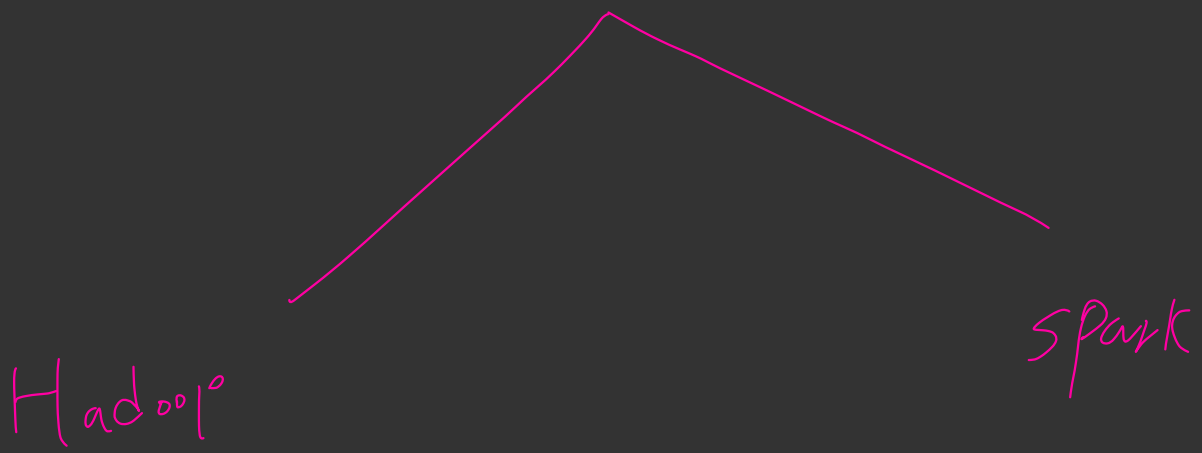
다수의 컴퓨터에 데이터 처리를 분산하기 위해, 실행관리 프레임워크가 필요하다



SQL로 데이터 질의시, 테이블의 컬럼명, 데이터 형, 관계등의 schema를 정의한다.
(table)

└ 구조화된 data

빅데이터는, 이미지데이터, 자연언어로 작성된 데이터, 동영상등... 데이터.

└ 비구조화된 data.

