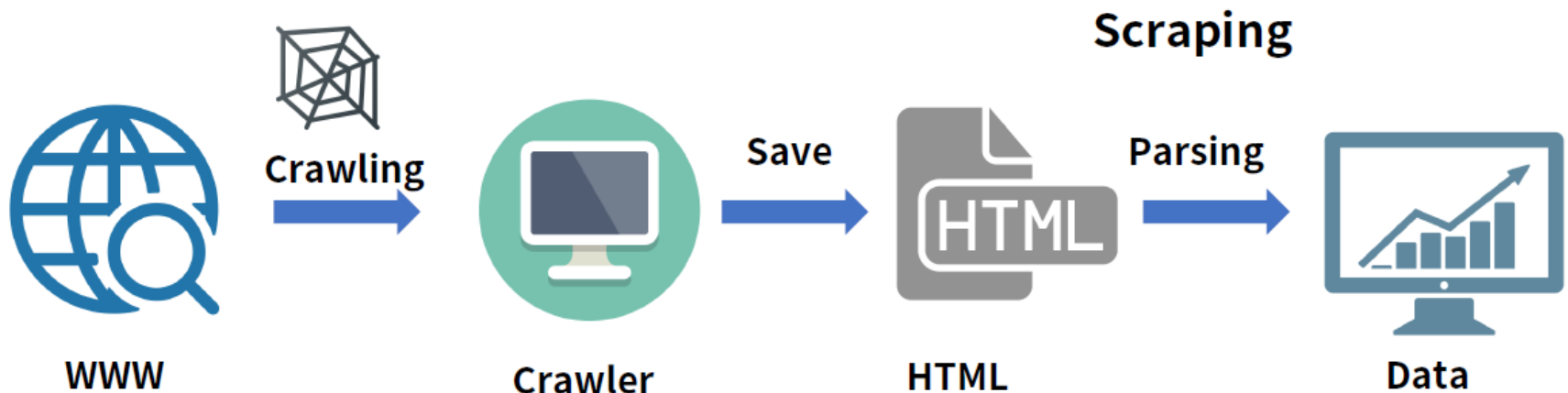


3강. 데이터 엔지니어링

1. 크롤링 (Crawling)

1.1 크롤링 (Crawling)

- ❖ 다양한 정보를 활용하기 쉽도록 수집하는 행위가 크롤링
- ❖ 크롤링을 하는 프로그램을 크롤러(Crawler)라고 함
- ❖ 웹의 데이터를 자동화해 가져오는 크롤러가 웹 크롤러(Web Crawler)




1. 크롤링 (Crawling)

1.2 HTML

❖ 하이퍼텍스트 마크업 언어(Hyper Text Markup Language)

❖ Web 페이지의 내용, 구조 등을 정의한 언어

[메인](#) | [출연&스텝](#) | [평점](#) | [관련영화](#) | [수상정보](#)



주먹왕 랄프 2: 인터넷 속으로 (2018)

Ralph Breaks the Internet

★★★★☆ 7.8/10

애니메이션/어드벤처/코미디/판타지 | 미국

2019.01.03 개봉 | 112분, 전체관람가

(감독) 필 존스톤, 리치 무어

(주연) 존 C. 라일리, 사라 실버맨, 권창욱, 소연

예매 4위 | 누적관객 1,220,996명

[예매하기](#)

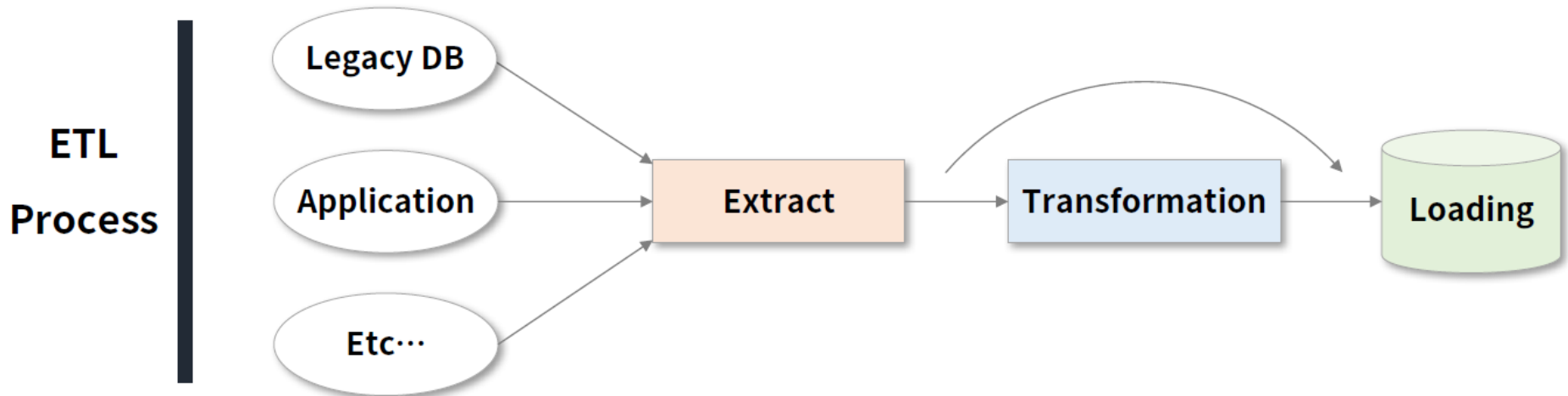
```
▼<div class="movie_basic">
  ▼<div class="main_detail">
    <!-- 2016-02-02 모바일 제목 -->
    ▶<div class="mobile_subject">...</div>
    <!-- //2016-02-02 모바일 제목 -->
    ▼<div class="detail_summarize">
      ▶<span class="thumb_summary #info #poster">...</span>
      ▼<div class="movie_summary">
        ▼<div class="subject_movie">
          <strong class="tit_movie">주먹왕 랄프 2: 인터넷 속으로 (2018)
          </strong> == $0
          <span class="txt_origin">Ralph Breaks the Internet</span>
          <!-- 2016-04-12 추가 -->
          ▶<a href="/moviedb/grade?movieId=118493" class="
            "raking_grade">...</a>
          </div>
          <!-- 2017-01-10 순서변경 및 수정 -->
          ▶<dl class="list_movie list_main">...</dl>
          ▶<dl class="list_placing">...</dl>
          ▶<div class="wrap_pbtn">...</div>
          ...
```

2. ETL

2.1 ETL

❖ Extract, Transformation, Loading

- ❖ 내외부의 다수의 데이터를 추출하고 이를 필요에 맞게 변환 후 저장하는 일련의 절차를 의미

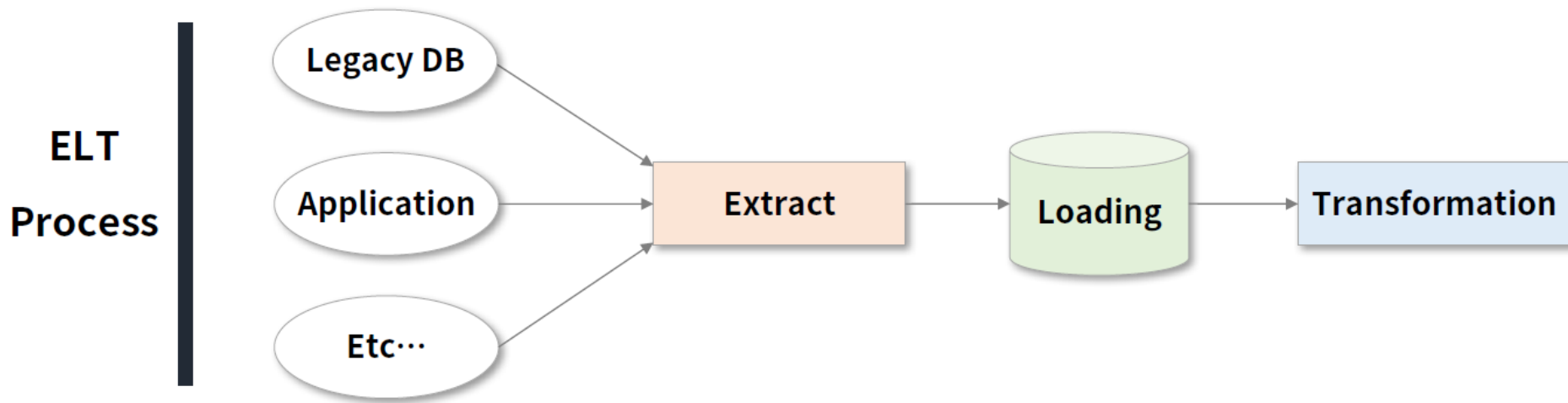


2. ETL

2.2 ELT

❖ Extract, Loading, Transformation

❖ ETL의 Transformation과 Loading의 순서를 바꿔 진행하는 절차를 의미





2. ETL

2.3 ETL 오픈소스 도구

- ❖ Talend
- ❖ Pentaho
- ❖ KNIME
- ❖ Apache NIFI
- ❖ StreamSets
- ❖ ...

3. 정형데이터/비정형데이터

3.1 정형데이터

❖ Structured Data

- ❖ 엑셀 등의 스프레드시트에서 작업하듯 **열과 행을 정리하여 일목요연하게 표**로 만들 수 있는 데이터
- ❖ 정형데이터 쉽게 다루기 위해 관계형 데이터베이스(RDB : Relational Database)가 활용되기도 함
- ❖ 정형 데이터를 File로 변환할 경우에는
 - CSV(Comma Separated Values)
 - TSV(Tab Separated Values)

3. 정형데이터/비정형데이터

3.2 정형데이터 구조와 예시

스키마에 의해 정의된 컬럼

column1	column2	column3	column4
data	data	data	data
data	data	data	data
data	data	data	data
data	data	data	data

컬럼에 의해 정의된 데이터

[그림1] 정형데이터 구조
Source : www.dbguide.net

Sepal.Length Sepal.Width Petal.Length Petal.Width Species

1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

[그림2] 정형데이터 예시
Source : www.dbguide.net

3. 정형데이터/비정형데이터

3.3 비정형데이터

❖ Unstructured Data

- ❖ 문서, 동영상, 사진, 음성 등의 **형태를 정의할 수 없는 데이터**
- ❖ 정형 데이터를 다루는 RDB에서 활용이 불가능 함
- ❖ 분석을 위해서는 비정형 데이터를 정형화하는 다양한 과정이 필요

3. 정형데이터/비정형데이터

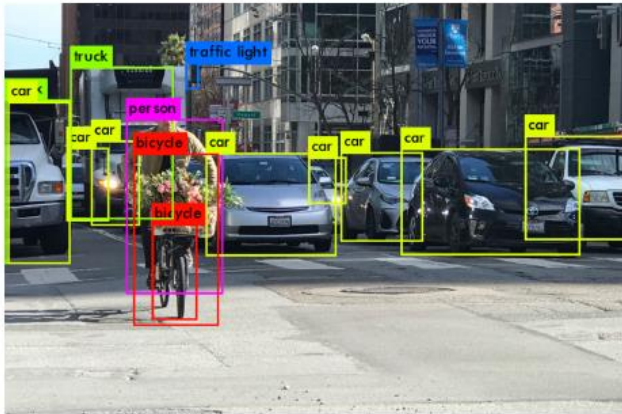
3.3 비정형데이터

❖ Unstructured Data

- ❖ 문서, 동영상, 사진, 음성 등의 **형태를 정의할 수 없는 데이터**
- ❖ 정형 데이터를 다루는 RDB에서 활용이 불가능 함
- ❖ 분석을 위해서는 비정형 데이터를 정형화하는 다양한 과정이 필요

3. 정형데이터/비정형데이터

3.4 비정형데이터 예시



[그림3] Image Object Detection
Source : medium.com



[그림4] Voice Data
Source : Information Age



[그림5] Text Mining
Source : 소셜마케팅코리아

3. 정형데이터/비정형데이터

3.5 반정형데이터

- ❖ **semi-structured data**
- ❖ 관계형 데이터베이스나 다른 형태의 데이터 테이블과 연결된 정형 구조의 데이터 모델을 준수하지 않는 정형 데이터의 한 형태이다.
- ❖ **각 의미를 구분할 수는 있지만 행과 열 형태의 표로 쉽게 정리가 어려움**
 - ➔ 파싱(Parsing) 필요
- ❖ JSON, XML, HTML

