Big Data

No es Business Intelligence. Sino el transformar datos en tales cantidades que no es posible hacerlo en una sola máquina. Hay un crecimiento exponencial en la cantidad de datos que se generan. De 44zb a 64zb. La data abierta a nosostros se esta haciendo mas grande o pequenia.

History

El mito de la máquina, 1950s. El pentágono del poder.

- 1958: HP Luhn: BI es la colección y transformación de datos para tomar decisiones estratégicas. Pero no habló de las dimensiones.
- 1962: El gob. de USA disenia los primeros centros de datos. Para almacenar las declaraciones de impuestos de todos los ciudadanos, en forma de documentos.
- 1972: Edgar F Codd. Se disenia el modelo de datos relacional.
- 1976: Comienzan los MRPs (Manufacture Resource Planner?). Sistemas para administrar los materiales, control de inventario. De estos surgen los ERP (Enterprise Resource Planning).
- 1989: Erik Larson, de Lotus Notes, habla de la Big Data. En un correo escribe “Los que mantienen la información (big data) dicen que lo hacer para el beneficio de los clientes, pero la información tiene formas de usarse de otras formas que las que a lo que estaban destinado.” El propósito de los datos puede cambiar sobre la misma base, para obtener otro tipo de objetivos que no siempre nos benefician.
- 1991: Tim Berners-Lee. WWW
- 1999: Big Data aparece en una conferencia
- 2001: Doug Laney habla de las 3 Vs of Data.
- 2005: Nace Web 2.0
- 2007: Concepto de Big Data como lo conocemos ahora

5Vs de la información

- Volume: Qué tan grande es la información?
- Velocity: Qué tan rápido se produce tu información. Si tienes muchos datos rápido vale la pena iniciar a pensar en Big Data aunque aun no sea grande, pues eventualmente lo será.
- Variety: La forma en que está formada la información, qué es? Eso afecta cómo se va a almacenar
- Veracity: Qué tan confiables son los datos.
- Valor: De qué sirve almacenar muchos datos si no se procesan para que aporten algo a la empresa. “Las empresas no necesitan big data, necesitan la data correcta”

Ejemplos

- Deportes:

  Pelicula: Brad Pitt. Moneyball

  : Vieron que lo que necesitaban eran jugadores que dieran asistencias.
- Sistemas de recomendación: Por ejemplo el de TikTok que funciona en tiempo real. Es un modelo sencillo. En base a tu interacción con la pantalla y los videos, actualiza el score para cada categoría. En Netflix, en cambio, tiene que pasar tiempo.
-

Modelo de datos

- Conceptual: Cómo se va a realizar la realción entre las entidades, se necesita un marco de trabajo para la necesidades de la información.
- Logical: Conceptos, relaciones profundas between entities.
- Physical: Requirements, horas de trabajo. La propuesta formal, qué es (data lake) dóne (hosteado en x con tantos núcleos)

Arquitecturas

Data Warehouse

- Datos relacionales
- Aka. Un montón de tablas con relaciones y esquemas
- Centralizado y debe ser data de forma estructurada

Data Mart

- El configurar interfaces para que las áreas interesadas puedan acceder a la información que les sirve

Data Lake

De lo que más se usa en las empresas

- Se almacenan de forma cruda los datos. Puede almacenar estructurado, semi-estructurado o no estructurado.
- Todos pueden consumir del lago de datos.
- Apache Spark y Databaricks (Spark de paga con UI amigable).

Data Fabric

Data Mesh

- Réplicas de data-lakes. Data lakes para distintos fines (uno para los proveedores, otro para x cosa y otro para n cosa).
- Hay un mesh-catalog que agrega los datos de los distintos lagos. El catálogo nos permite identificar la fuente del lago donde proviene.
- Hadoop

Qué arquitectura elegir?

De 3 áreas, solo es posible darle prioridad a 2: (Lo mismo de Barcena)

- Consistency: Todos los clientes ven el mismo estado, incluso con actualizaciones ocurriendo.
- Availability: Todos los clientes pueden encontrar una replica de la información, incluso en el caso de fallos parciales.
- Partitioning: El sistema continua el trabajo como se espera, incluso en el caso de la falla parcial de red.

Este es mi diseno para el sistema.