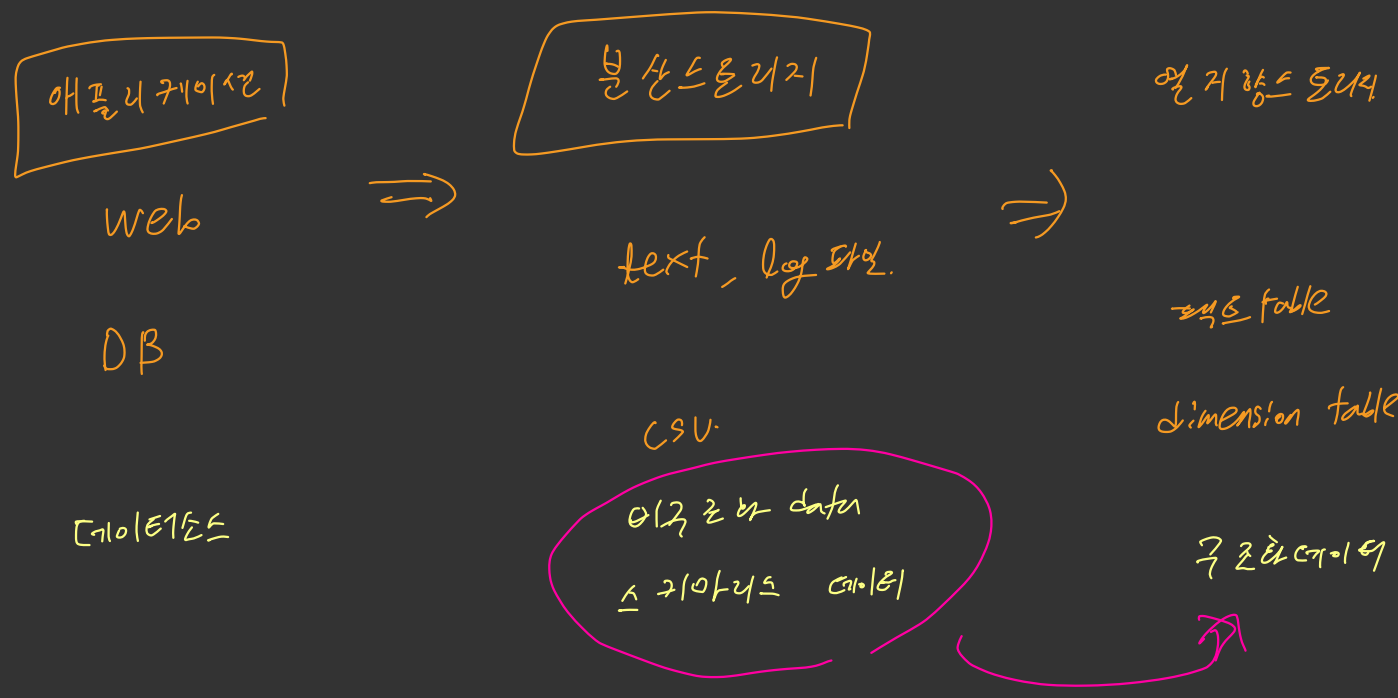
↳ 분산 스토리지에 저장, 분산시스템처리

↳ Data Lake 의개념

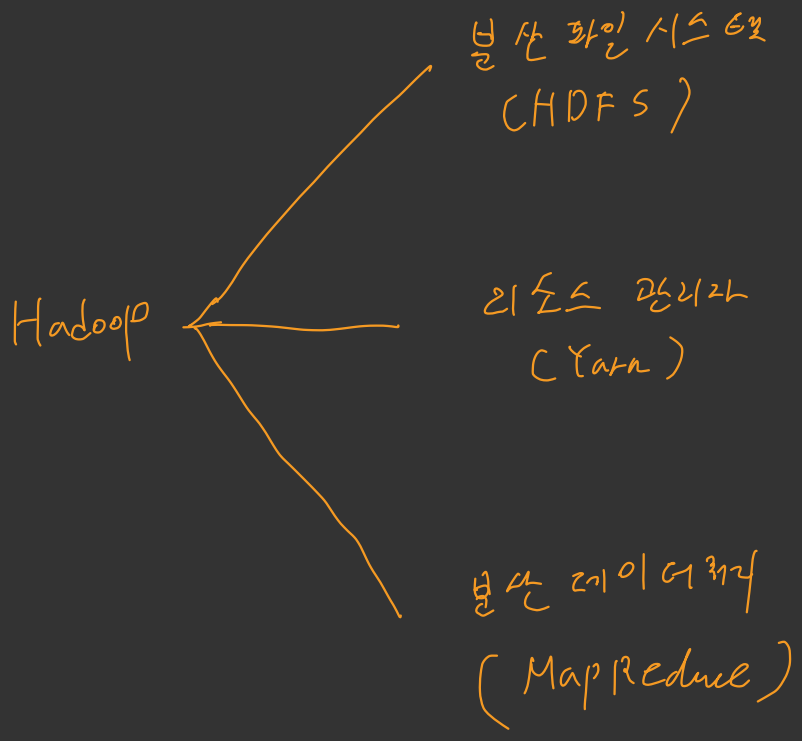
스키마리스데이터 - CSV, JSON, XML등의 데이터 서식은 있으나, 관리적,

데이터명은 명확하지 않은

데이터 구조화의 파이프라인



비구조화 데이터를 읽어들여 열지향 스토리지로 변환하는 과정 및 방법에 많은 컴퓨터 리소스 필요. ⇒ Hadoop Spark



모든 분산시스템이 Hadoop에 의존하는것이 아닌, 일부사용, Mesos, Spark등의 존재감

Hadoop에서의 처리되는 데이터 대부분은 분산파일 시스템인 HDFS에 저장.
↳ 다수의 컴퓨터에 파일을 복사하여 중복 성을 높임.