3. 정형데이터/비정형데이터

3.6 반정형데이터 예시

```
<div class="movie_basic">
  <div class="main_detail">
    <!-- 2016-02-02 모바일 제목 -->
    <div class="mobile_subject">...</div>
    <!-- //2016-02-02 모바일 제목 -->
    <div class="detail_summarize">
      <span class="thumb_summary #info #poster">...</span>
      <div class="movie_summary">
        <div class="subject_movie">
          <strong class="tit_movie">주먹왕 랄프 2: 인터넷 속으로 (2018)
          </strong> == $0
          <span class="txt_origin">Ralph Breaks the Internet</span>
          <!-- 2016-04-12 추가 -->
          <a href="/moviedb/grade?movieId=118493" class="
            "raking_grade">...</a>
          </div>
          <!-- 2017-01-10 순서변경 및 수정 -->
          <dl class="list_movie list_main">...</dl>
          <dl class="list_placing">...</dl>
        </div class="wrap_pbtn">...</div>
      ...
    </div>
  </div>
</div>
```

[그림6] html
Source : Movie.daum.net

```
{
  "orders": [
    {
      "orderno": "748745375",
      "date": "June 30, 2088 1:54:23 AM",
      "trackingno": "TN0039291",
      "custid": "11045",
      "customer": [
        {
          "custid": "11045",
          "fname": "Sue",
          "lname": "Hatfield",
          "address": "1409 Silver Street",
          "city": "Ashland",
          "state": "NE",
          "zip": "68003"
        }
      ]
    }
  ]
}
```

[그림7] JSON
Source : GoAnywhere

```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
  more questions later.-->
</quiz>
```

[그림8] XML
Source : en.wikipedia.org

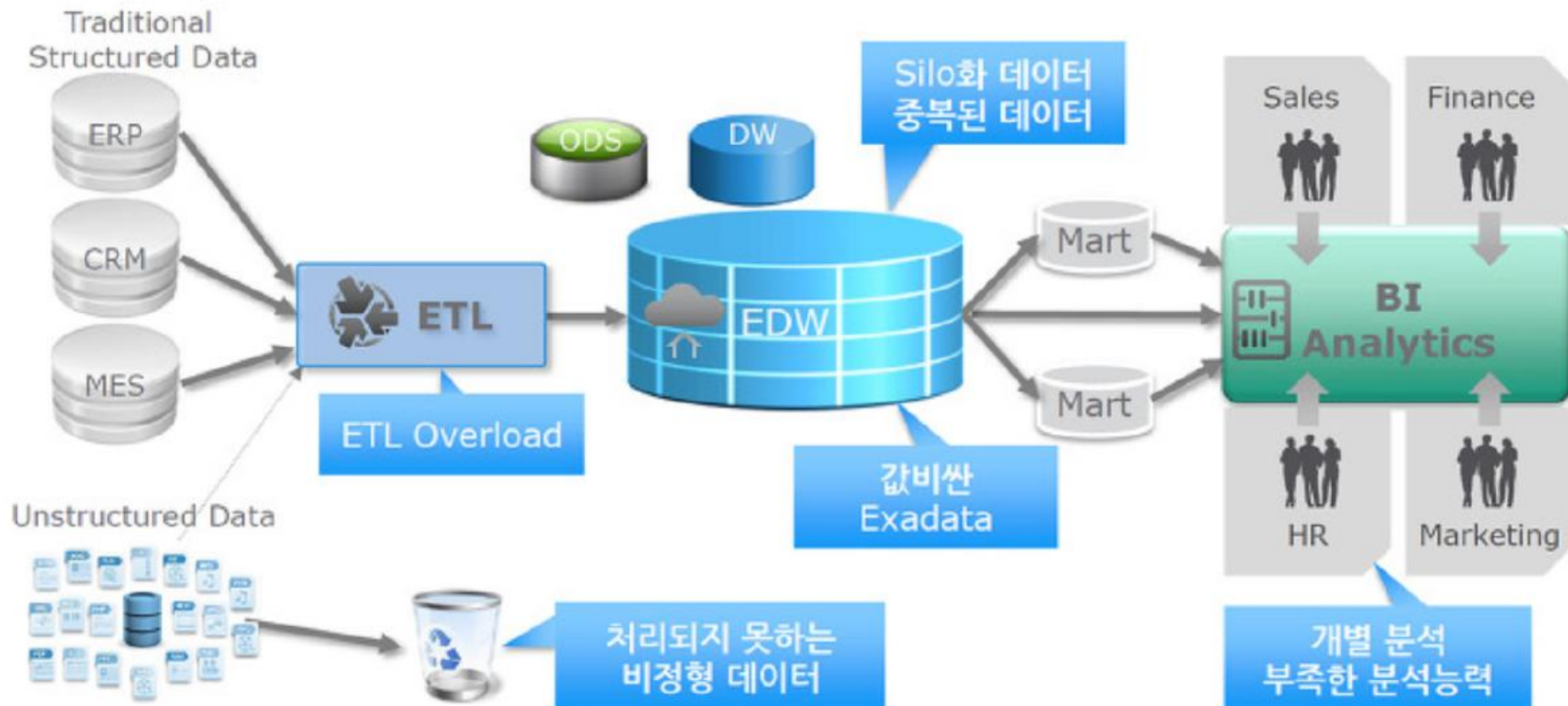
4. Data Warehouse vs Data Lake

4.1 Data Warehouse

- ❖ 사용자의 의사 결정에 도움을 주기 위하여, 여러 시스템의 데이터베이스에 축적된 데이터를 공통의 형식으로 변환해서 관리하는 데이터베이스를 말함
- ❖ 짧게 줄여 DW라고도 함
- ❖ DW를 구축 한다는 것은?
 - 다양한 분석을 할 수 있도록 여러 테이블을 가공해 Mart(일종의 주제별 테이블)를 생성
 - 이후 사용자는 분석 툴(e.g. OLAP, SQL)을 활용해 보다 편리하게 분석할 수 있도록 함

4. Data Warehouse vs Data Lake

4.2 Data Warehouse Architecture



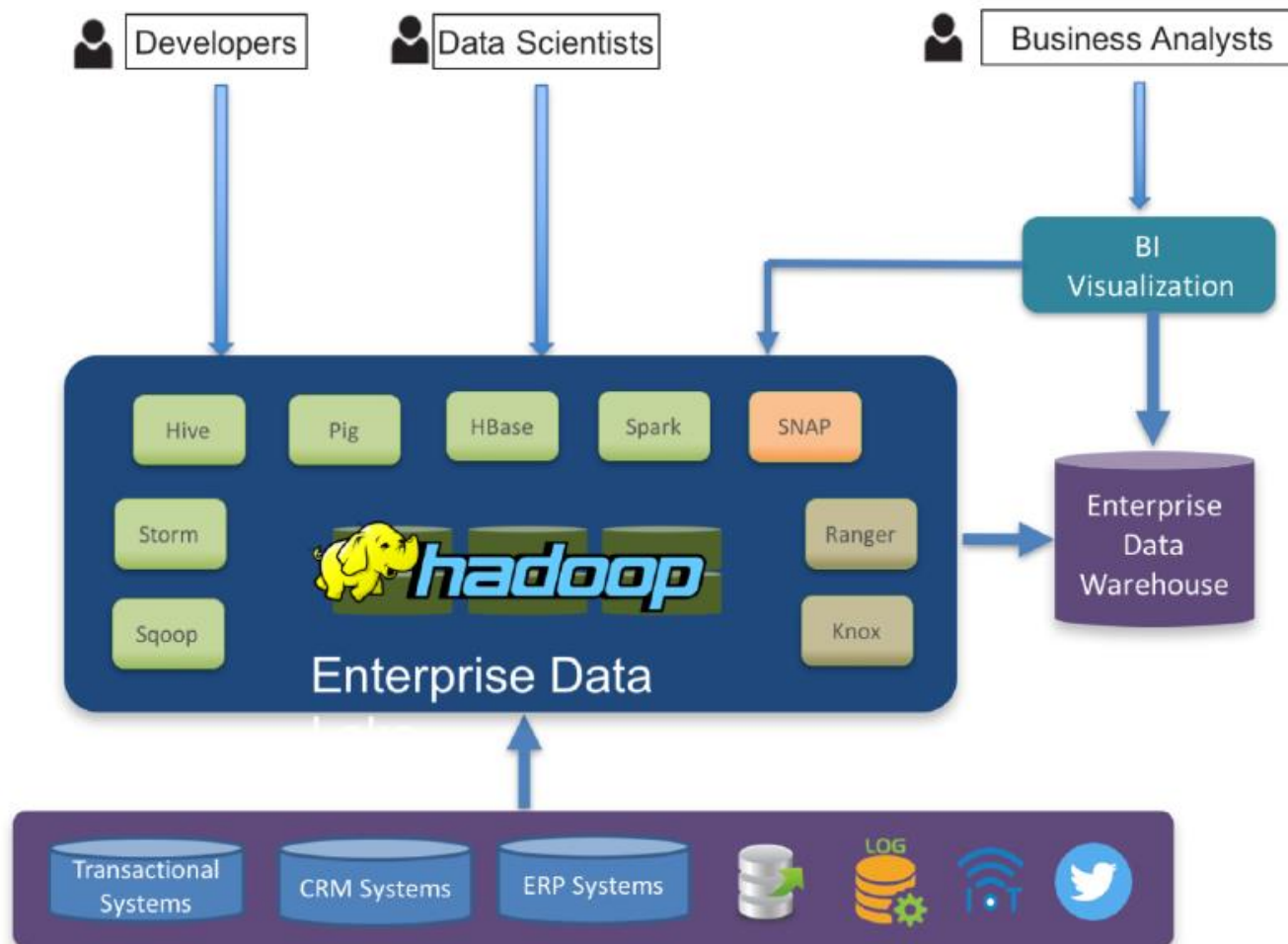
4. Data Warehouse vs Data Lake

4.3 Data Lake

- ❖ Data Warehouse의 정보 분석 한계를 개선하기 위해 생긴 데이터 저장 개념
- ❖ Data Warehouse는 정형데이터를 가공하여 저장
- ❖ Data Lake는 원래 형식대로 대량 저장하여 분석에서 활용
- ❖ 정형, 비정형, 반정형 모두 적재
- ❖ Data Warehouse에 비해 더 큰 빅데이터 저장 공간 필요

4. Data Warehouse vs Data Lake

4.4 Data Lake Architecture





5. 데이터 스트림(Stream)과 배치(Batch)

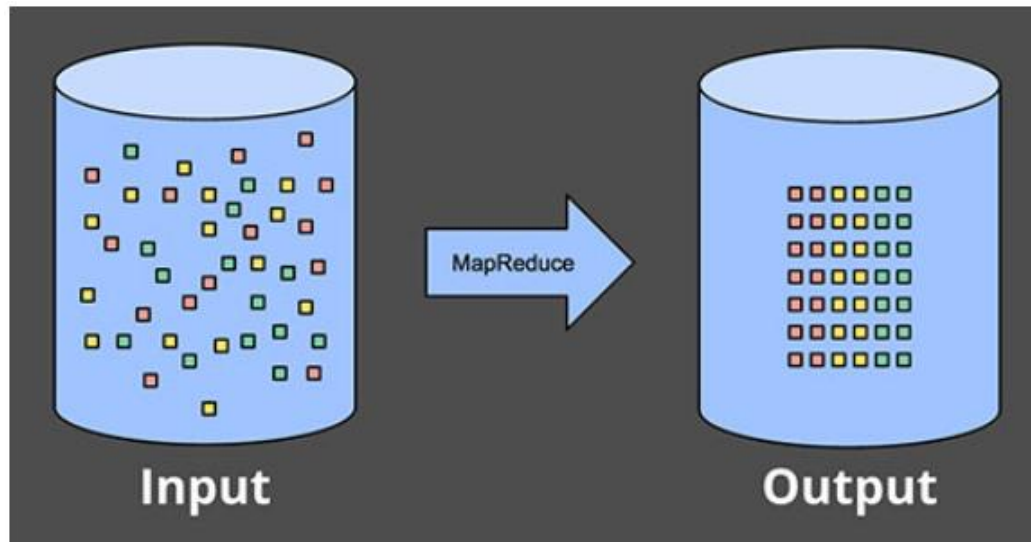
5.1 데이터 적재 유형

- ❖ Bounded Data
- ❖ Unbounded Data

5. 데이터 스트림(Stream)과 배치(Batch)

