datos: Visualizar y comunicar resultados

- Evaluar, reportar y comunicar resultados.
- Facilitar la interpretacion de los datos
- Promover la toma de decisiones informada / basada en datos.
- Storytelling
    a. ¿Entendemos a quién le estamos comunicando, cuáles son sus conocimientos e intereses y cómo difieren de los nuestros?

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 / Amplitude /
- Inglés
- Git (hacer pull request)
- CURIOSIDAD!

Tipos de Datos

¿Qué datos generamos cuando salimos a correr escuchando música?

Fuentes de datos

- Internas / Externas / Comerciales / Públicas

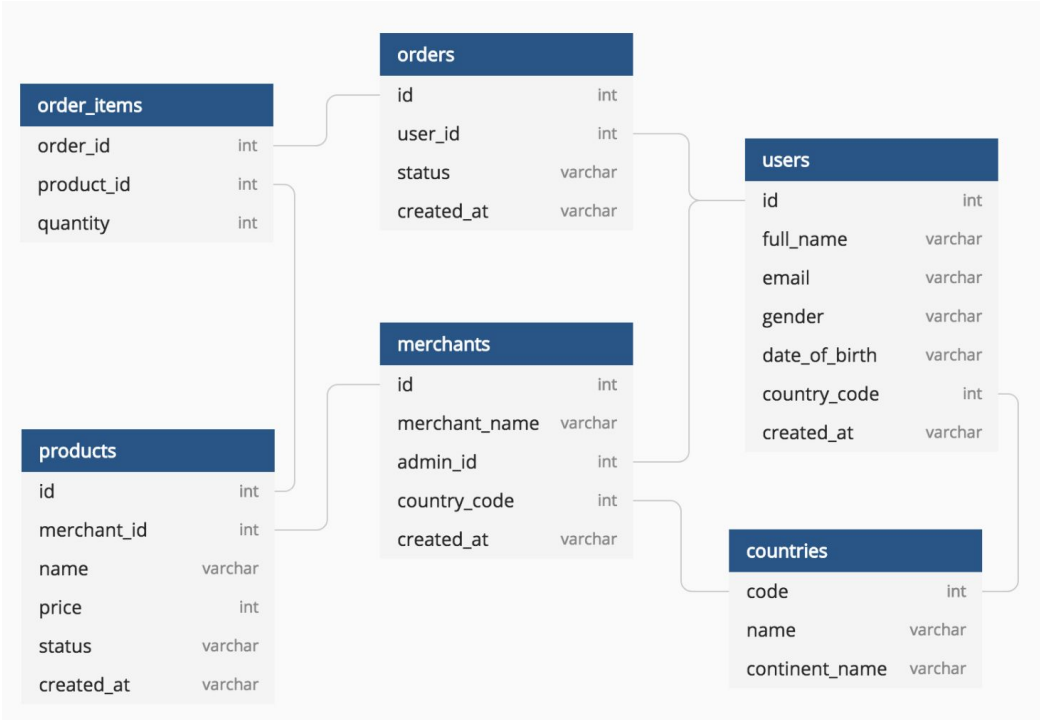
- Transaccionales: Compras, ventas, cotización de acciones en la bolsa, etc.
- Conversacionales: chats, mails, grabaciones telefónicas.
- Fotos y videos: subir a las redes sociales, etc.
- Sensores: posición del gps, acelerómetro, etc.
- IoT: smart tv, smart watch, alexa, etc.
- registro de actividades: escuchar música, 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
- Filas y columnas, formato consistente
- Fácil de almacenar, consultar y analizar.