CPU, 메모리등의 계산 리소스는 리소스 매니저인 YARN에게 관리.
CPU 코어와 메모리를 컴퓨터가 몇개나 갖고있는지 관리.

YARN

Map Reduce
(비구조화 데이터가공)

SQL Hive 등의 확장언어로
데이터정제가 가능하나,
속도면에서 Tez가 나옴

핵심은 Hive on Tez로 보면됨.

Hive를 고속화하는것이 아닌,
처음부터 대화형 쿼리 실행만 전문으로
하는 쿼리엔진으로 개발되고있다.

ex) Apache Impala
Presto

YarN이 사용할 수 있는 리소스를
모두 활용.

SPARK - MapReduce 보다 더 효율적인 데이터 처리를 실현하는 프로젝트

- 대량의 메모리를 활용해 고속화한다.

MapReduce - 대부분을 디스크의 입출력에 사용. Tez와 마찬가지로,
데이터 처리과정에서 만들어진 중간 Data는 기본적으로 디스크에 기록

가능한한 많은 Data를 메모리상에 올려고 처리

SPARK — Hadoop(X) MapReduce 대체된 거.

ㄴ YARN, HDFS 등은 스파크에서 그대로 사용이 가능하다.

ㄴ S3, Cassandra로 대체 가능.

Hive에 의한 구조화 데이터 작성.

① CREATE EXTERNAL TABLE (외부 테이블 정의)

Hive는 데이터를 내부로 가져오지 않아도, 텍스트 파일 그대로 집계 가능.

CSV

ㄴ AD HOC에 용이, 매번 Read에서 속도에서 불리

↙ 열지향으로 변환.

ORC 형식으로 변환

Hive로 비정규화 테이블을 작성하기.

Table을 결합 및 집약해서

"비정규화" table 생성

ㄴ 데이터의 구조화가 완료되었으니 → 데이터 마트 구축.

(Presto 와 Hive)

모든 메모리를 사용..

ㄴ 시간 걸리면 무조건 Hive.

① 비정규화 Table을 만드는데 Runtime이 걸리는것을 분산 처리, 가능한 효율적인 쿼리를 작성해야한다.

↓

Hive
- ① 서브쿼리 안에서 레코드 수를 줄이는 방법
- ② 데이터 편향을 방지하는 방법

① 서브쿼리 안에서 레코드 수 줄이기.

└ SQL과 비슷하나, 완전한 RDB는 아니다.

Hive ≠ DB but 데이터처리를 위한 데이터구조.

ex) 비효율. — Table 결합 후 where 조건

```
select ...
from access_log a
join users b on b.id = a.user_id
where b.created_at = "2017-01-01"
```

ex) 효율 select ... — 처음에 서브로 필요 Table 걸러.

```
from (
    select * access_log
    where time >= timestamp "2017-01-01 ....." ) a
```

(join users...

where.

☆

초기 팩트 table을 작게 하는것이 중요하다!!!

☆ Group by로 데이터를 집계하기 전, SubQuery 안에서 중복제거를 하자.

데이터를 갖는 시점 추에 Table을 결합.

-distinct count.

② 데이터 편향 피하기 (order by에 의한 정렬 등...