5.2 Bounded Data 처리

- ❖ 일단 저장되면 이후 변화가 없는 데이터
- ❖ e.g. 매 월 단위 매출 데이터, 매월 신규 고객 유치 수 등
- ❖ 처리는 묶어서 한 번에 일괄 처리



5. 데이터 스트림(Stream)과 배치(Batch)

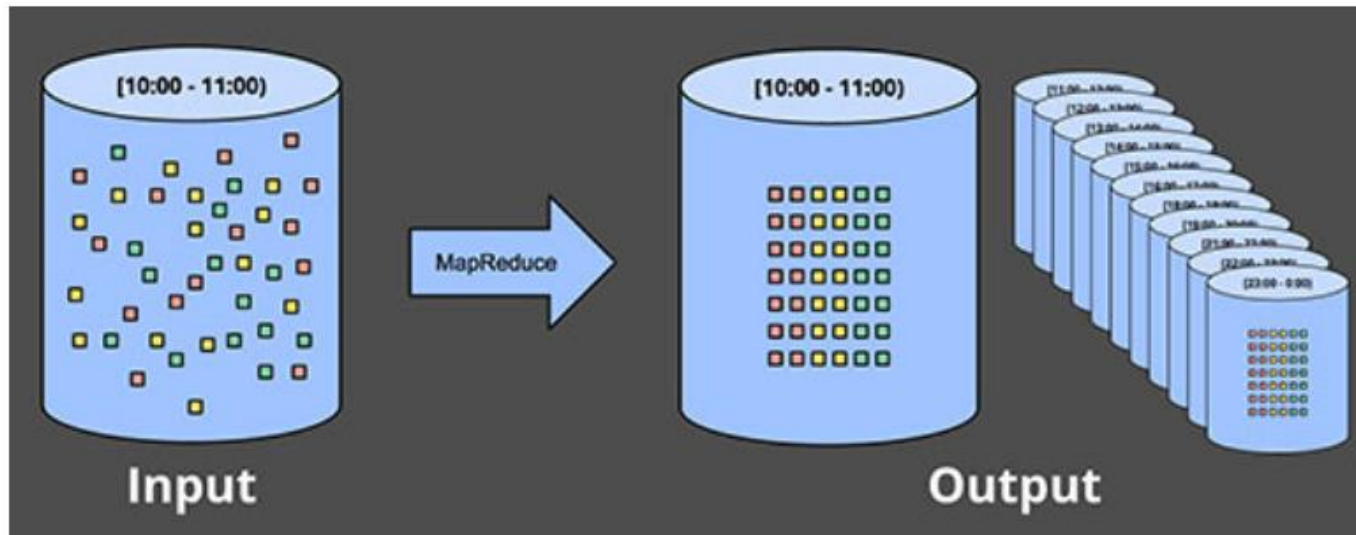
5.3 Unbounded Data 처리

- ❖ Bounded Data 와는 달리, 끝을 지정할 수 없는 지속적으로 적재되는 데이터
- ❖ e.g. 시스템 로그 데이터, 주식 가격 변동 데이터 등
- ❖ 끝이 정해지지 않기 때문에 주기적 처리 또는 실시간 처리함

5. 데이터 스트림(Stream)과 배치(Batch)

5.4 배치 처리(Batch Processing)

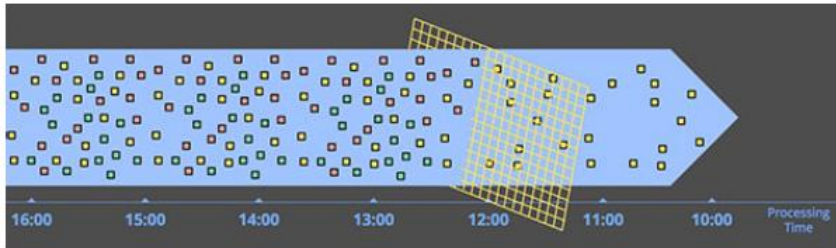
- ❖ (= 일괄처리)
- ❖ 대량의 데이터를 특정 시간에 한번에 처리하는 것을 의미
- ❖ Daily Batch, Hourly Batch 등의 형태



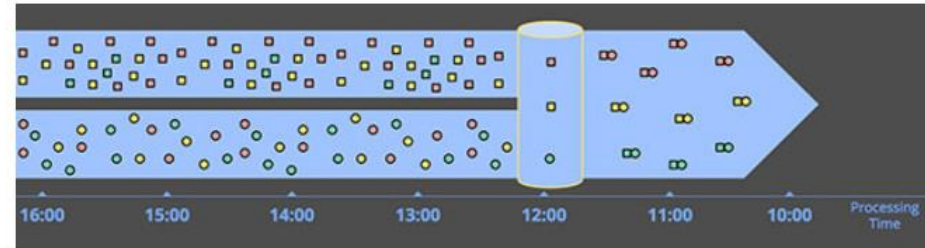
5. 데이터 스트림(Stream)과 배치(Batch)

5.5 스트림 처리(Stream Processing)

- ❖ Stream, 영어 뜻으로는 개울, 시내
- ❖ 물의 흐름처럼 지속적으로 유입되는 데이터의 연속성 있는 처리
- ❖ 처리 예시



Filtering



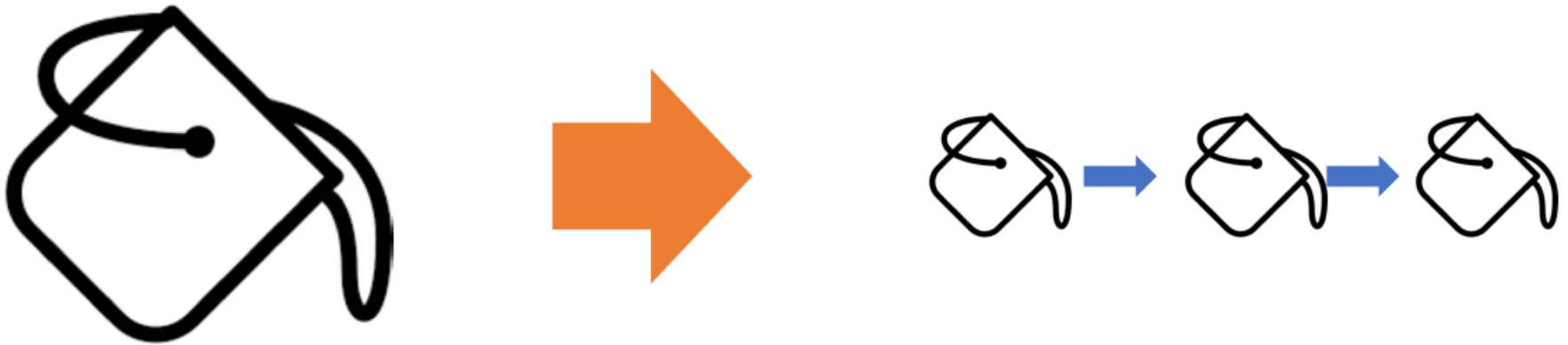
Inner-Joins

5. 데이터 스트림(Stream)과 배치(Batch)

5.5 스트림 처리(Stream Processing)

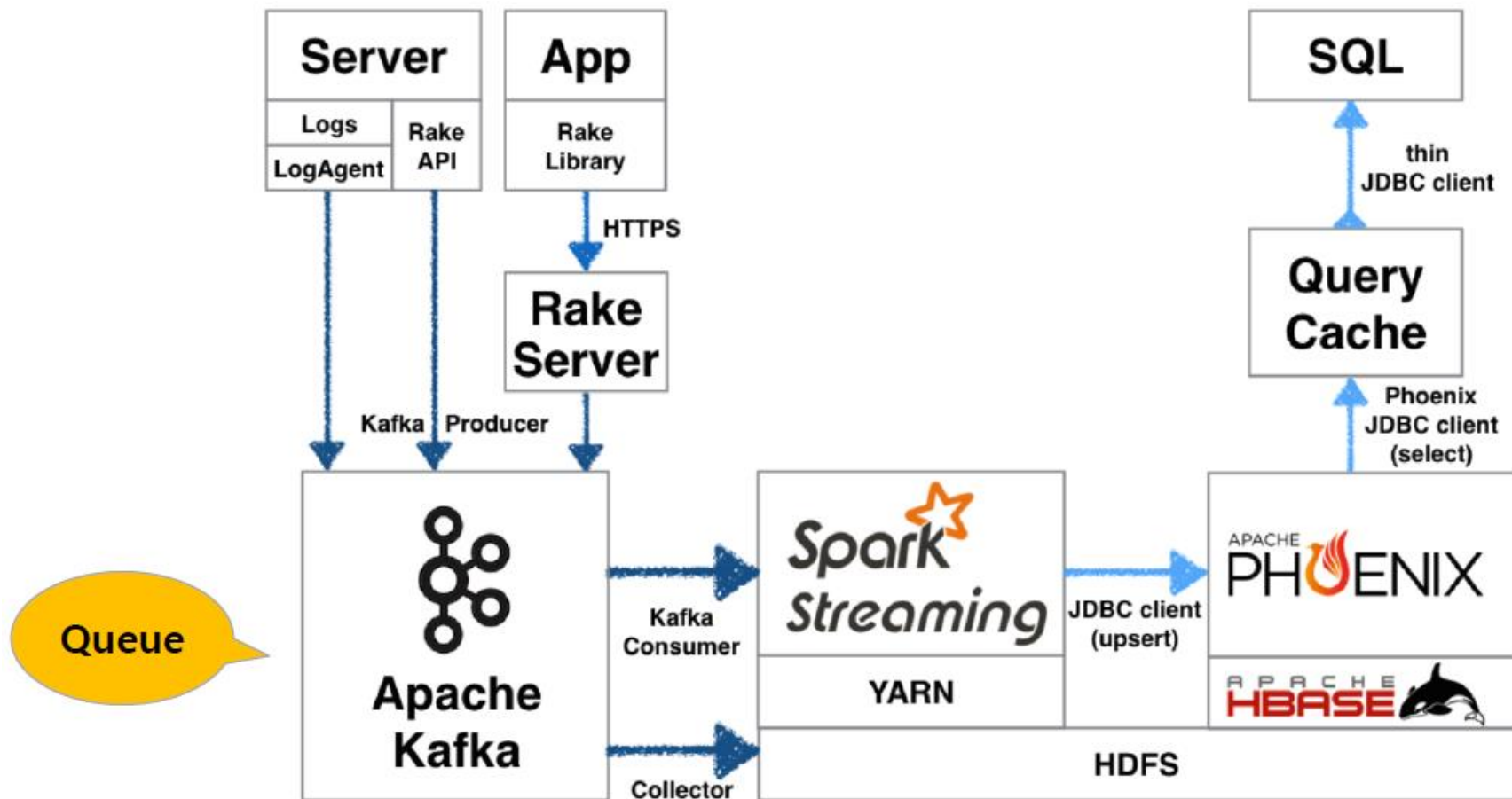
❖ Micro Batch

- 배치의 주기나 데이터 크기를 상대적으로 짧게 설정하여 준 실시간으로 처리 하는 것을 의미 (일종의 스트림 처리)



5. 데이터 스트림(Stream)과 배치(Batch)