- 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

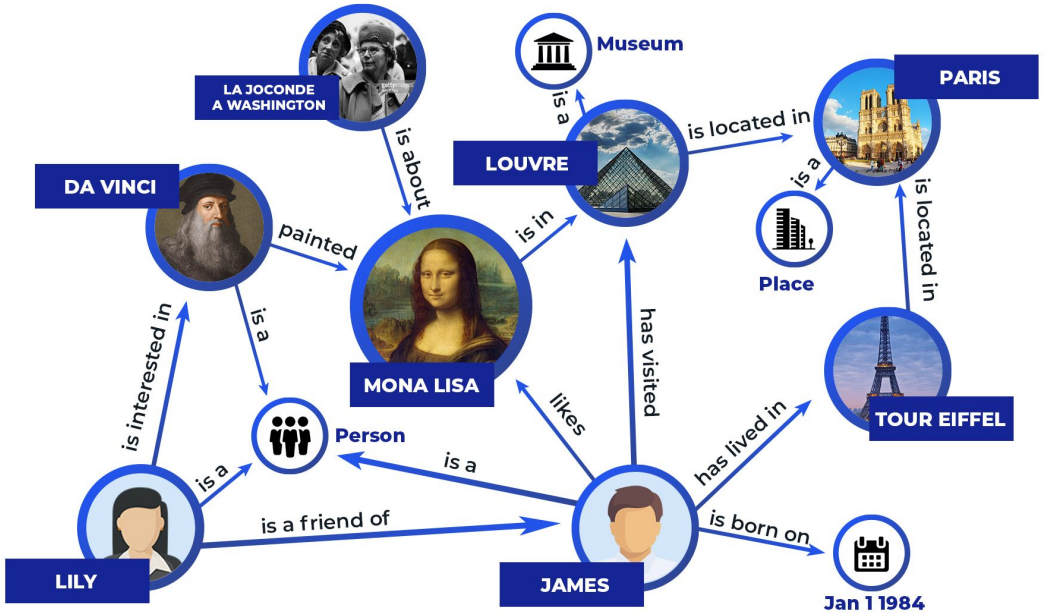
- Cada registro tiene un conjunto distinto de características
- Los registros pueden estar anidados
- Formato flexible, pero requiere procesamiento adicional para analizarlo


```
{"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
- Requiere técnicas especiales para su procesamiento y análisis

Calidad de datos

- **Completitud** — Tenemos toda la información necesaria, valores faltantes?
- **Validez** — Es el tipo de datos correcto
- **Precisión** — Los datos reflejan exactamente 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?

Ingesta de Datos

Formatos de Datos

- Tabulares: como una planilla, con filas y columnas.
  - Formatos de Archivos: CSV, TSV, XLS
  - Estructura de Datos: Dataframe
  - Son fáciles de manipular y analizar utilizando herramientas como Pandas en Python
- Jerárquicos: con valores anidados dentro de otros valores.
  - Formatos de Archivos: JSON, XML
  - Estructura de Datos: Lista de Objetos
  - utilizado para el intercambio de datos en aplicaciones web y servicios web.
- Crudos: sin estructura específica
  - Formato de Archivos: TXT
  - Estructura de Datos: String
  - Requieren procesamiento adicional antes de poder ser analizados y utilizados

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 primarias
- Permiten combinar columnas de una o más tablas (JOIN)
- Utilizan el lenguaje SQL (Structured Query Language) para consultar y manipular los datos

Modelo Relacional: Ejemplo

Modelo relacional

Ejemplo de datos

jugadores

id_jugador	nombre	id_equipo
1	C. Ronaldo	1
2	Messi	2
3	Griezmann	3
4	Luis Suárez	2
5	Bale	1

Clave secundaria
FK

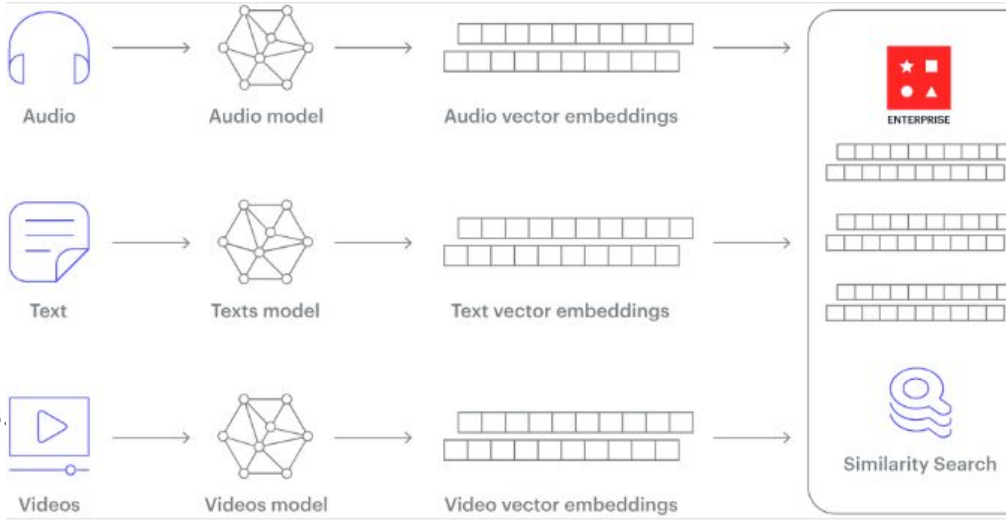
equipos

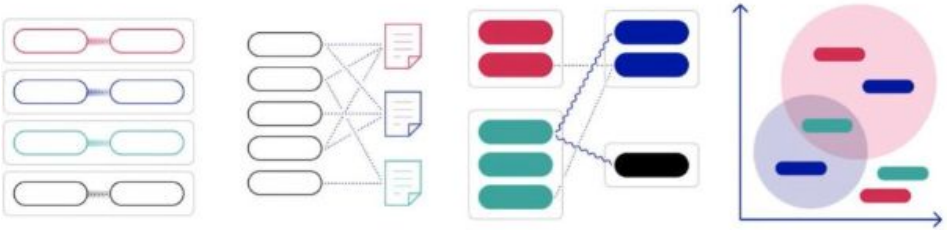
id_equipo	nombre
1	Real Madrid
2	F.C. Barcelona
3	Atlético de Madrid

Clave primaria
PK

Bases de Datos No Relacionales

- Conocidas como NoSQL.
- Manejan datos semi estructurados
- No permiten JOIN, cruzar tablas
- Varios tipos:
  - Documentales ej MongoDB
  - Grafos Neo4j
  - Vectorizadas, representan embeddings.





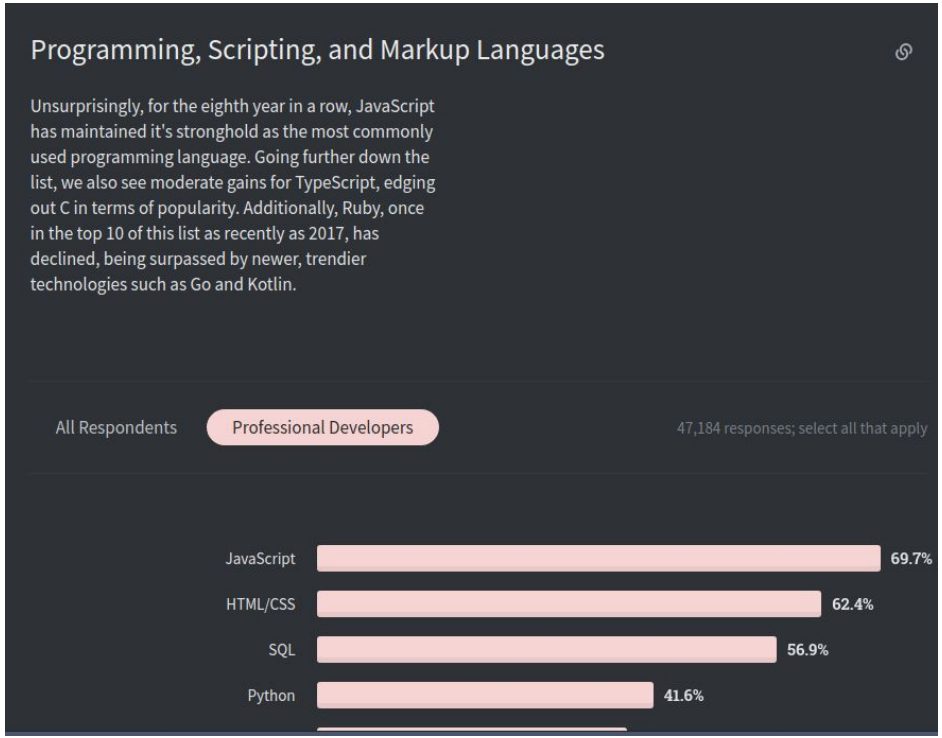
Demo notebook

01 Exploracion.ipynb

SQL

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)
- Tutorial: https://www.sqlitetutorial.net/tryit/

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*)

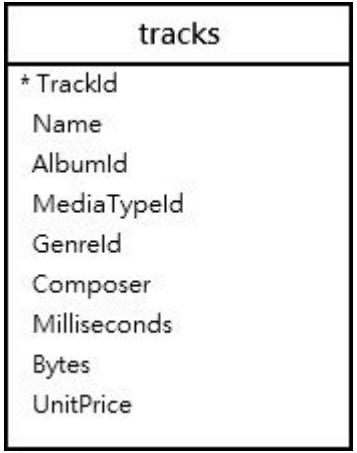
Consultas - SELECT / FROM

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

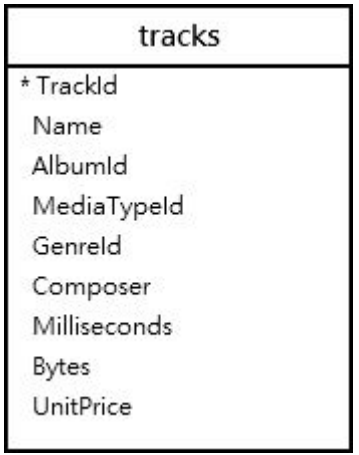
```
SELECT name, composer
FROM tracks
```



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

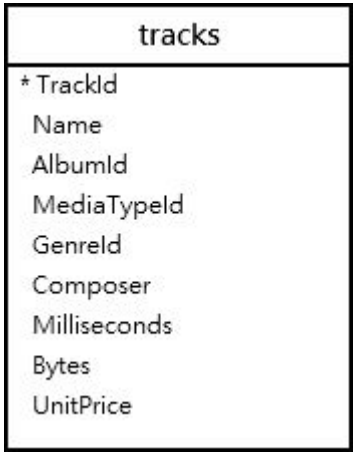
Ejemplo: Ciudades y países de los clientes