L skewness

ex) 웹페이지당 고유 방문자수를 알고 싶을때, 조회수에는 큰 편차가 존재

ex) 비효율. - distinct count는 효율 X

select date, count (distinct user_id) users
from access_log group by date

효율 - 최초에 중복제거

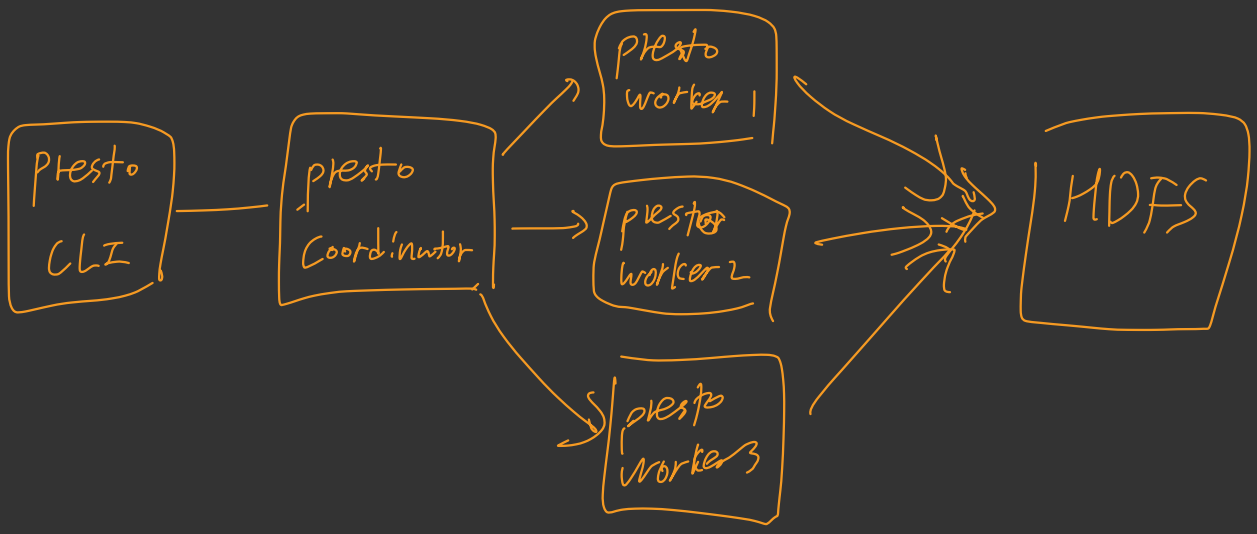
select date, count (*) users
from (
select distinct date, user_id from access_log) t
group by date

◎ 대화형 쿼리 엔진 Presto.

↔ MPP DB

Presto - 플러그인 가능한 스토리지 엔진.

└ 다양한 소스에서 적절데이터를 읽어들임 = Hive



Presto가 그 성능을 최대한 발휘하기위한 저장형식과 압축방식

데이터구조여야함.

↳ ORC 형식에 load에 최적.

Presto는 SQL실행에 특화된시스템. 쿼리를 분석해서 최적의 실행계획을 생성, 자체의 파이프라인으로 변환.

또는 Resource 를사용.

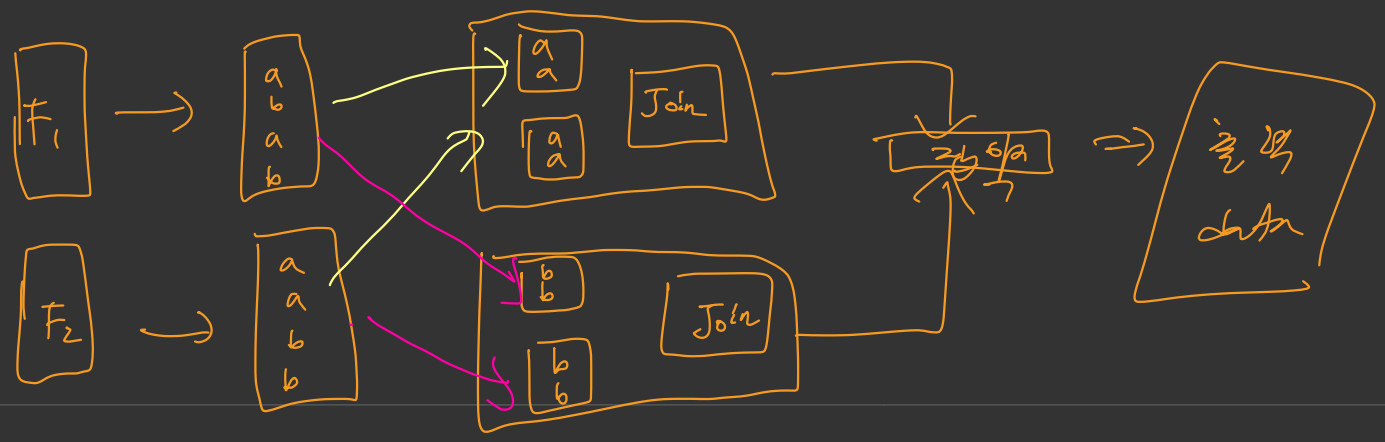
- CPU 처리의 최적화 - 멀티스레드라 => 단일머신에서 테스트 용이함
- 인메모리 처리에 의한 고속화 - 디스크 쓰기 X
  메모리상 데이터 처리.