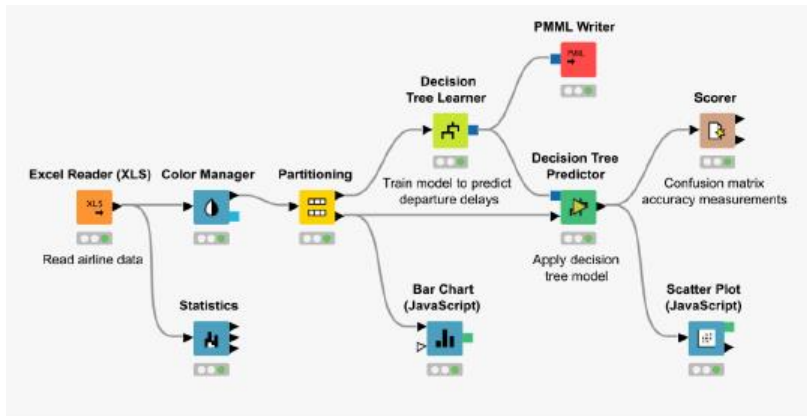
5.6 스트림 처리 인프라 아키텍처 예시



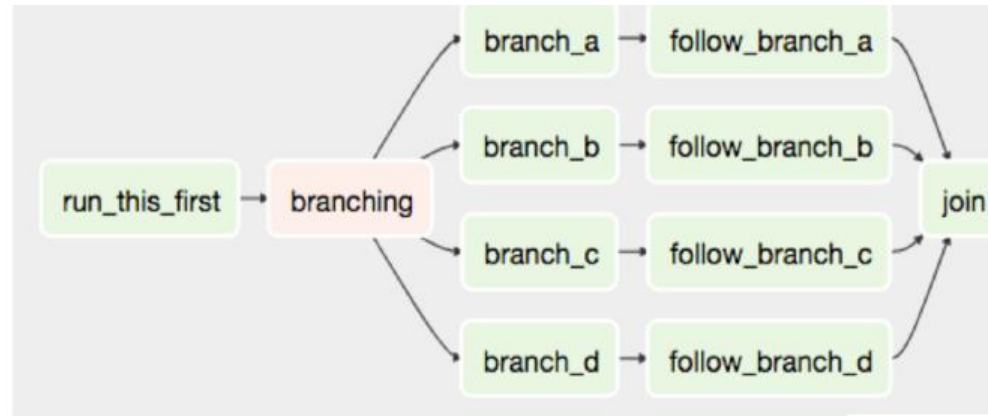
6. 워크플로우 (Workflow)

6.1 워크플로우 (Workflow)

- ❖ 작업 절차를 통한 정보 또는 업무의 이동을 의미 : 작업 흐름
- ❖ 데이터 워크플로우는 데이터 처리의 작업 절차를 의미
- ❖ ETL 작업같은 데이터 처리 흐름을 워크플로우 스케줄링 툴로 처리함



Source : knime

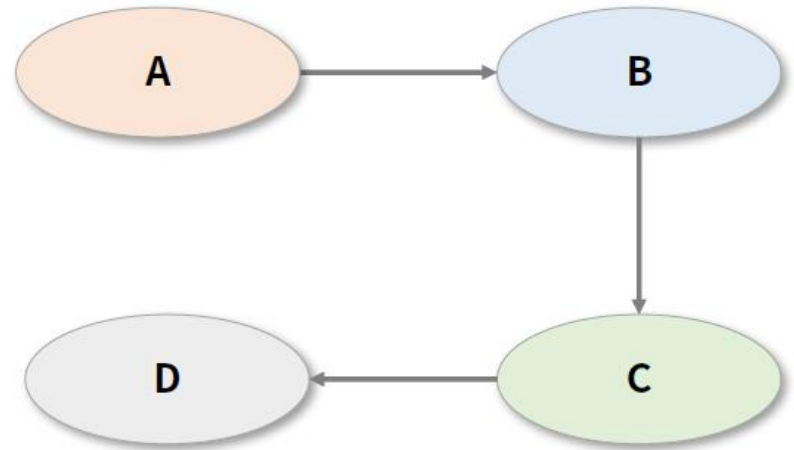
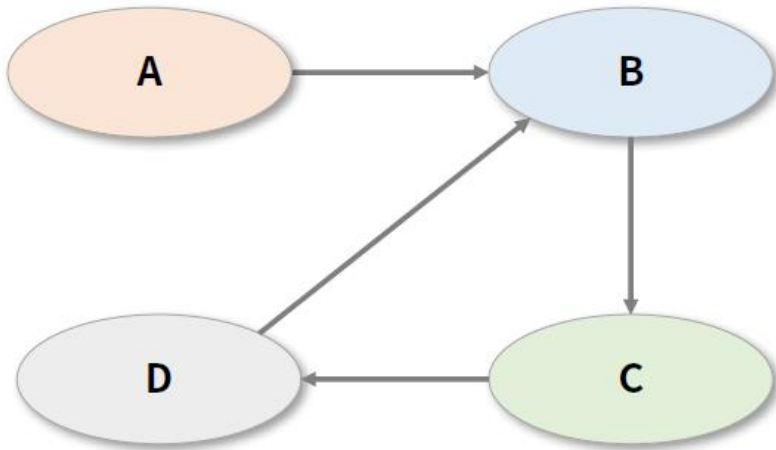


Source : michal.karzynski.pl

6. 워크플로우 (Workflow)

6.2 DAG

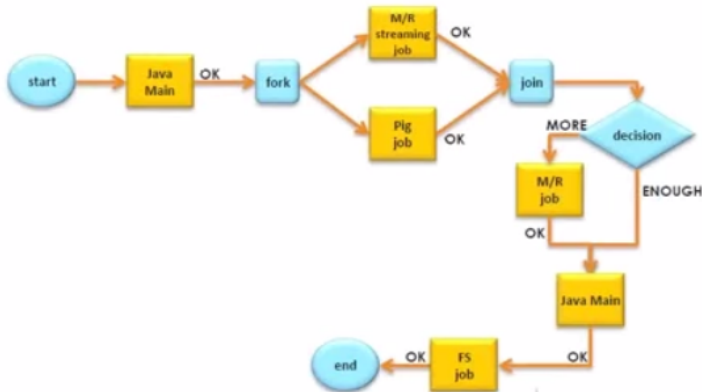
❖ 방향성 비순환 그래프, Directed Acyclic Graph



6. 워크플로우 (Workflow)

6.3 Apache Oozie

- ❖ Oozie is a workflow scheduler system to manage Apache Hadoop jobs



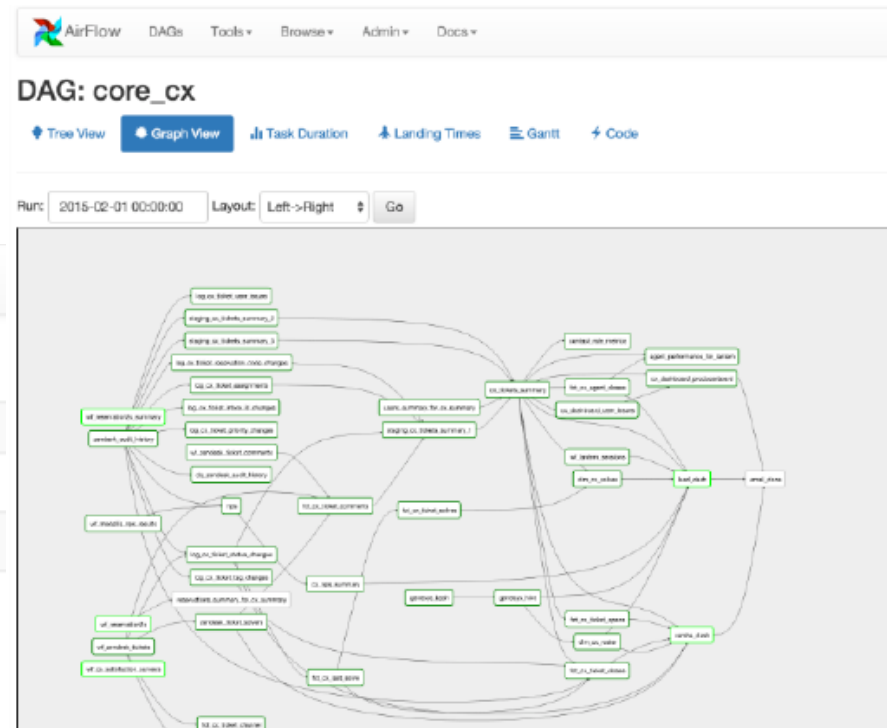
```
<workflow-app name='wordcount-wf' xmlns="uri:oozie:workflow:0.1">
  <start to='wordcount'/>
  <action name='wordcount'>
    <map-reduce>
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <configuration>
        <property>
          <name>mapred.mapper.class</name>
          <value>org.myorg.WordCount.Map</value>
        </property>
        <property>
          <name>mapred.reducer.class</name>
          <value>org.myorg.WordCount.Reduce</value>
        </property>
      </configuration>
    </map-reduce>
  </action>
</workflow-app>
```

6. 워크플로우 (Workflow)

6.4 Apache Airflow

- ❖ 에어비앤비(Airbnb) 엔지니어링팀에서 개발된 도구
- ❖ **Python 기반으로 워크플로우 설계 가능**
- ❖ 여러 머신에서 분산하여 수행
- ❖ UI 기반의 모니터링 도구 제공

AirFlow DAGs Tools Browse Admin Docs				
DAGs				
DAG	Filepath	Owner	Task by State	Links
example1	example_dags/example1.py	airflow	80 1 0	Tree View Graph View Task Duration Landing Times Gantt Code
example2	example_dags/example2.py	airflow	128 10 0	
example3	example_dags/example3.py	airflow	138 5 0	



7. Computer Cluster

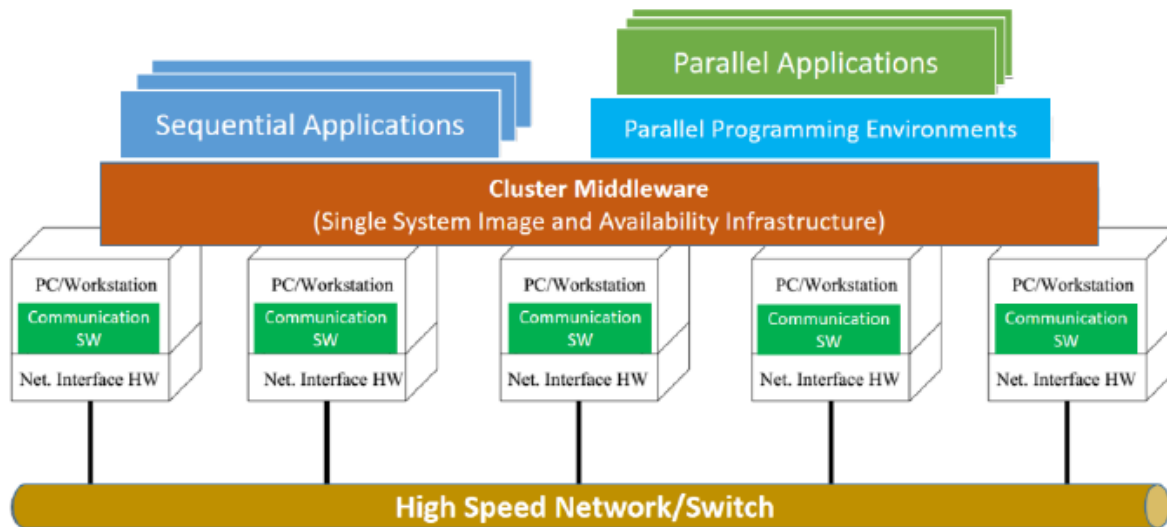
7.1 Computer Cluster

- ❖ 여러 대의 컴퓨터들이 연결되어 하나의 시스템처럼 동작하는 컴퓨터들의 집합을 말함
- ❖ 물리적으로는 여러 대의 컴퓨터이지만 외부 사용자는 마치 한 대의 컴퓨터인 것으로 보임
- ❖ 클러스터의 구성요소
 - Node(Master + Slave)
 - Network
 - OS
 - Middleware