```
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

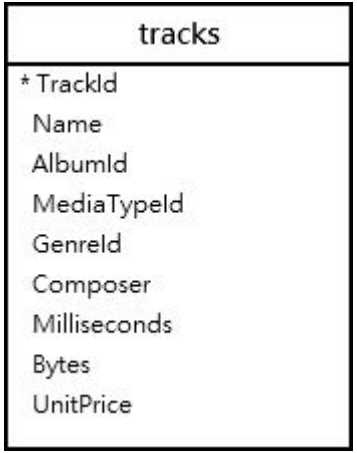
```
SELECT name, composer
FROM tracks
WHERE composer LIKE '%SMITH%'
```

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
```

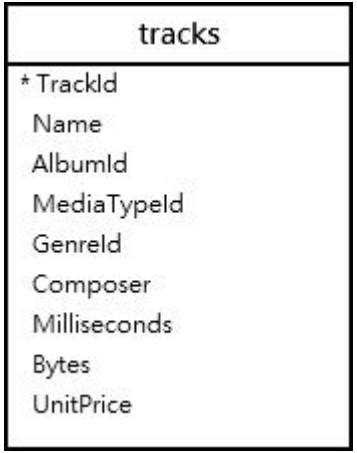


Consultas - GROUP BY agrupar resultados

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.

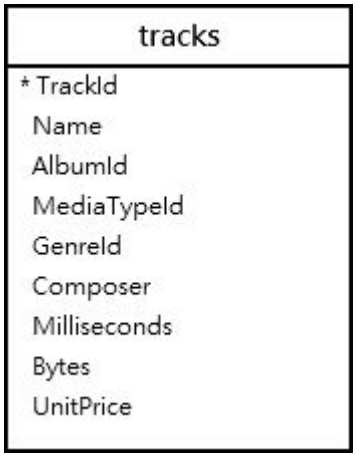
Ejemplo: Los 10 compositores más prolíficos y cantidad de canciones

```
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
```

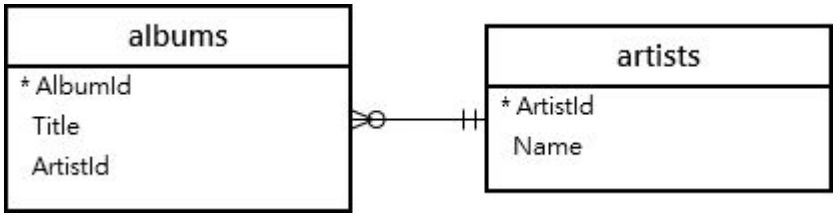
Consultas - JOIN

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.

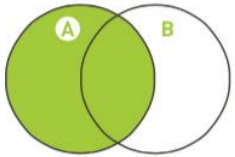
Ejemplo: Álbums y sus artistas

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

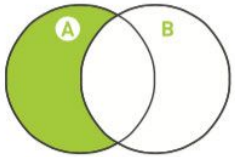


SQL JOINS

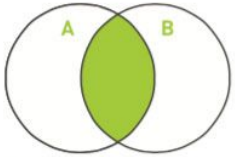
A CHEATSHEET BY WEBDEZIGN.CO.UK



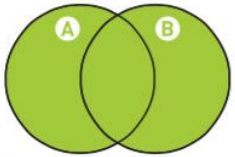

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



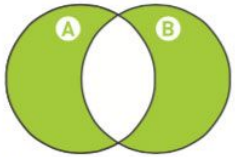

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



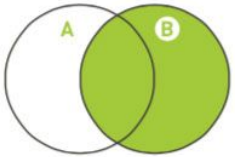

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



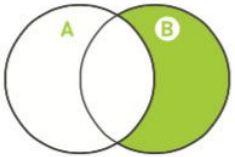

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



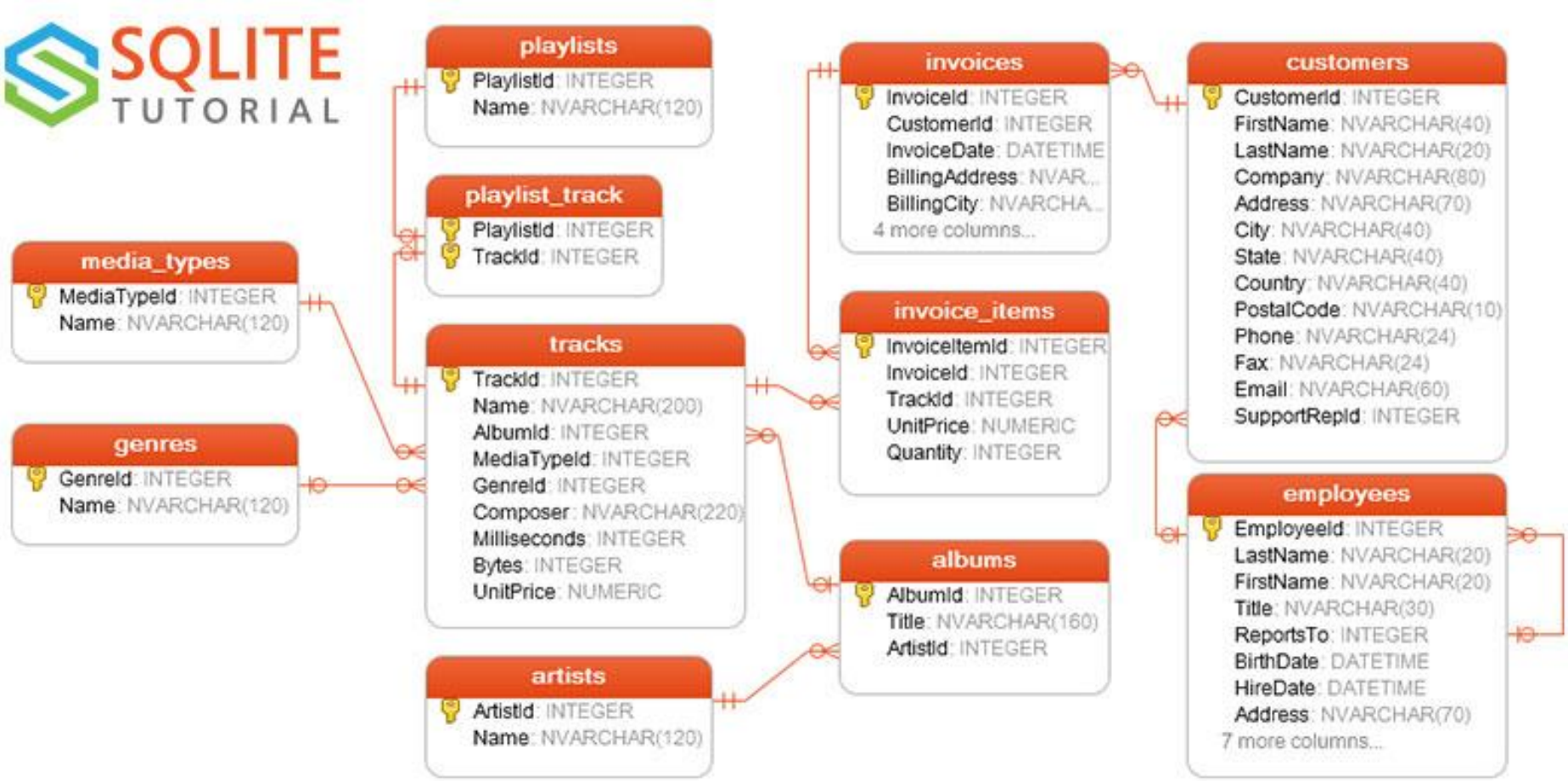

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```




