Procesamiento de Datos Masivos

Clase 03 - Data Warehousing

OLTP vs OLAP

OLTP

OLTP: Online Transactional processing

Se basan en las técnicas de índices tradicionales

Una BD transaccional está pensada para hacer muchas operaciones livianas al día:

- Inserción, actualización y eliminación de tuplas
- Búsquedas simples: traer el usuario **x**, buscar los productos de la compra **y**, ...
- Además queremos soporte para propiedades ACID

OLTP

En general:

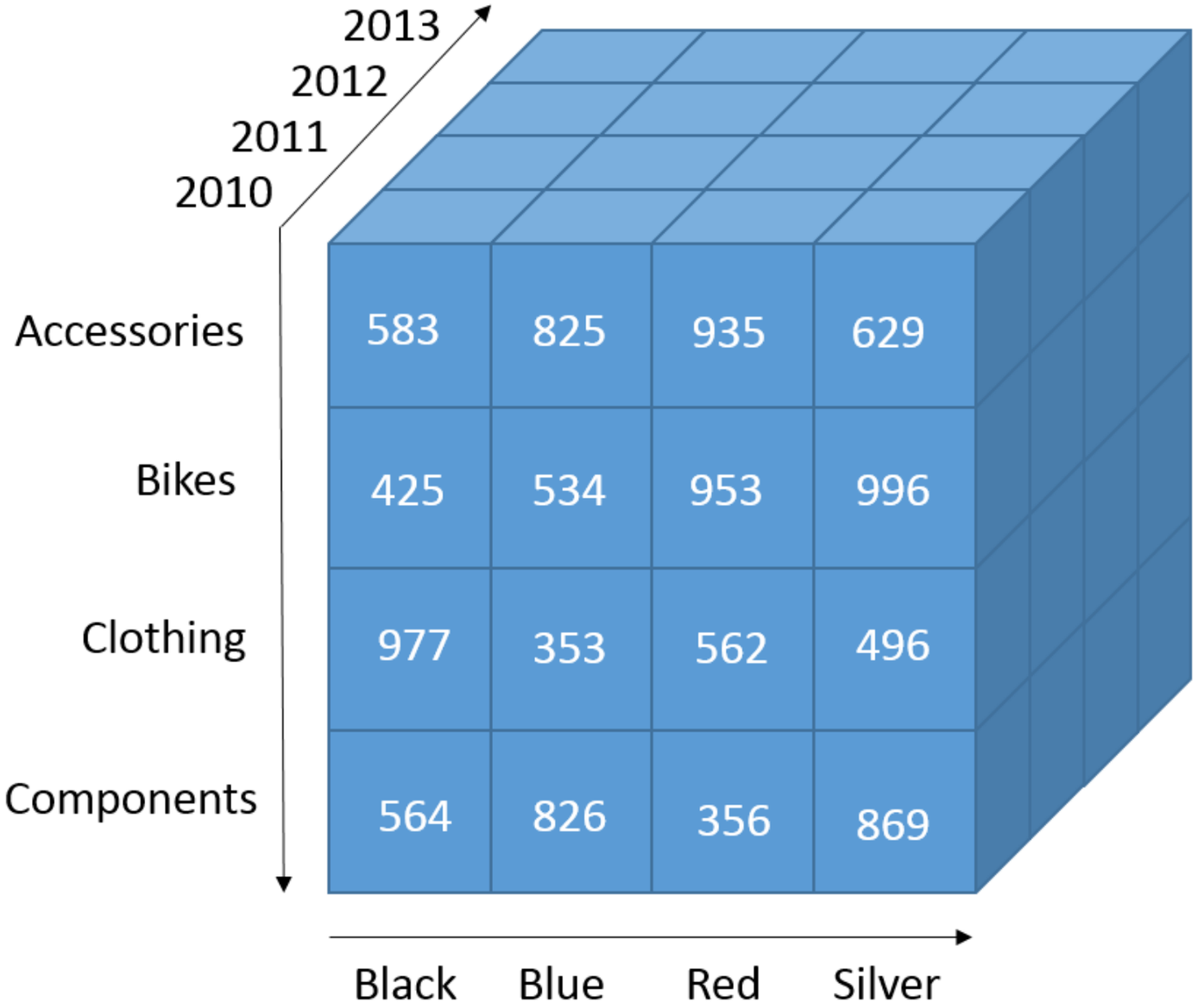
- Una aplicación en producción se sostiene sobre bases de datos transaccionales
- Estos sistemas no están pensados para realizar consultas pesadas de análisis de datos
- No es buena idea hacer análisis directamente en la base de datos de producción

OLAP

OLAP: Online Analytical Processing

OLAP es un paradigma para responder consultas de analítica de un negocio considerando diferentes dimensiones (ubicación, tiempo, tipos de producto, ...)