7. Computer Cluster

7.2 Computer Cluster의 목적

- ❖ 서버의 확장을 통한 우수한 성능을 얻을 수 있음
- ❖ 분산 컴퓨팅

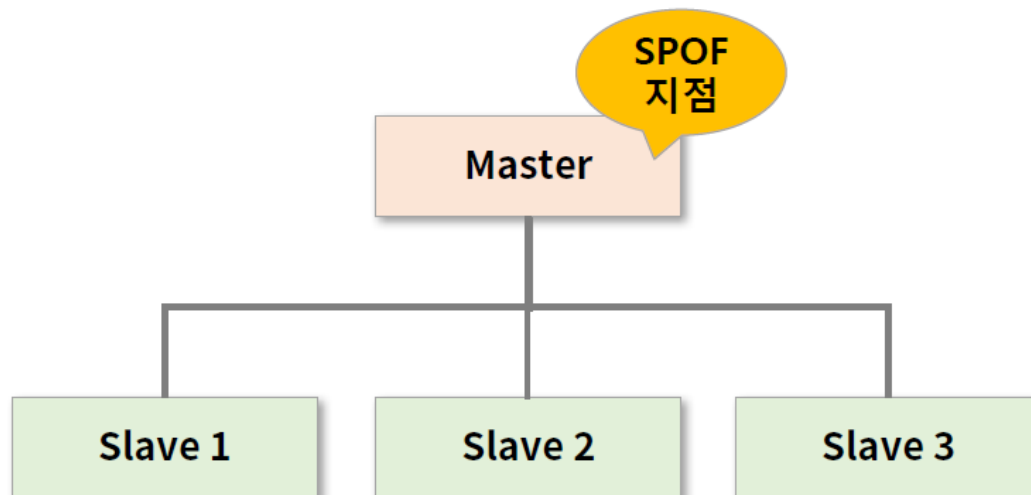


Computer Cluster Architecture

7. Computer Cluster

7.3 SPOF(Single point of failure)

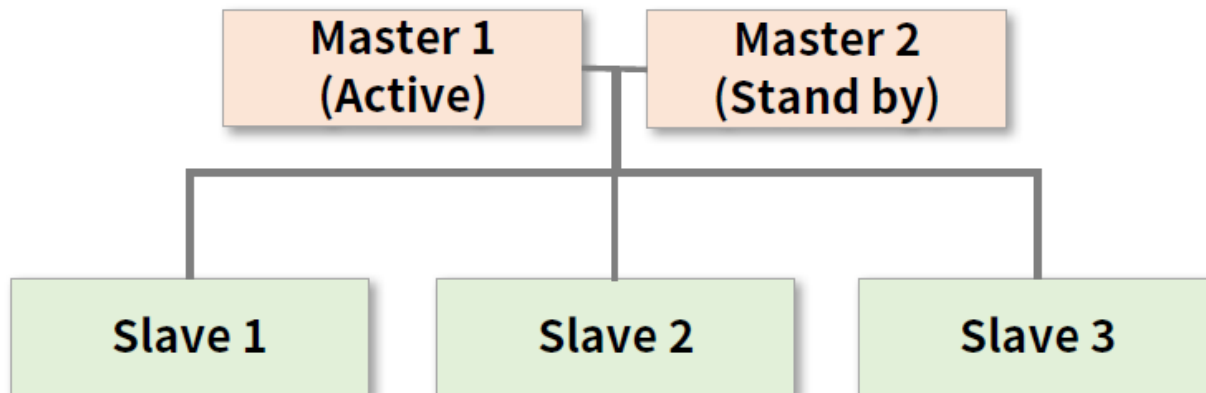
- ❖ 단일 장애지점
- ❖ 시스템 구성 중에 동작하지 않으면 전체 시스템이 중단되는 요소
- ❖ 일종의 치명적인 약점으로 클러스터에서 SPOF가 없어야 함



7. Computer Cluster

7.4 HA(High Availability)

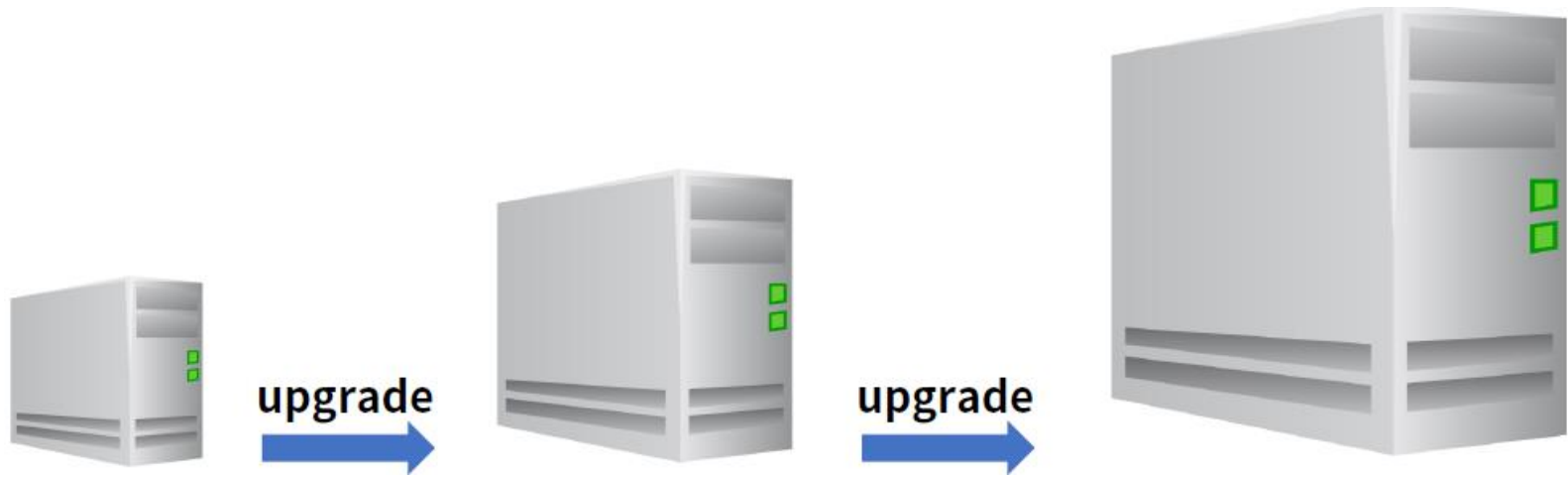
- ❖ 고가용성 (高可用性)
- ❖ 서버와 네트워크, 프로그램 등의 시스템이 지속적으로 운영이 가능한 성질
- ❖ 클러스터에서는 SPOF가 없어야 하며, HA가 유지되어야 함



8. Scale Up and Scale OUT

8.1 Scale Up

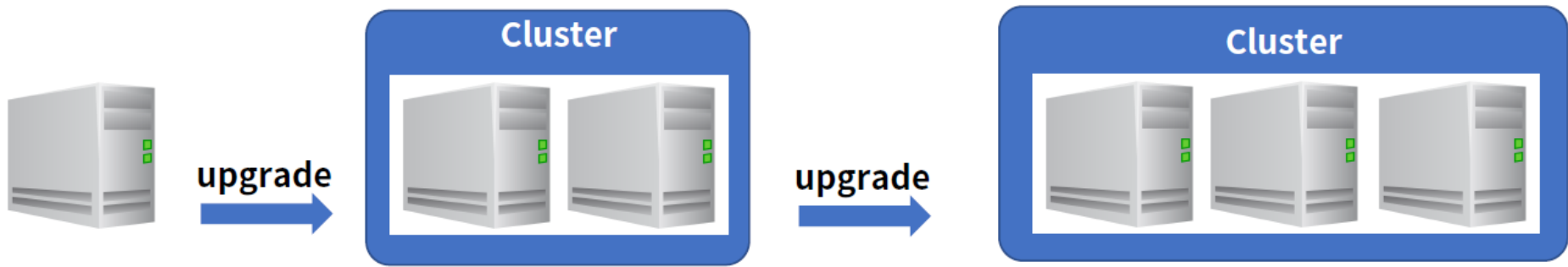
- ❖ 서버(또는 컴퓨터) 자체 RAM, CPU, Disk 등의 구성요소 자체를 업그레이드 하여 컴퓨팅 성능을 향상 시키는 방법
- ❖ 1의 처리 능력을 가진 서버를 5의 능력을 가진 서버로 기능을 향상 시키는 것



8. Scale Up and Scale OUT

8.2 Scale Out

- ❖ 네트워크 상의 서버(또는 컴퓨터)의 수를 늘려 컴퓨팅 성능을 향상 시키는 방법
- ❖ 1대의 컴퓨터가 하던 일을 5대의 컴퓨터가 나눠서 처리함
- ❖ 각 서버를 Cluster로 묶기 때문에 사용자는 1대를 활용하는 것으로 느낌



8. Scale Up and Scale OUT

8.3 Scale Up 과 Scale Out 비교

구분	Scale Up 형태의 서버구성	Scale Out 형태의 서버 구성
확장	RAM, CPU 등의 하드웨어 서버의 성능을 올림	하나의 서버에서 하던 일을 여러 대의 서버에서 처리
제약사항	무한 확장이 불가능함	이론상으로 무한 확장 가능
비용	성능 증가에 따라 가격이 급등함	비교적 저렴한 서버를 여러 대 사용하여 비용부담이 덜한 편
운영	확장에 따른 큰 변화 없음	서버가 늘기 때문에 설치 장소의 공간확보가 필요
장애관련	부하가 한 대의 서버에 집중할 수 있음	서버 간 네트워크 비용이 증가할 수 있음. 장애발생시 어느 서버에서 발생했는지 확인이 필요함
구성요소	단일서버	Master + Slave

9. SQL

9.1 SQL

- ❖ 구조적 질의 언어(Structured Query Language)
- ❖ SQL은 관계형 데이터베이스의 데이터를 관리하기 위해 설계된 특수 목적의 프로그래밍 언어
- ❖ 데이터베이스에서 정형 데이터 분석을 위한 필수 요소
- ❖ 내부적으로는
 - 데이터 정의 언어 (**DDL**, Data Definition Language)
 - 데이터 조작 언어 (**DML**, Data Manipulation Language)
 - 데이터 제어 언어 (**DCL**, Data Control Language)
 - 트랜잭션 제어 언어 (**TCL**, Transaction Control Language)

9. SQL

9.2 DDL (Data Definition Language)

- ❖ 데이터정의 언어
- ❖ 관계형 데이터베이스에 테이블 구조를 정의하고 생성하거나 기존의 테이블의 구조 변경 또는 삭제하는 명령어
- ❖ 세부 명령어
 - CREATE : 새로운 데이터베이스, 테이블의 생성
 - ALTER : 기존 테이블 구조의 변경
 - DROP : 기존 데이터베이스, 테이블의 삭제

9. SQL

9.3 CREATE 구문 이해

```
CREATE TABLE {테이블명}
(  
  {컬럼A} {속성},  
  {컬럼B} {속성},  
  {컬럼C} {속성},  
  ...  
);
```



```
CREATE TABLE test_results
(  
  name  STRING,  
  student_id INT,  
  birth_date DATE,  
  grade  DOUBLE,  
  passwd STRING  
);
```

9. SQL

9.4 DML (Data Manipulation Language)

- ❖ 데이터 조작 언어
- ❖ 데이터베이스의 테이블에 들어있는 데이터들을 조회하거나 변경하는 명령어
- ❖ 세부 명령어
 - SELECT : 데이터의 검색
 - INSERT : 새로운 데이터의 삽입
 - DELETE : 기존 데이터의 삭제
 - UPDATE : 기존 데이터의 수정