```
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)
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
albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: 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

02 SQL

Pandas

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 ej: sum(), mean(), count().
   - 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

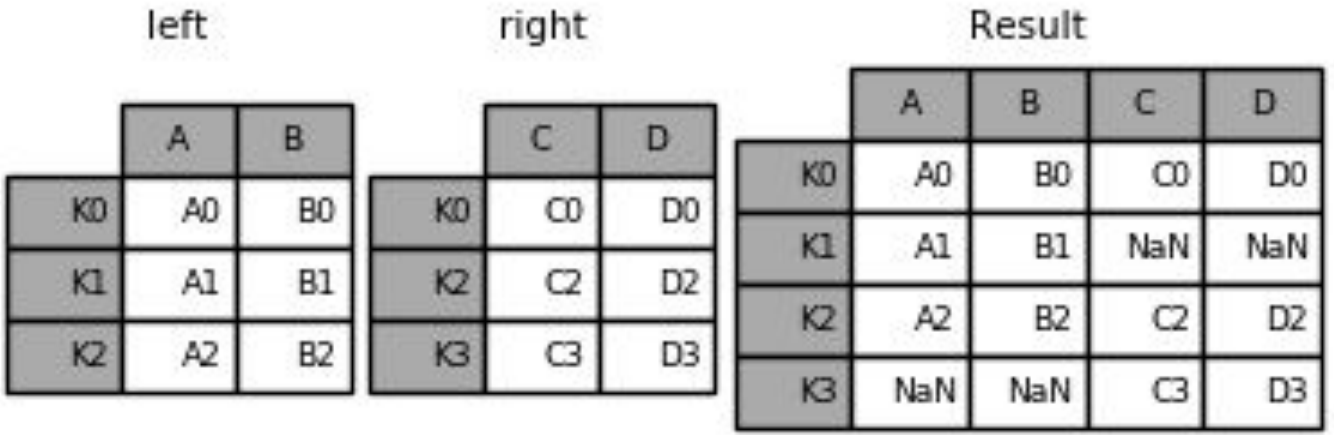
SUM

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, basado en su índice (horizontalmente) los DataFrames y hace coincidir las filas donde el valor del índice es el mismo



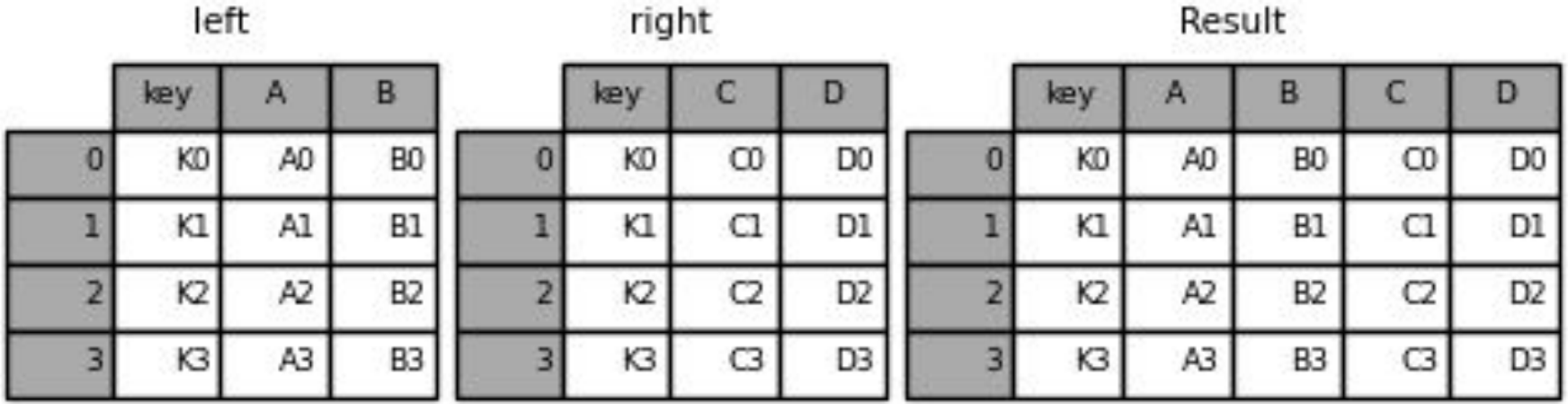
	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