- ¿Considera qué pasa cuando falla? ¿Se recupera?
- ¿Se rompe sin internet?
- ¿Se rompe si no esta disponible todo el sistema?

Nota: Hay que ser capaces de hacer una buena arquitectura sin sobre-explotar recursos/ presupuesto.

Ejemplo de Netflix:
- La base de datos de películas es no relacional. Para que se pueda hacer búsquedas rápidas?
- Dependiendo de el consumo de los recursos el cache va recordando la información.
- El experimento más grande de Netflix sobre la reacción de usuarios y la interacción en base a la portada y título fue con Stranget Things.
- Mucho método científico. Son necesarios grupos de control, AB testing, etc.

Intentar entender el tipo de datos, conociendo las arquitecturas que hay, nos pueden a elegir a elegir la mejor.

Data Types
- Estructurado: Tablas, SQL
- Semi-estructurado: JSON, XML. NoSQL. Aunque hay etiquetas e identificadores no hay una estructura pre-definida para cada JSON/XML que se procese.
- No estructurada: Información en crudo

Aproximaciones híbridas
Emplear una mezcla de las distintas formas de almacenamiento, basado en sus fortalezas:
- Structured: Se puede volver limitante por la forma fuerte de los datos, en casos donde evoluciona mucho los requerimientos.
- Información semi-estructurada: con cuidado a la hora de la transformación y evolución del esquema.
- No estructurada, aunque presenta retos en la forma de almacenarlo y leerlo. Como no hay reglas, puedes hacer todo mal.

Bases de Datos Relacionales

Pipeline de servicios
- Kafka permite hacer mensajes de cosas en tiempo real. Hay un servidor en Nueva York que tiene en tiempo real donde están ubicados los taxis.
- Pub/Sub o SNS y SQS es un sistema de mensajería para arquitectura basada en eventos.
- Dataflow: Para programar pipelines de transformación de datos usando pi-colecciones y que se guarde donde quieras.
- Streaming Analytics es caro. Hacer las cosas en tiempo real es complicado.
- Hadoop/Spark: Escrito en java. Procesa datos en lotes. En nuestro caso nuestro proyecto de segundo parcial. La gente muda su clúster y scripts pero resulta.
- Dataprep es para armar streams de procesamiento con UI (scriptless con clicks). Es la herramienta de los huevones, para empresas que no tienen un data scientist.
- DataLake: Los buckets de S3 en amazon para almacenar cosas en bruto.
- Cloud Firestore no relacional.
- BigTable NoSQL. para data mayor a 300 GB (mínimo, es muy caro y está en nodos que sortea para decirte tus llaves más pesadas y cómo actualizar los queries). Optimiza los índices para hacer que sea más eficiente. Optimiza hot-spots automáticamente. Si habla de un terabyte, análisis de series de tiempo y cosas bancarias va en big table.
- Cloud Spanner: Para cuando necesitamos disponibilidad en distintas regiones geográficas de los datos.
- DataForm maneja pipelines de SQL para irlo transformando, cuenta con versionamiento de git.

- BigQuery ya tiene muchas cosas para visualizar y eso.
- Cloud SQL: Si ya tenemos un servidor de mysql o postgres con scripts y cosas asi podemos usar Cloud SQL para más rápido.
- Looker pero es muy caro, 4000 dolares.
- Looker Studio es gratis y el pro a 9 dólares el mes. Cada dos semanas hay nuevos conectores.
- Vertex AI: Igual muy caro. Se rentan TPUs y eso.
- Cloud Composer: Si tenemos varias fuentes y no sabemos a cuál acceder para obtener los datos podemos usarlo para encontrarlo.
  - Basado en Apache Airflow para coordinar las cosas que están pasando.
- Cloud schedule: Cómo traemos data de minuto a minuto y se guarden en tal lado. Usa chron, un job que se corre cada determinado tiempo. Hay pub-sub para que cuando termine se guarde.
- Cloud function: Empaqueta tu función. Puede tener un trigger.

- Cuando algo sale mal vemos los logs.
- S2 y ese tipo de cosas de instancias van por otro lado, es más como.
- BigQuery está en todas las etapas, es un data-warehouse que puede hacer de todo, incluyendo información semiestructurada en JSONs