9. SQL

9.5 DML (Data Manipulation Language)

❖ SELECT 구문 이해

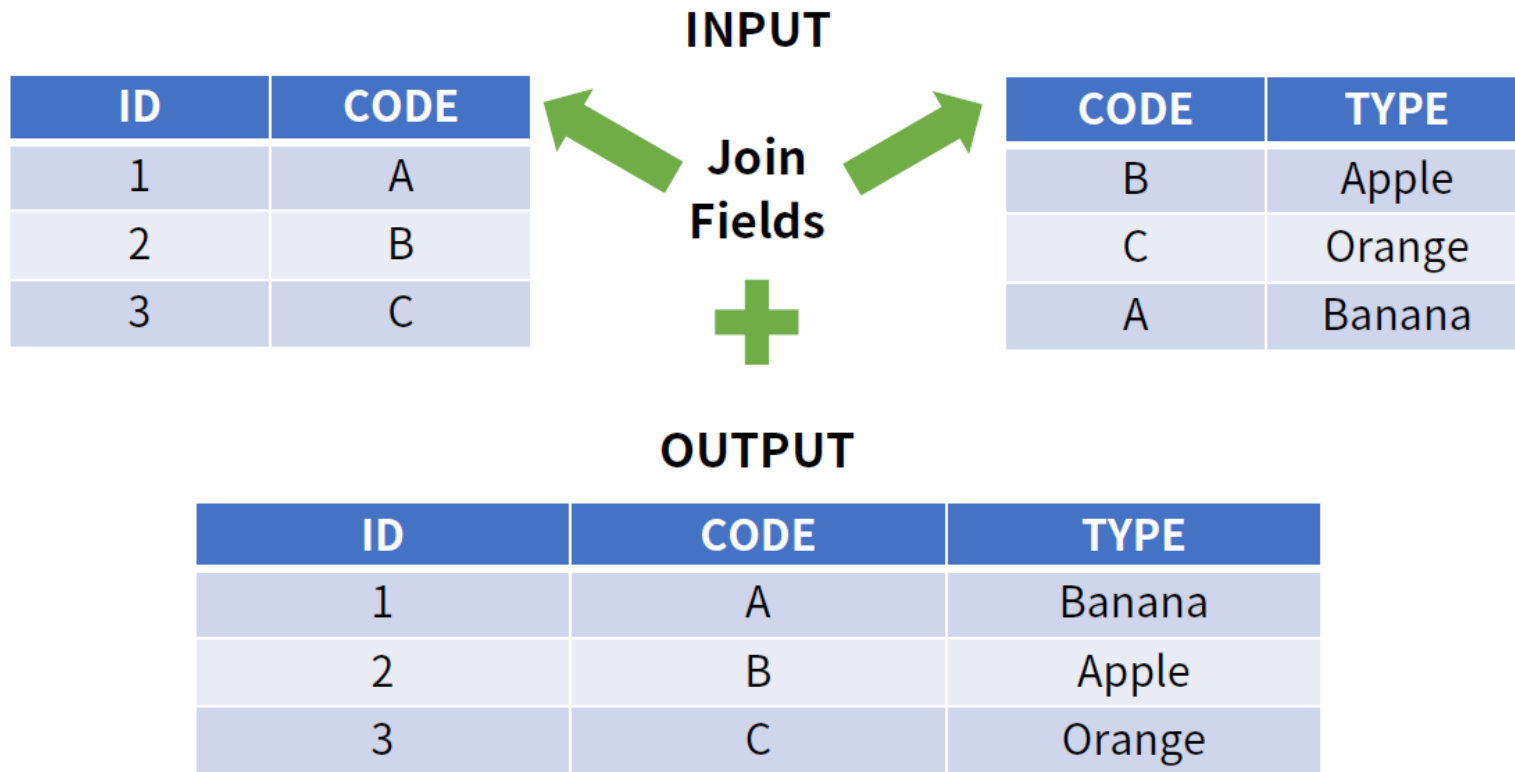
```
SELECT [DISTINCT] 필드명[, 필드명, ...]  
FROM 테이블명  
[WHERE 검색조건]  
[ORDER BY 필드명 [ASC or DESC]]  
[GROUP BY 필드명[, 필드명, ...]]  
[HAVING 검색조건]
```

```
SELECT year, month, date, name  
FROM source_table  
WHERE country = 'KR'  
      AND year = 2014  
      AND month = 05  
      AND day = 30;
```


9. SQL

9.5 DML (Data Manipulation Language)

❖ 데이터 JOIN 이해

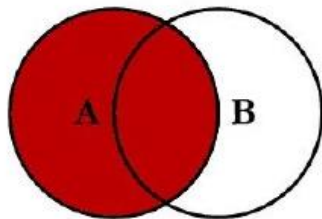


9. SQL

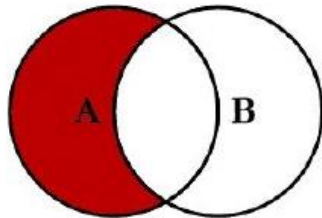
9.5 DML (Data Manipulation Language)

❖ JOIN 종류

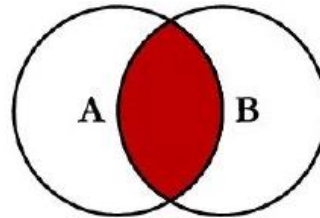
SQL JOINS



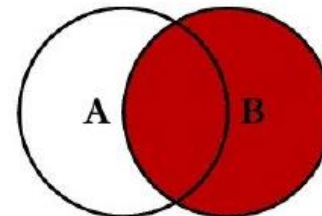
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



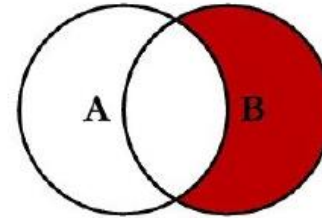
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



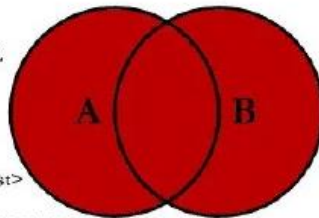
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



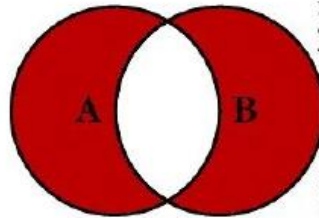
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



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



```
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
```

© C.L. Moffett, 2008

9. SQL

9.6 DCL (Data Control Language)

- ❖ 데이터 제어 언어
- ❖ 데이터베이스에 있는 테이블에 대한 사용 권한 부여나 회수를 위한 명령어
- ❖ 세부 명령어
 - GRANT : 권한부여
 - REVOKE : 권한회수

9. SQL

9.7 TCL (Transaction Control Language)

❖ 트랜잭션 제어 언어

❖ 트랜잭션

- 수행 결과에 완전성을 보장하는 단위
- 원자성, 일관성, 독립성, 지속성

❖ 세부 명령어

- COMMIT : 트랜잭션 작업 결과 반영
- ROLLBACK : 트랜잭션 작업 결과 취소, 조작 명령전으로 복구

10. 하둡 (Hadoop)

10.1 Hadoop

- ❖ Open Source Java Software Framework
- ❖ Google의 GFS (google File System) 논문을 기반으로 출발
- ❖ 2006년 더그 커팅과 마이크 캐퍼렐라가 개발
- ❖ 더그 커팅의 아들이 갖고 놀던 노란 코끼리 이름에서 유래
- ❖ 구성요소
 - 하둡 분산 파일 시스템 (HDFS: Hadoop Distributed File System) → 저장 기능
 - 맵리듀스 (MapReduce) → 연산 기능

10. 하둡 (Hadoop)

10.2 HDFS

❖ Hadoop Distributed File System

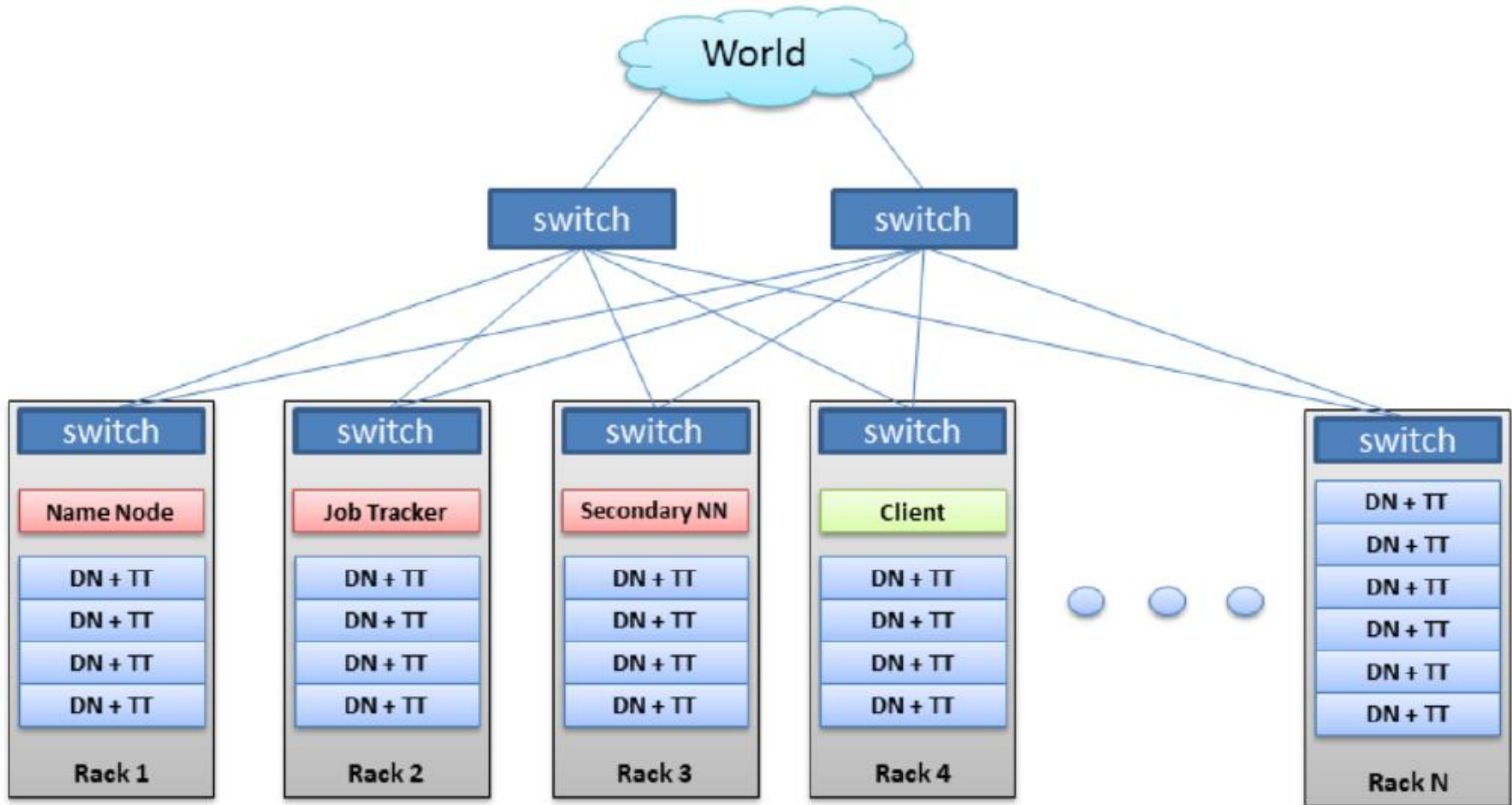
❖ 하둡의 분산형 파일 시스템

❖ 구성요소

- Name node : 파일의 메타(meta) 정보를 관리
- Data node : 실제 데이터를 저장하고 내보내는 역할

10. 하둡 (Hadoop)

10.3 Hadoop Cluster



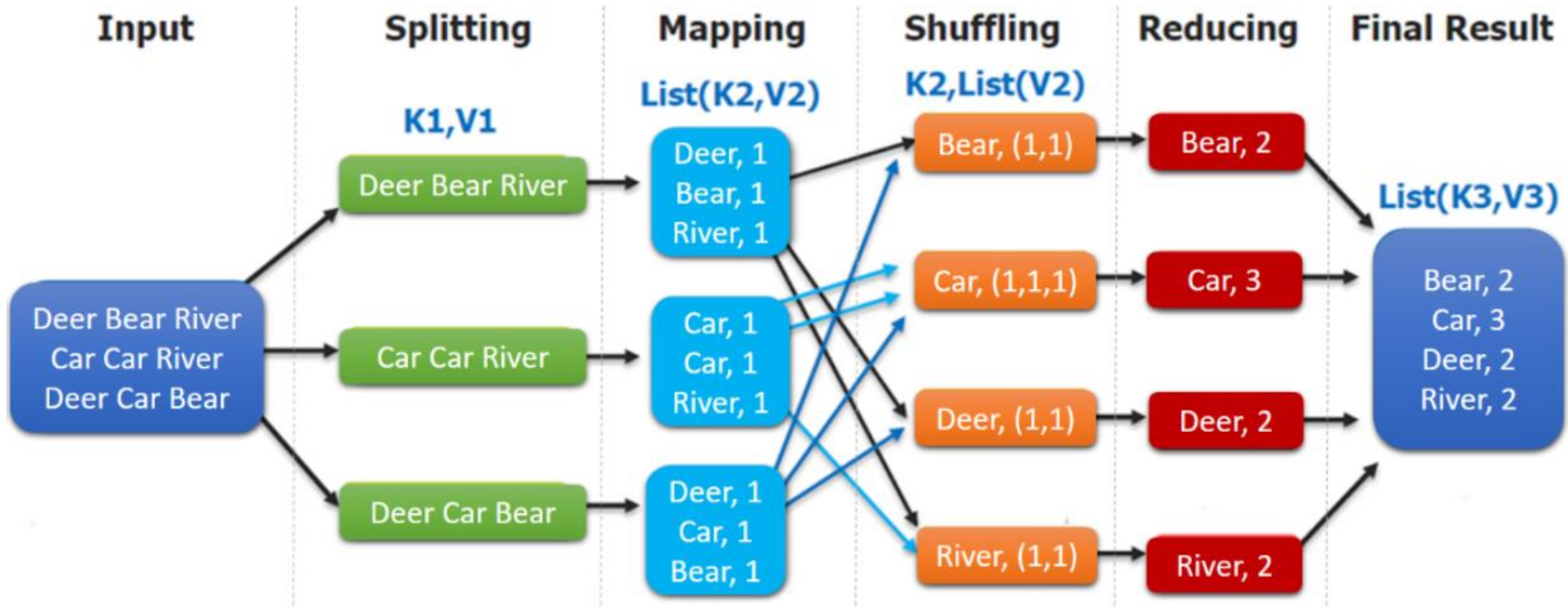
10. 하둡 (Hadoop)