	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

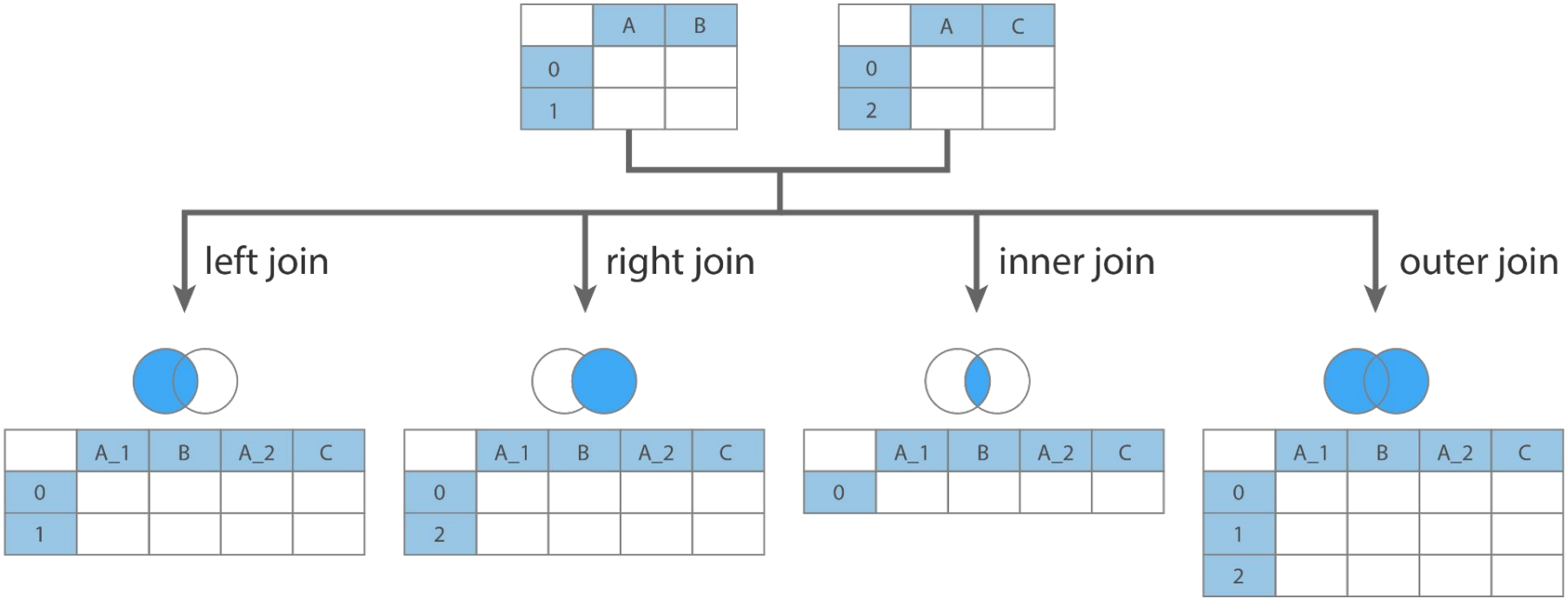
right

	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

Link a la notebook

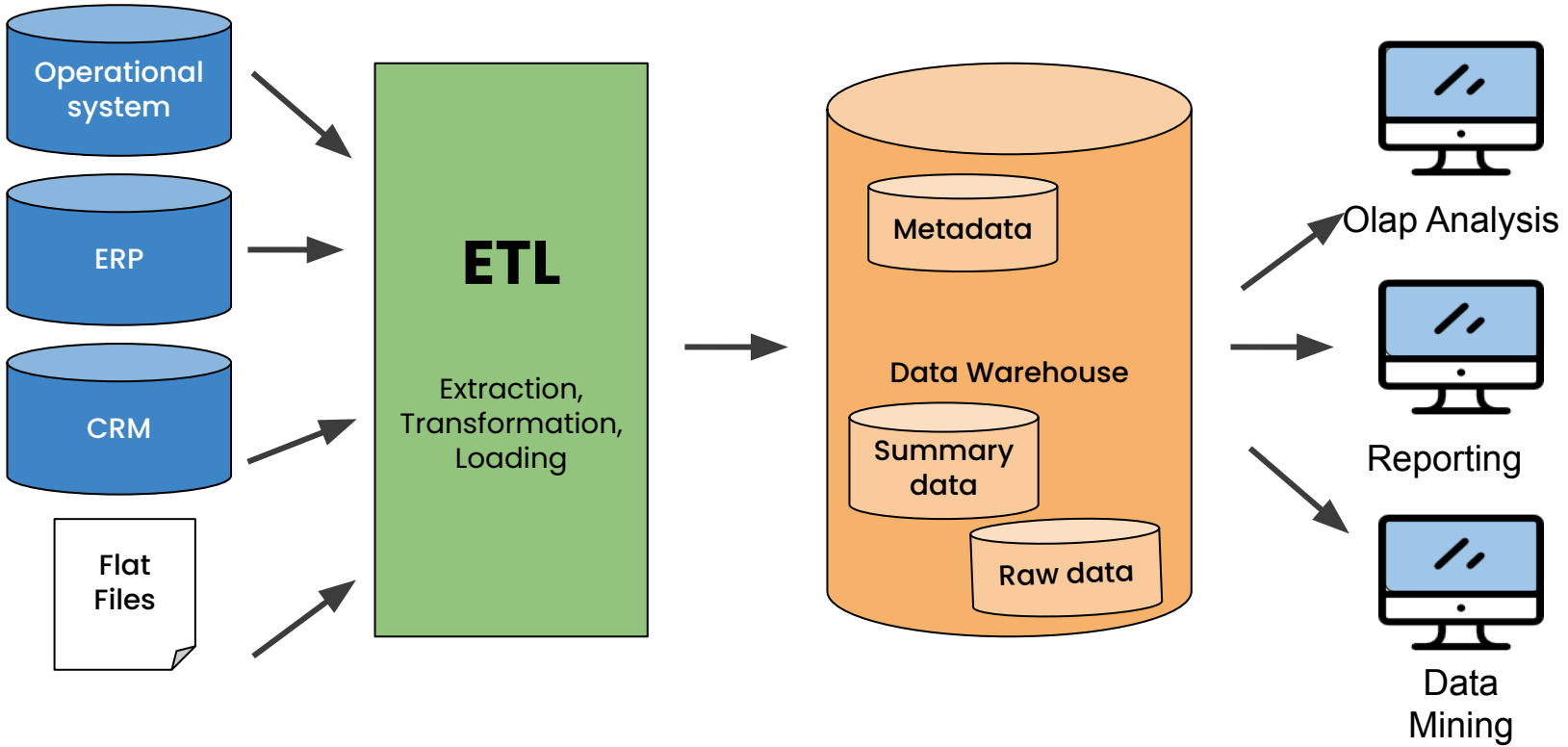
Pandas Vs SQL

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

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

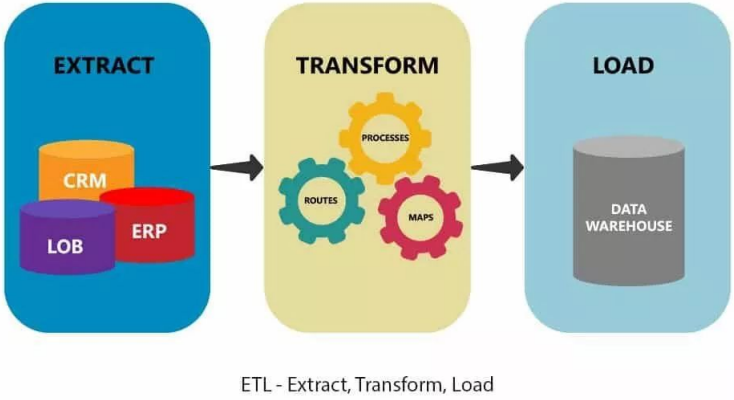
Mecanismos de Ingestión de Datos

Data Warehouse / Data Lake



ETL

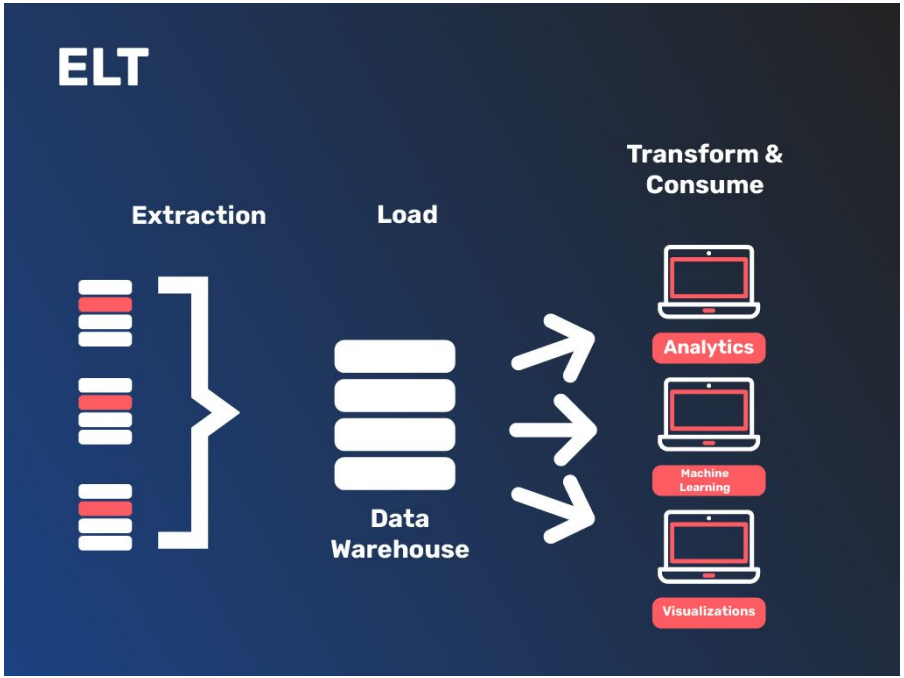
- **Extracción**:
  - Extraer datos heterogéneos de las distintas fuentes provistas.
- **Transformación**:
  - Normalizar
  - Elegir solo las partes relevantes
  - Interrelacionar los datos de las distintas fuentes
  - Adaptar al esquema de destino
- **Load** (carga):
  - Agregar toda estos datos al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



ETL - Extract, Transform, Load

ELT

Carga datos en el warehouse **antes de que sean transformados**. Luego, cada business user los transforma de la forma más eficiente para c/u.