---

① 분산결합과 브로드캐스트 결합

L 2개의 팩트 table을 결합하는경우 매칭되는 조인키를 메모리상에 계속유지해야함.

Presto - 분산결합을 실시 : 조인키를 갖는 data는 동일노드에 모인다



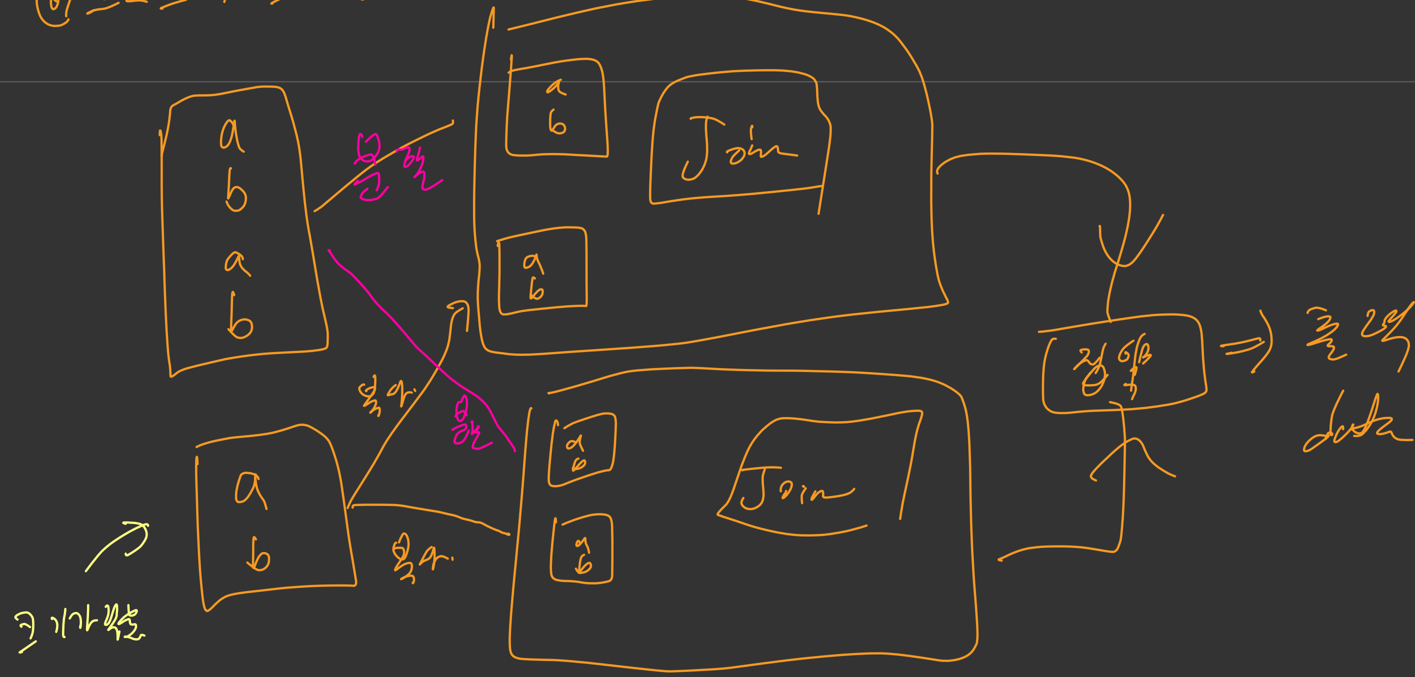
- 분산결합에서는 노드간의 데이터전송을 위한 네트워크 통신이 발생.
  중점 관리 지역.