10.4 MapReduce

- ❖ Map, Reduce 라는 두 개의 함수의 조합
- ❖ 분산/병렬 시스템에서 데이터를 처리하는 프레임워크
- ❖ 구성요소
 - Map : 나누어진 단위에서 계산
 - Reduce : Key 별로 합침

10. 하둡 (Hadoop)

10.5 MapReduce Word Count 예제



11. 스파크 (Spark)

11.1 스파크 (Spark)

- ❖ **In-memory기반** Open Source Cluster Computing Framework
- ❖ Hadoop의 MapReduce와 유사한 역할을 하지만 Memory 활용을 극대화 했기 때문에 연산 속도가 월등히 빠름
- ❖ 최근에는 Memory 가격이 많이 저렴해졌고 H/W 성능도 많이 올라가서 MapReduce 보다는 Spark를 선호
- ❖ Scala, Python으로 개발 가능

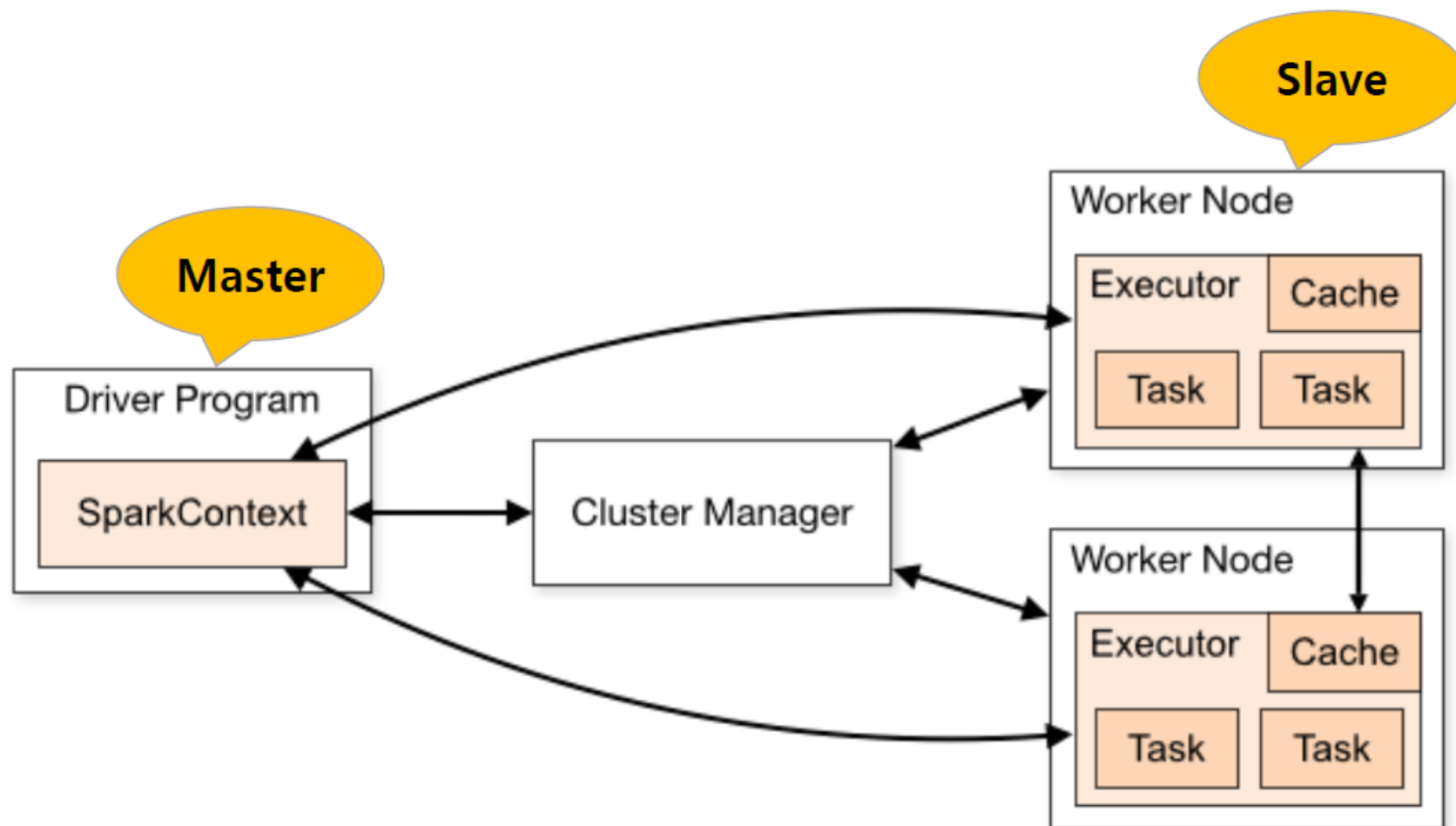
11. 스파크 (Spark)

11.2 스파크의 핵심 개념-RDD

- ❖ Resilient Distributed Datasets
- ❖ 스파크에서 활용되는 내부 데이터 모델
- ❖ 병렬처리가 가능하고 장애가 발생할 경우에도 스스로 복구될 수 있는 내성을 가지고 있음
- ❖ RDD는 한번 생성되면 바뀌지 않으며 다른 형태로 변환이 필요할 경우 새로운 RDD를 만들어 냄
- ❖ 장애시에는 RDD 진행 절차를 기억했다가 그대로 수행하여 빠르게 복구함

11. 스파크 (Spark)

11.3 Spark Architecture



11. 스파크 (Spark)

11.4 기타 Spark의 요소

❖ Spark SQL

- SQL을 사용해 데이터를 처리

❖ Spark Streaming

- 실시간 스트리밍 데이터를 처리하는 프레임워크

❖ Spark Mlib

- 머신 러닝 알고리즘 라이브러리

❖ Spark GraphX

- 그래프 연산용 서브 모듈

11. 스파크 (Spark)

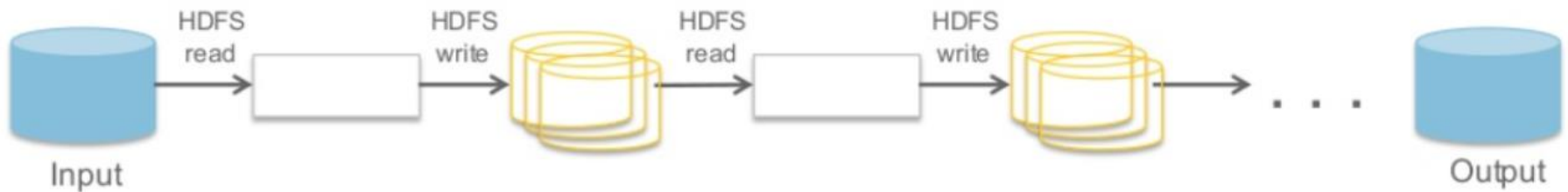
11.5 Hadoop vs Spark

구분	Hadoop	Spark
License	Open Source, Apache	Open Source, Apache
Processing	Model On-Disk, Batch	In-Memory, On-Disk, Batch, Streaming(Near Real-Time)
Language written	Java	Scala

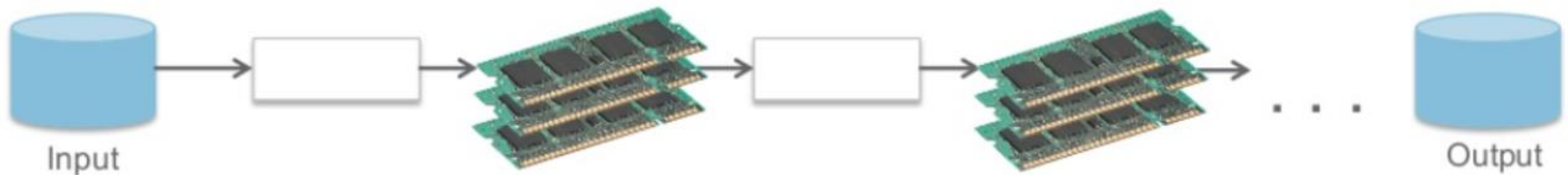
11. 스파크 (Spark)

11.5 Hadoop vs Spark

Hadoop MapReduce: Data Sharing on Disk



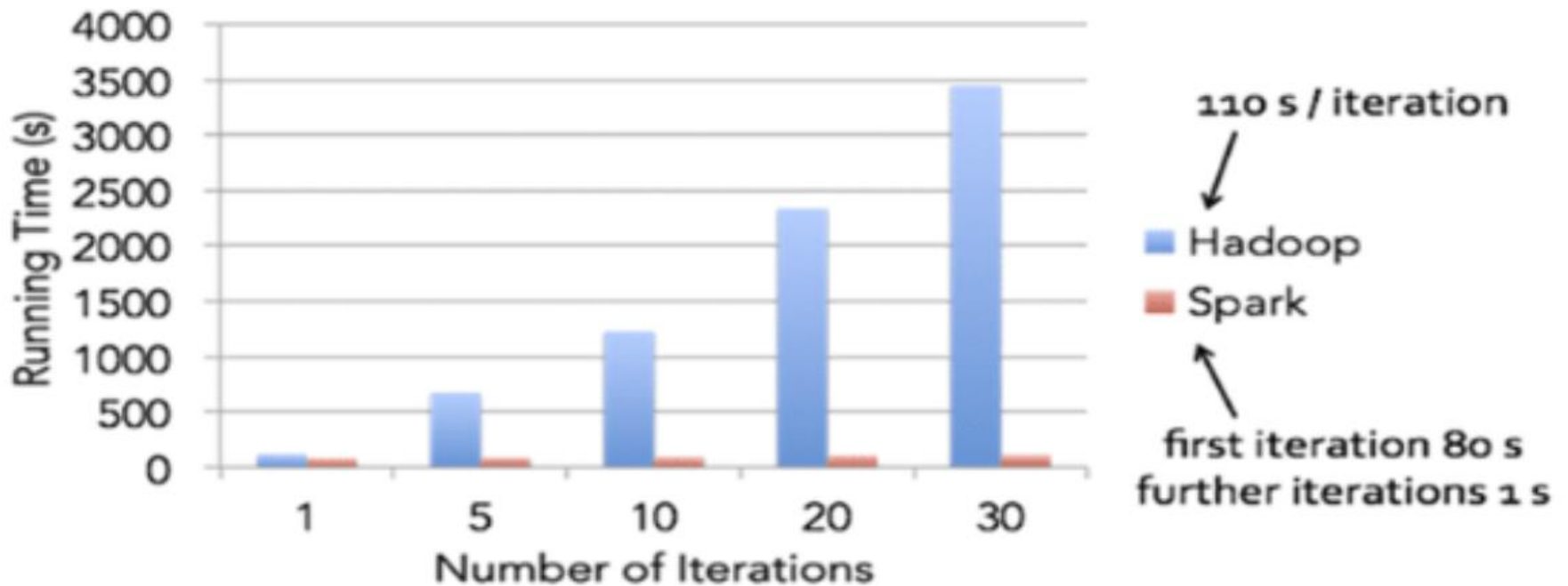
Spark: Speed up processing by using Memory instead of Disks



11. 스파크 (Spark)