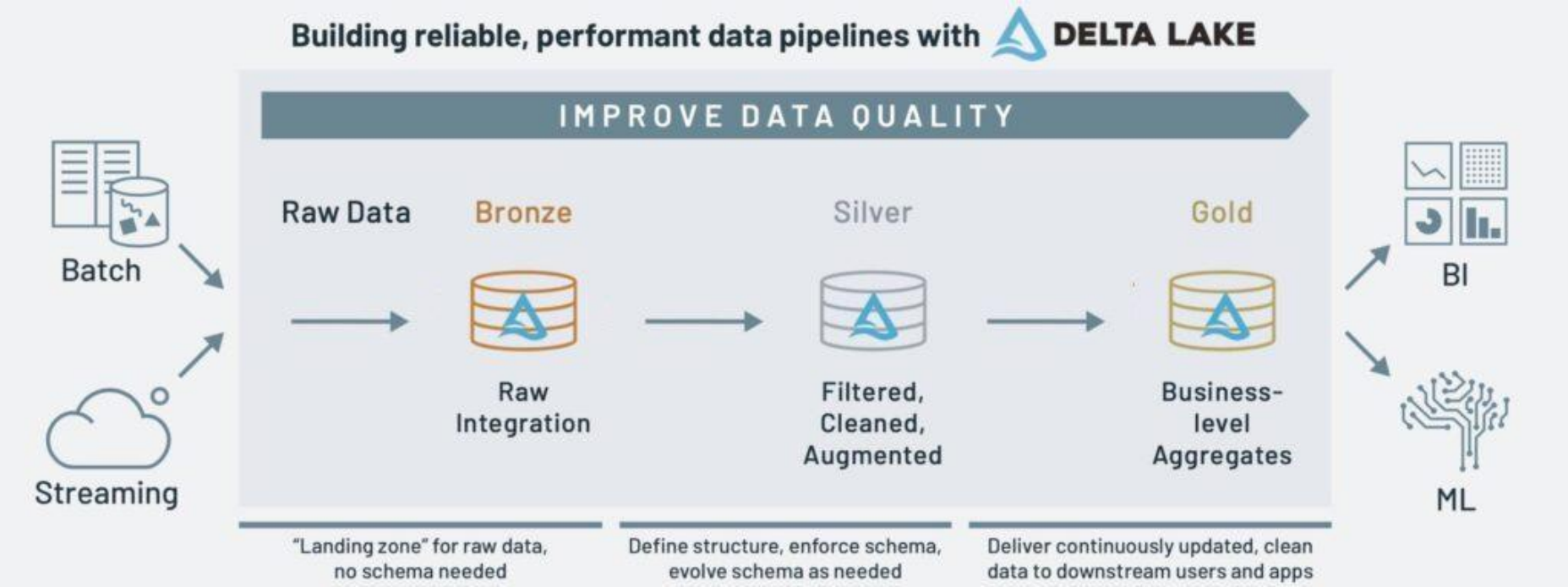


Batch vs Real time

El procesamiento de datos puede ser de a “Batchs” o “Real Time”.

- **Batch**: permite hacer análisis complejos sobre grandes volúmenes de datos.
  - *Ejemplo*: ETLs/ELTs.
  - se hace cada ciertos Intervalos regulares, por ej 1hs, 1 dia, etc.
- **Real time**: requiere de un procesamiento muy performante, volúmenes de datos menores.
  - *Ejemplos*: Posnet, streaming, reserva de asientos en una aerolínea.
  - Herramienta popular: Kafka

Modelado Datos por capas



➢ Capa Bronce (raw Data)

- Datos de los **sistemas de fuentes externas.**
- Puede recibir datos tanto de fuentes en **streaming** como de **batch**.
- Estructuras de la tabla “tal cual” a la tabla del sistema de origen, junto con las columnas de metadatos adicionales que capturan la fecha/hora de carga, el ID del proceso, etc.
- El enfoque en esta capa es la **captura rápida de datos de cambio** y la capacidad de **proporcionar un archivo histórico** de origen (almacenamiento en frío)
- Además permite hacer tareas de lineaje, auditoría o re-procesarlos en caso de ser necesario.
- Estos datos pueden incluir errores, duplicados o incosistencias

Capa Bronce → Capa Silver

Los datos de la **capa bronce** se **comparan**, **fusionan**, **conforman** y **limpian** ("lo suficiente") para que la **capa silver** pueda proporcionar una ***"vista enterprise"*** de todas sus entidades comerciales clave, conceptos y actas.

➢ Capa Silver

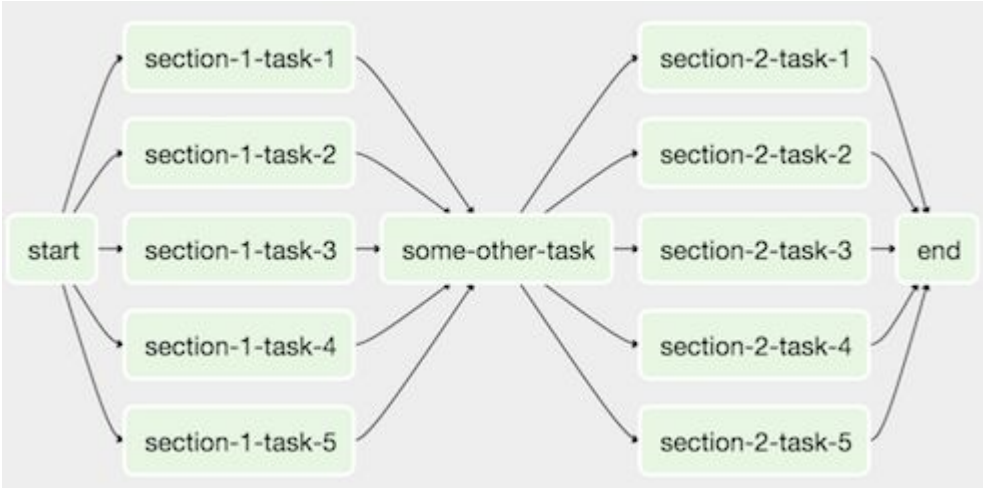
- Trae los datos de diferentes fuentes a una **vista empresarial** y permite self-service analytics, reportes ad-hoc, análisis avanzado y ML.
- Permite a **analistas**, **data engs** y **data scientists** responder a problemas comerciales a través de proyectos de datos empresariales y departamentales en Gold Layer.
- Para un data lake los data engs siguen la metodología ELT frente a ETL:
    - **Transformaciones mínimas** o "suficientes" y reglas de **limpieza** de datos mientras se carga la capa Silver**. Calidad de datos**
    - Prioriza **velocidad** y **agilidad** para ingerir y entregar los datos en el lago de datos
    - **Transformaciones complejas** y **reglas comerciales específicas** del proyecto mientras se cargan los datos de la capa Silver a Gold.