Se basan en la idea de armar un **cubo OLAP** (o hipercubo)


2013
2012
2011
2010
Accessories
Bikes
Clothing
Components
583
825
935
629
425
534
953
996
977
353
562
496
564
826
356
869
Black
Blue
Red
Silver

ROLAP

ROLAP: Relational Online Analytical Processing

Hacemos analítica sobre una base de datos que usa un esquema relacional, como por ejemplo BigQuery

"Typical BigQuery use cases include large-scale storage and analysis or online analytical processing (OLAP)"

Consultas de analítica

Cuando usamos un sistema pensado para hacer analítica, como BigQuery, estamos pensando en **un sistema que va a ser pocas consultas por segundo, pero cada consulta es muy pesada**

Data Warehousing

Data Warehousing

Un **Data Warehouse** es un tipo de sistema de datos que está diseñado para habilitar y apoyar en tareas de Business Intelligence (BI), especialmente el análisis de datos

En un Data Warehouse los datos están procesados y listos para ser consumidos por los analistas

Data Warehousing

En general, en un Data Warehouse relacional, voy a disponibilizar tablas que no están normalizadas para facilitar la analítica

Los dos tipos de modelamiento más clásico en este contexto son **Star Schema** y **Snowflake Schema**

Star Schema

Es un esquema más desnormalizado, donde hay una tabla de hechos (facts) central, rodeada de dimensiones

Star Schema

```sql
-- Tabla de Hechos: Ventas
CREATE TABLE Ventas (
    ID_Venta INT,
    ID_Producto INT,
    ID_Tiempo INT,
    ID_Cliente INT,
    ID_Tienda INT,
    Cantidad INT,
    Precio_Total DECIMAL
);

-- Tabla de Dimensiones: Productos
CREATE TABLE Productos (
    ID_Producto INT,
    Nombre_Producto VARCHAR,
    Categoría VARCHAR,
    Subcategoría VARCHAR,
    Precio_Unitario DECIMAL
);

-- Tabla de Dimensiones: Tiempo
CREATE TABLE Tiempo (
    ID_Tiempo INT,
    Fecha DATE,
    Mes INT,
    Año INT,
    Dia_de_la_semana VARCHAR
);
```

```sql
-- Tabla de Dimensiones: Clientes
CREATE TABLE Clientes (
    ID_Cliente INT,
    Nombre_Cliente VARCHAR,
    Email VARCHAR,
    Segmento VARCHAR
);

-- Tabla de Dimensiones: Tienda
CREATE TABLE Tienda (
    ID_Tienda INT,
    Nombre_Tienda VARCHAR,
    Ubicación VARCHAR,
    Tipo VARCHAR
);
```

Snowflake Schema

Es un esquema que busca normalización, donde hay una tabla de hechos (facts) central, rodeada de dimensiones y estas dimensiones suelen estar normalizadas

Snowflake Schema

```sql
-- Mantenemos las otras tablas

-- Tabla de Dimensiones: Productos
CREATE TABLE Productos (
    ID_Producto INT,
    Nombre_Producto VARCHAR,
    ID_Subcategoría INT,
    Precio_Unitario DECIMAL
);

-- Tabla de Dimensiones: Subcategorías (Nueva tabla para normalizar Productos)
-- Nos aseguramos que cada subcategoría tenga una única categoría
CREATE TABLE Subcategorías (
    ID_Subcategoría INT,
    Nombre_Subcategoría VARCHAR,
    ID_Categoría INT,
);

-- Tabla de Dimensiones: Categorías
-- (Nueva tabla para normalizar Productos)
CREATE TABLE Categorías (
    ID_Categoría INT,
    Nombre_Categoría VARCHAR
);
```

¿Cómo llevamos los datos a un Data Warehouse?