- 한쪽테이블이 충분히 작은경우에는 브로드캐스트 결합 을 사용하여 처리속도를 고속화할 수 있다.

결합하는 table의 모든데이터가 각노드에모임.

① Presto에서 브로드캐스트결합시, 분산결합 무효화, fact table을 기준으로 디멘젼 table을 결합

⑧ 브로드캐스트 결합



---

⑨ 데이터 분석의 프레임워크 선택하기 (Hive, presto, Spark)

data 양에 좌우되지 않는 쿼리엔진

① Hive - 높은 확장성, 내결함성
- 대규모 배치 처리를 꾸준히 실행.
- text data 가공, 열지향 스토리지 만드는 등의 무거운 처리

② presto - 속도중시 and 대화식으로 특화된 쿼리엔진.

③ spark - 분산 시스템을 사용한 프로그래밍 환경
- └ 인 메모리 데이터 처리가 중요.
- └ ETL 프로세스에서 SQL까지 일련의 흐름을 하나의 data pipeline으로 기술 가능

└ 즉 text data 들어와서 ⇒ 관리형 스토리지로 변환 → SQL 로 정제 과정과 내보내는 등의 process 가 한방에!!!

---

① 데이터 파이프라인 구축.

└ 분산시스템이 필요하면, 시각화툴에서 데이터파이프를 만드는 절차에 들어간다.

② 팩트테이블 - 서비스 data 축적하기

└ 아주작으면 메모리에 올릴 수 있는데, 그렇지 않으면 관리형 스토리지에서 데이터를 불러와야함.

팩트 테이블의 작성에는 두 가지 방법이 있다.

① 추가 (append)

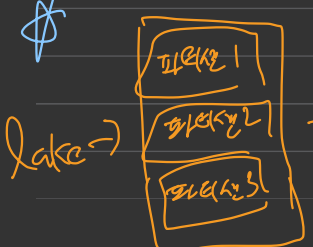
새로운 데이터만추가

문제점: 추가에실패했을경우 fact table

① 원복과정은.

② 추가중복시, 중복 데이터

③ 나중에 팩트table다시 만들고 실패건은 다시넣기.



② 치환 (replace)

전기의 data 포함해서 table 전체를 처리

1시간이내에

fact table 생성 가능시

치환처리를 해라.

summary table

① 집계테이블. — 팩트 테이블을 어느 정도 모아서 집계하면 데이터 양이 줄어든다

ㄴ 1일 보고서를 만드는데 주로 사용.

(숫자 계산에 주의!
(값이 축소되었기 때문)

① 카디널리티 — 각 칼럼이 취하는 값의 범위

ㄴ 성별 등은 값이 적고,

IP address는 카디널리티가 커진다.

ㄴ 국가등으로 변환해서 줄인다.

무리해서 줄이면, 원래 정보가 크게 손실.

---

① 스냅샷 테이블 — 마스터의 상태를 기록하기

ㄴ 정기적으로 테이블을 통째로 저장

ㄴ 시간에 주의!! 00:00:00 등... 겹치는 구간

스냅샷에서 매출부터 비정규화하는 것이 효율적

---

① 디멘션 추가해 비정규화 테이블 완성

ㄴ f table, d table 결합해서 비정규화.

ㄴ 중간중간 스냅샷 사용,

Ex) 처음 액세스 시간과, 마지막 액세스 시간을 정리함

└ 그러나! 새로 ID는 카디널리티가 너무 커서 table 형식,

시각화도 못 함... 그래서 cardinality가 적은 다른 열을 만들어 결합.