➢ Capa Gold

- Datos organizados para que queden **listos para ser consumidos** por proyectos específicos.
- Diseñado para **reportes** y optimizado para lectura de datos con **menos joins.**
- Se aplican reglas de calidad de datos y transformaciones específicas generando **“data marts”** según los casos de uso:
    - Customer segmentation, Product recommendations, Customer Analytics, etc.

Orquestar ETLs: Airflow

- Permite programar, monitorear y administar procesos.
- 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



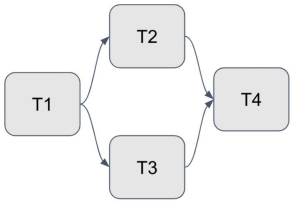
Apache Airflow

Airflow Conceptos principales

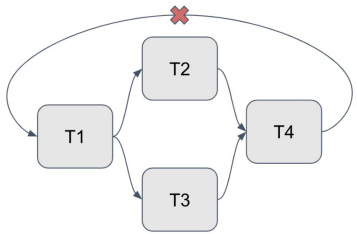
DAG
Un grafo acíclico dirigido, o DAG, es un data pipeline definido en código Python.
Cada DAG representa una colección de tareas que desea ejecutar y está organizado para mostrar las relaciones entre las tareas en la interfaz de usuario de Airflow.

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.



Valid



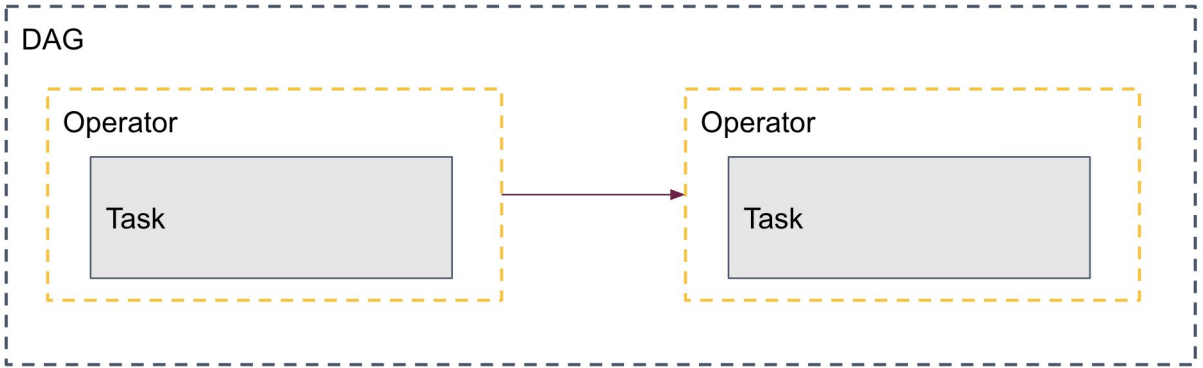
Invalid

Airflow Conceptos principales

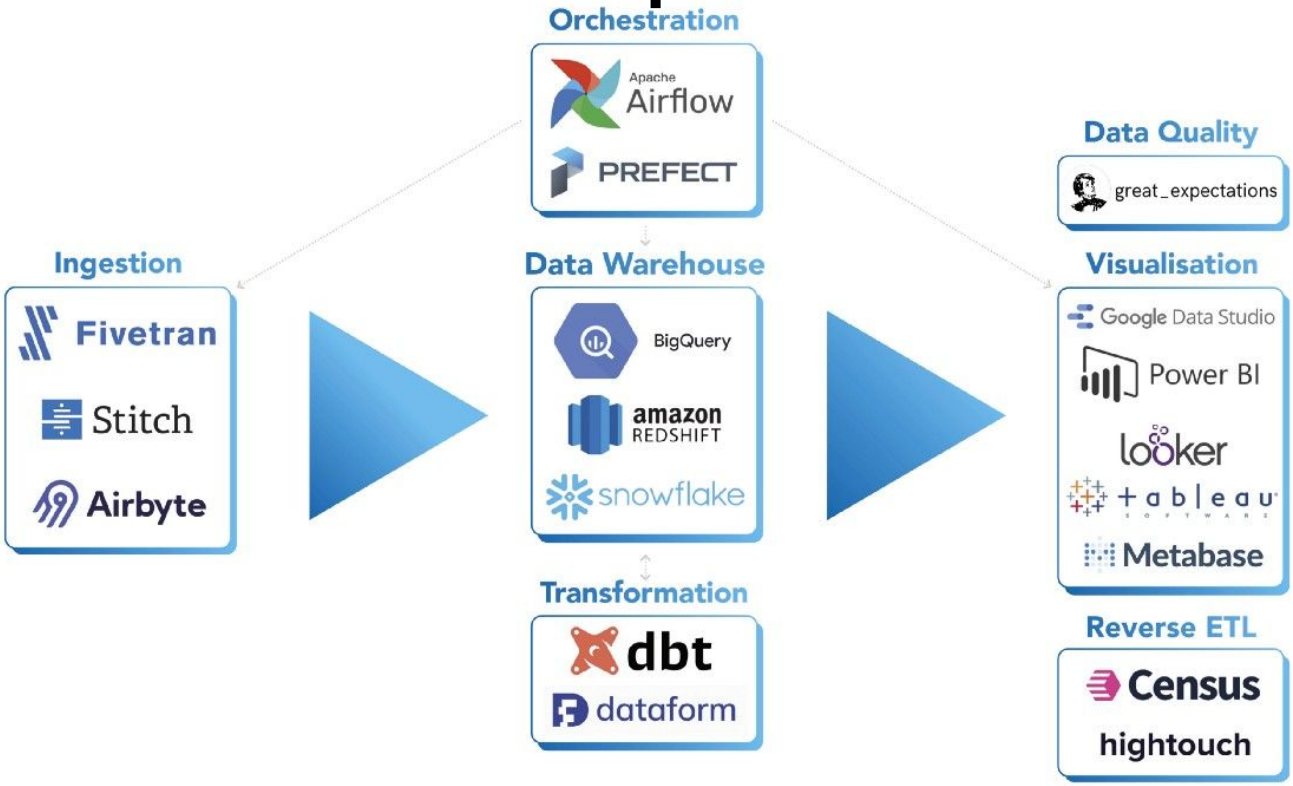
Task
Representa cada nodo de un DAG definido. Son representaciones visuales del trabajo que se realiza en cada paso del flujo de trabajo, y los operadores definen el trabajo real que representan.

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/



Modern Data Stack - Arquitectura y Componentes



Data Sources (Collection and Tracking)

Fuentes de datos:

- Distintas bases de datos productivas (ERP, CRM, microservicios, etc)
- Logs o eventos de un aplicación web o mobile
- Datos externos

Ejemplo: Cuando se concreta una venta, eso queda registrado como una **transacción** en la base de datos. Pero si el usuario dejó el carrito abandonado, eso comúnmente se ve con los **logs** o **eventos** de la aplicación web. Para saber la ubicación geográfica estimada de ese usuario se pueden consultar fuentes externas.