Pipeline de datos

Un pipeline de datos es la infraestructura que se va a encargar de mover los datos desde su fuente hasta el Data Warehouse

Stack de tecnologías

Un stack moderno para construir un pipeline de datos incluye las siguientes tecnologías:

- Airflow (u otro framework de orquestación de *jobs*)
- Docker y Kubernetes
- DBT: herramienta para armar las vistas
- Herramientas en la nube (GCS, BigQuery, Cloud Composer, GKE)

Apache Airflow

Apache Airflow es una herramienta para orquestar tareas

Estas tareas son encapsuladas en operadores (ej. PythonOperator, BashOperator, KubernetesPodOperator)

Algunas personas lo definen como "cron on steroids"

Apache Airflow

Se basa en la idea de definir un DAG (grafo dirigido acíclico) de tareas; este DAG se define mediante código Python

```python
from datetime import datetime

from airflow import models
from airflow.operators.python_operator import PythonOperator

with models.DAG(
    dag_id="dag_example_pythonop",
    schedule_interval="@once",
    start_date=datetime(2022, 1, 1),
    catchup=False,
    tags=["example"],
) as dag:

    def return_constant():
        return 42

    python_op_example = PythonOperator(
        task_id="python_op_example",
        python_callable=return_constant,
    )

    python_op_example
```

```
with models.DAG(
    dag_id="dag_example_pythonop_gcs",
    schedule_interval="@once",
    start_date=datetime(2021, 1, 1),
    catchup=False,
    tags=["example"],
) as dag:

    def return_constant():
        return 42

    python_op_example = PythonOperator(
        task_id="python_op_example",
        python_callable=return_constant,
    )

    list_bucket = GCSListObjectsOperator(
        task_id="list_bucket",
        bucket=GCS_BUCKET
    )

    python_op_example >> list_bucket
```

Docker

Docker es una herramienta que nos permite encapsular código en imágenes, y ejecutarlas como contenedores

Nos permite tener un entorno de ejecución uniforme en todos los lugares en que el código corra

Ejemplo de un Dockerfile

```dockerfile
FROM python:alpine

# Install GCC
RUN apk add gcc musl-dev libffi-dev

# Upgrade pip
RUN pip install --no-cache-dir --upgrade pip

# Install poetry
RUN pip install poetry

# Installing dependencies
RUN mkdir /app
WORKDIR /app

COPY pyproject.toml /app/

RUN poetry install --no-dev

COPY . /app

CMD ["poetry", "run", "python", "/app/src/main.py"]

```

Docker: algunos detalles

Un Dockerfile nos permite definir una imagen de Docker; podemos definir una imagen a partir de otra

Una imagen es un "computador base" que está listo para correr un comando

Las instancias de una imagen se conocen como contenedores

Docker: algunos detalles

En el ejemplo anterior:

- Tomamos un "computador base" con una distribución mínima de Linux (Alpine) que viene con Python instalado
- Copiamos nuestro código a ese computador base
- Corremos el comando `poetry run`...

Arquitectura del pipeline

En general, nuestros jobs van a ser imágenes de docker que cumplen funciones específicas:

- Copiar datos hacia el data lake
- Cargar los datos en el *data warehouse*
- Realizar un análisis sobre los datos del *warehouse*

Estas imágenes van a ser orquestadas por Airflow

¿Y las imágenes en qué lugar corren?

Kubernetes

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications

Kubernetes es un tópico en si mismo, pero estudiemos las nociones necesarias para construir un pipeline de datos

Kubernetes

La idea a grandes rasgos es que vamos a tener un *cluster* de Kubernetes que va a tener los recursos computacionales necesarios para correr nuestras imágenes

Airflow va a estar montado en un cluster de Kubernetes

Data Warehousing en la nube

Data Warehousing en la nube

Ya sabemos que un Data Warehouse es el lugar donde guardamos información procesada, lista para que los analistas puedan consumirla

Un Data Warehouse moderno se construye sobre un sistema en la nube como BigQuery, Snowflake o Redshift

En este curso vamos a aprender a usar BigQuery, pero todos estos sistemas tienen los mismos principios

Big Query

BigQuery is Google's fully managed, ***serverless data warehouse*** *that enables scalable analysis over petabytes of data*

Serverless

Serverless computing is a cloud computing execution model in which the cloud provider allocates machine resources on demand, taking care of the servers on behalf of their customers

Serverless

En un modelo serverless:

- No manejamos servidores, solo una interfaz de alto nivel
- No nos preocupamos de hacer escalar nuestro servicio, el proveedor de la nube lo hace solo
- En general mientras más recursos usamos más pagamos ($)

Data Warehouse en la Nube

Cuando usamos una herramienta de Data Warehouse en la nube:

- Tenemos espacio "ilimitado" (en la medida que podamos pagar)
- Tenemos recursos para ejecutar consultas de analítica **muy** pesadas
- Tenemos integraciones con diversos lenguajes/ programas

Data Warehouse en la Nube

Arquitectura

La arquitectura de un sistema de Data Warehouse en la nube se basa en un almacenamiento distribuido

Al hacer consultas comunicamos datos a un cluster de cómputo de alta disponibilidad

Al hacer una consulta disponemos de **muchos** "computadores pequeños" para responderla

¿Cómo podemos responder una consulta utilizando varios computadores pequeños?

Algoritmos en BigQuery

Big Query divide una consulta en varias consultas pequeñas

Cada sub-consulta se envía a un *slot* de trabajo distinto

En general, las consultas hacen hashing a los distintos servidores

Algoritmos en BigQuery

Joins (de la documentación de BQ)

Broadcast joins

- When joining a large table to a small table, BigQuery creates a broadcast join where the small table is sent to each slot processing the large table
- Even though the SQL query optimizer can determine which table should be on which side of the join, it is recommended to order joined tables appropriately

Algoritmos en BigQuery

Joins (de la documentación de BQ)

Hash joins

- When joining two large tables, BigQuery uses hash and shuffle operations to shuffle the left and right tables so that the matching keys end up in the same slot to perform a local join. This is an expensive operation since the data needs to be moved.

Algoritmos en BigQuery

Agregación

¿Podemos usar Hashing para responder consultas de agregación?

Algoritmos en BigQuery

Big Query no usa índices típicos para responder consultas

Cada vez que hacemos una consulta, cada tabla mencionada se carga en memoria completa al menos una vez

Estos sistemas se basan en la "capacitad ilimitada" de cómputo

¿Hay índices en BigQuery?

Particiones y Clustering

En BigQuery no hay indices como los tradicionales porque todos los datos se cargan **completamente en memoria**

Por lo tanto los algoritmos son distintos porque no se busca minimizar los accesos a disco

En teoría, BigQuery ejecuta algoritmos para hacer consultas de analítica eficientes sin necesidad de usar índices

Particiones

Partitioning: técnica que permite no cargar todos los datos en memoria, sino que un subconjunto

En general se aplica sobre campos de tipo fecha/timestamp

Es clave para ahorrar dinero ($) a la hora de hacer consultas sobre datos masivos

Cobros en BigQuery

Los cobros en BigQuery se basan en la cantidad de datos que se llevan a memoria; antes de ejecutar una consulta podemos saber cuantos datos se cargarán en memoria

Por defecto, todas las tablas se cargan completas en memoria aunque sean parte de un **WHERE**, por eso es importante particionar porque es la única forma de llevar solamente una parte de una tabla

Clustering

Clustering: técnica análoga a un índice multicolumna que permite ahorrar costos en ciertas consultas

Solo puede haber uno por tabla, de a lo más 4 columnas, en donde el orden importa en la misma forma que un índice multicolumna

Puede ayudar a reducir costos

Particiones y Clustering

Partitioning: técnica que permite no cargar todos los datos en memoria, sino que un subconjunto

En general se aplica sobre campos de tipo fecha/timestamp

Es clave para ahorrar dinero ($) a la hora de hacer consultas sobre datos masivos

¿Cómo consultamos tablas en BigQuery?

Big Query y SQL

Big Query usa SQL para responder consultas

Tiene algunos comandos específicos, pero en general es igual al SQL de toda la vida

Big Query y SQL

Además de hacer joins y agregaciones, existe un tipo de comandos muy importante en Data Warehousing

Estos comandos son las **Window Functions**, funciones que nos permiten hacer cierto tipo de analítica

Estudiaremos estas funciones con algunos ejemplos prácticos

Antipatrones

En BigQuery se desincentiva el uso de joins que no sean por igualdad, o de productos cruz (cross join)

También, hacer ORDER BY requiere materializar todos los resultados en un nodo, por lo mismo para resultados grandes no se recomienda

Procesamiento de Datos Masivos

Clase 03 - Data Warehousing