➢ Data Ingestion

- Múltiples fuentes de datos → es necesario que la **ingesta** sea de forma estandarizada.
- **Herramientas** que permiten ingestar datos de las fuentes hacia el lugar de almacenamiento principal del warehouse.
- Herramientas populares: Airbyte, Fivetran, Stich

➢ Data Storage

- Es donde se **almacenan** todos los datos de las múltiples fuentes ingestadas
- Herramientas populares: Snowflake, Redshift, Databricks, Big Query

➢ Data Transformation and Modelling

- Una vez ingestados es necesario **transformarlos** en **“modelos de datos”** más amigables para que puedan ser explorados fácilmente sin tener que suponer o adivinar definiciones.
- Generalmente los datos se modelan por capas: ***raw***, ***bronze***, ***silver*** y ***gold***.
- Herramientas DBT

➢ Data Orchestration

- Para poder ingestar y luego procesar los datos es necesario programar que **todos los procesos establecidos** corran en el momento correcto, disponiendo de los recursos necesarios y siendo monitoreados por si alguno falla.
- **Orquestar** todas las otras herramientas para que funcionen de forma integrada

Ejemplo: todos los días a las 7am ¿Cuántas ventas por los distintos canales se hicieron en la última hora?

- Airflow, Prefect, Dagster

➢ Data Visualization

- A partir de los datos ya procesados y modelados se pueden construir **tableros de visualización** para mostrar insights o monitorear métricas principales del negocio.

Ejemplo: Dashboard de ventas por país/ciudad de último día, semana, mes y como viene siendo la tendencia de los últimos 3 meses.

- Herramientas populares: PowerBI, Tableau, Looker, Databricks SQL, Microstrategy

➢ Data Operationalization / Reverse ETLs

- Muchas veces es necesario exponer los datos ya procesados/modelados en otras **herramientas externas** para operacionalizarlos.

Ejemplo: Armar un segmento de usuarios que hicieron una compra la semana pasada para hacer una publicidad en Facebook/Google.

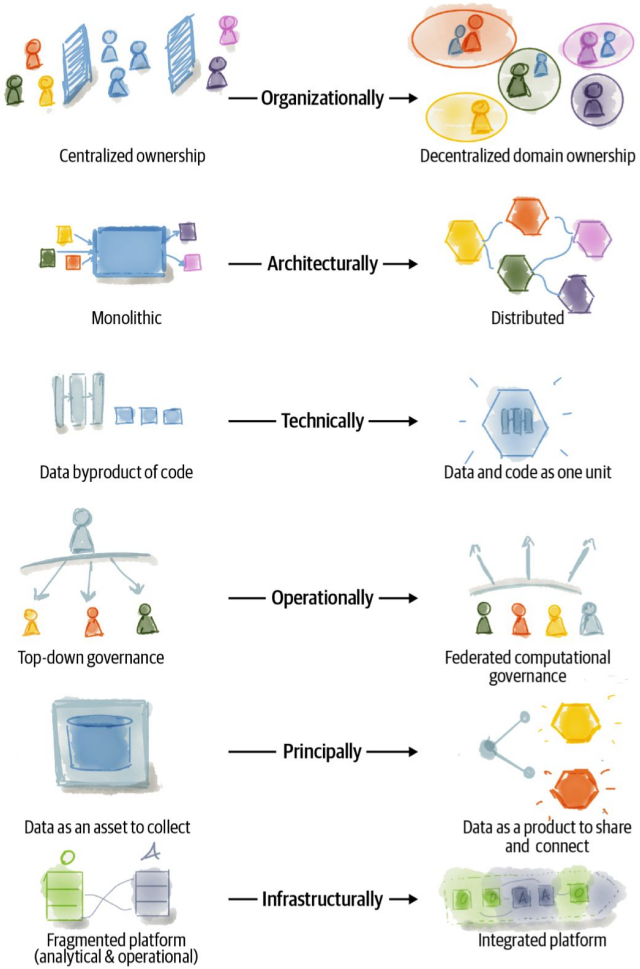
- Herramientas populares: Hightouch, Census, Segment

Otros componentes adicionales...

- **Product Analytics:** para ver los eventos de la web Amplitude, MixPanel, Google Analytics.
- **Events Taxonomy:** Estandarizar las propiedades de los eventos con Avo, Iteratively
- **Data Quality:** validar la calidad de los datos Great Expectations
- **Data Catalog:** Definir un catálogo de datos
- **Data Governance:** administrar permisos de acceso, edición, definir políticas de privacidad y ownership, gestión de datos sensibles.

Data Mesh

- Enfoque **descentralizado** sobre los datos analíticos, segmentados por **dominio**.
- Requiere cierta **madurez** tanto de los datos como de la organización para que pueda ser explotado exitosamente.
- Permite **moverse ágilmente** frente a cambios del negocio.
- Incrementa la **visibilidad** a los stakeholders sobre el valor de los datos.



Libro: Data Mesh: Delivering Data-Driven Value at Scale, Zhamak Dehghani

Bonus

Notebook ETL y DAGs

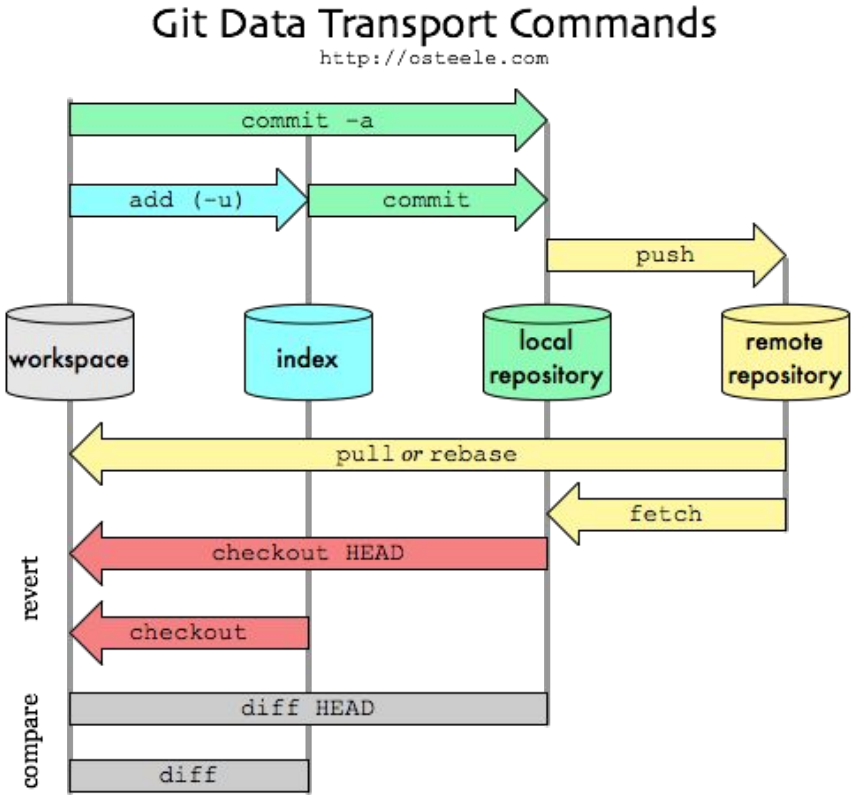
Entregable parte 1

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://airflow.apache.org/docs/apache-airflow/stable/tutorial/index.html
  - https://www.youtube.com/watch?v=IH1-0hwFZRQ
  - https://www.youtube.com/@MarcLamberti
  - https://medium.com/@dustinstansbury/understanding-apache-airflows-key-concepts-a96efed52b1a
  - https://www.udemy.com/course/the-ultimate-hands-on-course-to-master-apache-airflow/