11.5 Hadoop vs Spark

10-100X faster Data Management using Apache Spark



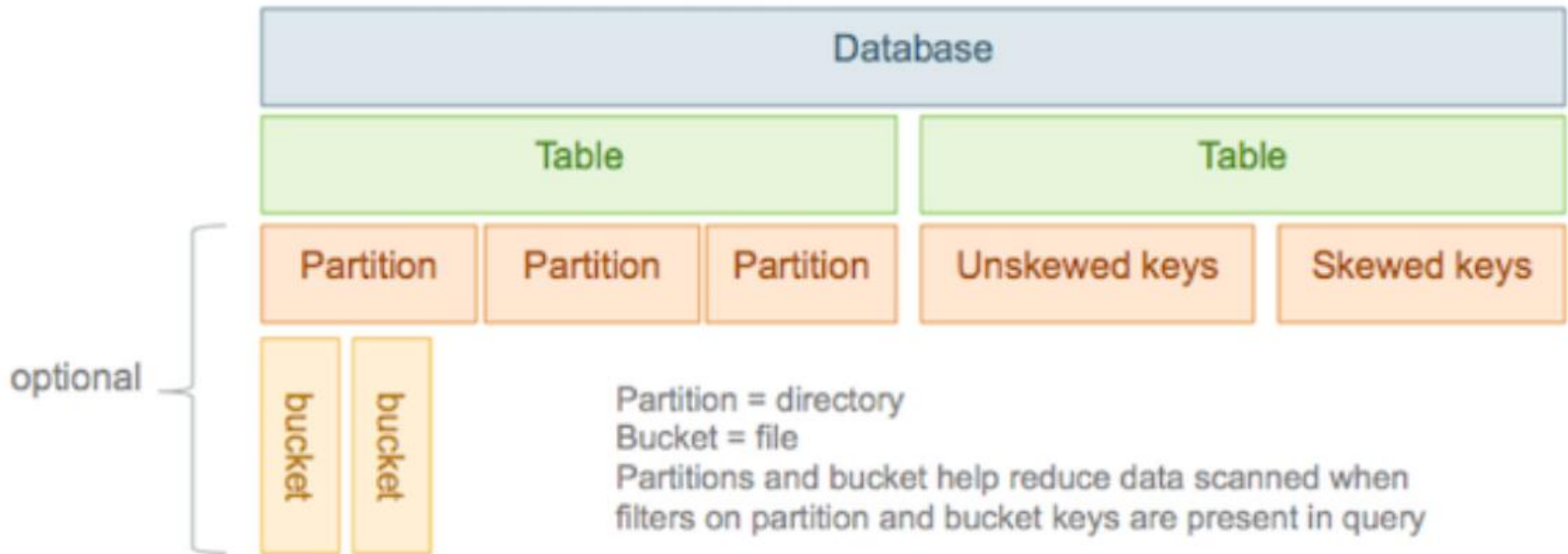
12. 하이브 (Hive)

12.1 하이브 (Hive)

- ❖ 하둡 기반의 데이터 웨어하우스(DW) 소프트웨어
- ❖ SQL기반의 데이터 요약, 질의 및 분석 기능을 제공
- ❖ 특징
 - HDFS & Hbase 등 여러 하둡 에코들과 호환
 - 기본 분석을 위한 QL(SQL-Like)언어 제공
 - 다수의 내부 함수, 분석 함수 제공
 - UDF(User defined function) 지원
 - 다양한 파일 형식으로 전환 가능

12. 하이브 (Hive)

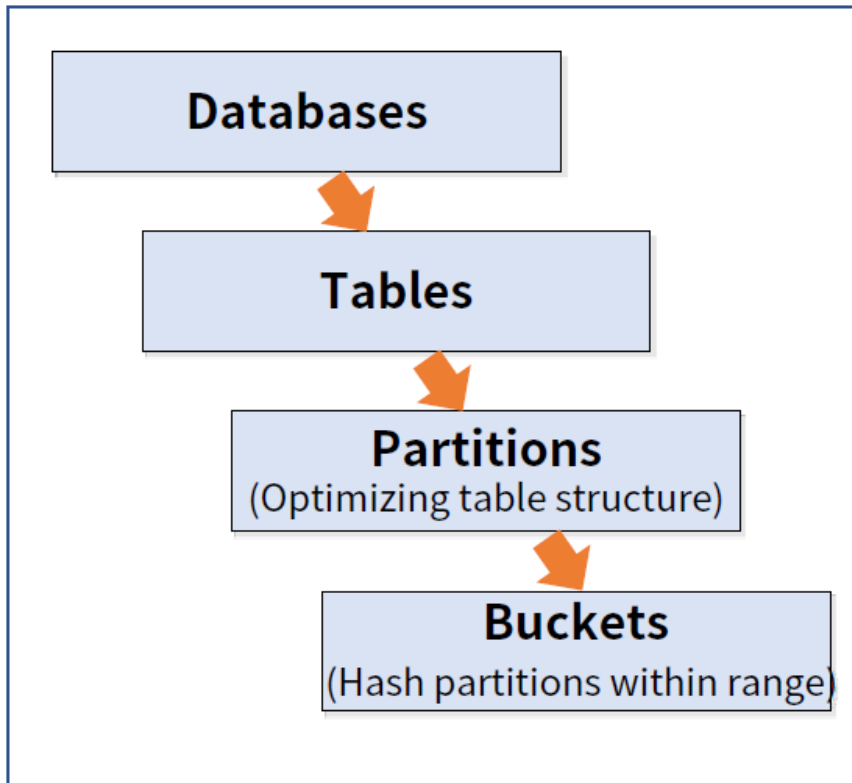
12.2 Hive Data Abstractions



12. 하이브 (Hive)

12.3 Hive Physical Layout

Data Unit



Physical Layout Warehouse directory in HDFS

/user/hive/warehouse : (default directory)

/user/hive/warehouse/**table**

/user/hive/warehouse/table/**partition**

/user/hive/warehouse/table/partition/***.***
(별도의 폴더 없이 파일 단위로 저장)

12. 하이브 (Hive)

12.4 Why Hive?

MapReduce

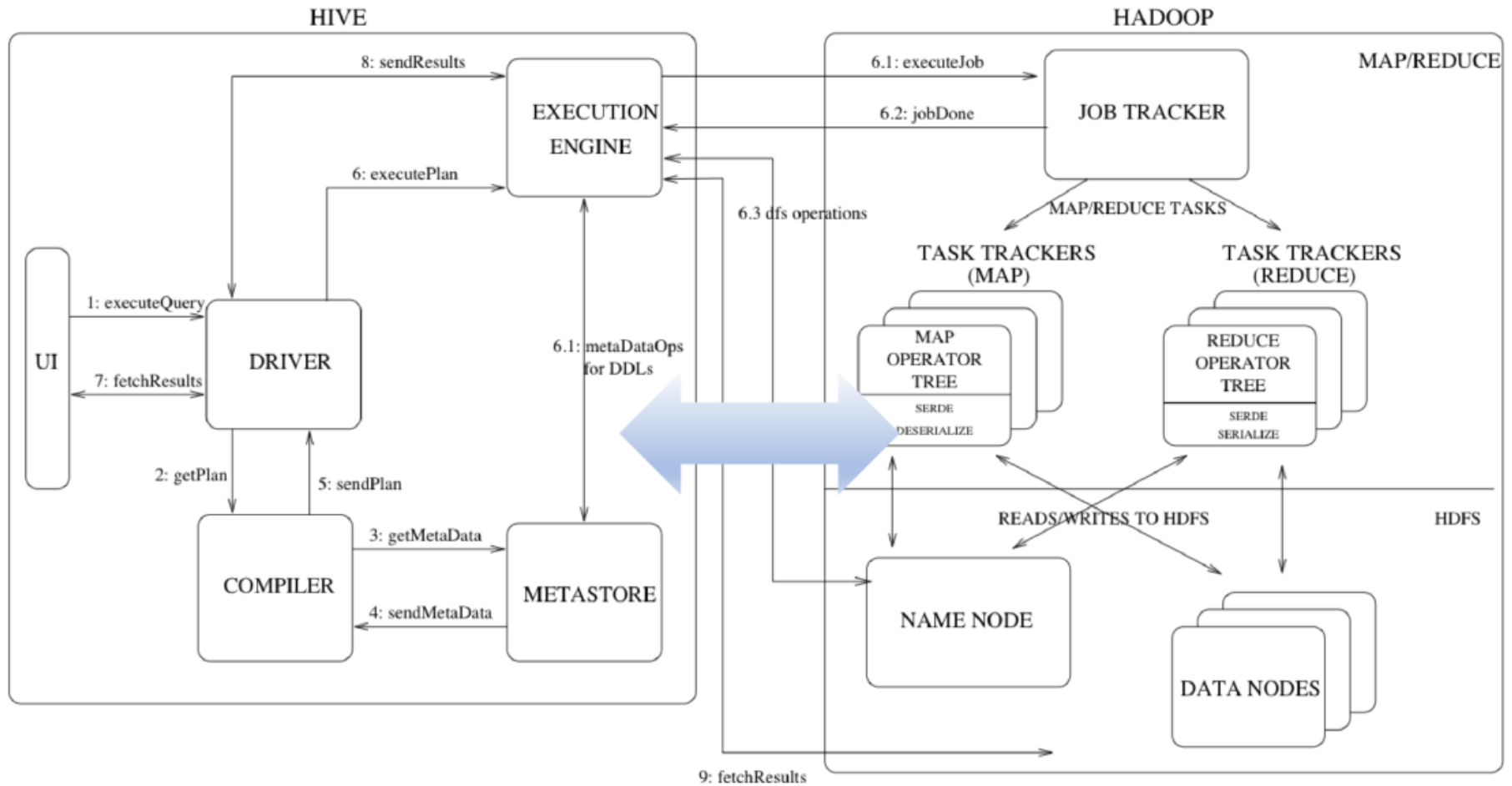
```
public class FruitPromotion {  
    public static class FruitMapper extends Mapper<Text, Text, Text, Text>{  
        public void map(Text key, Text value, Context context) throws  
        IOException {  
            String [] idx = value.toString().split(',');  
            context.write(idx[2], idx[3]);  
        } }  
    public static class FruitReducer extends Reducer<Text, Text, Text,  
    Text>{  
        public void reduce(Text key, Iterable<Text> values, Context context)  
        throws IOException {  
            int total = 0;  
            for (Text val : values) {  
                total = total + Integer.parseInt(val.toString()); }  
            context.write(key, new Text(total));  
        } }  
    public static void main(String[] args) throws Exception {  
        Configuration conf = new Configuration();  
        Job job = new Job(conf, "dictionary");  
        job.setJarByClass(FruitPromotion.class);  
        job.setMapperClass(FruitMapper.class);  
        job.setReducerClass(FruitReducer.class);  
        job.setInputFormatClass(KeyValueTextInputFormat.class);  
        FileInputFormat.addInputPath(job, new Path("/tmp/fruit/"));  
        FileOutputFormat.setOutputPath(job, new Path("output"));  
        System.exit(job.waitForCompletion(true) ? 0 : 1);  
    } }
```

Hive

```
hive> SELECT TB_SALE.F_NAME,  
           sum(TB_SALE.PRICE)  
FROM TB_SALE  
GROUP BY TB_SALE.F_NAME;
```

12. 하이브 (Hive)

12.5 하이브 Architecture



12. 하이버 (Hive)

12.6 SQL on Hadoop

- ❖ HDFS에 저장된 데이터에 대한 SQL 처리를 제공하는 시스템
- ❖ Hive를 포함하여 Impala, Presto, Drill 등 다양한 오픈소스가 있음



Apache Impala



13. NoSQL

13.1 NoSQL DB 배경

❖ 관계형 데이터베이스의 한계점 도달

- ❖ 전통적인 관계형 데이터베이스는 테이블 간 관계를 통해 다양한 데이터 조회 가능
- ❖ **but**, 빅데이터 시대로 오면서 대량의 데이터 적재에 따른 비용이 증가하고 처리 성능에 한계에 도달함



관계형 데이터베이스의 일부 기능을 포기하더라도
확장에 유연하면서 **저장/조회에 높은 성능**이 요구

13. NoSQL

13.2 NoSQL DB 특징

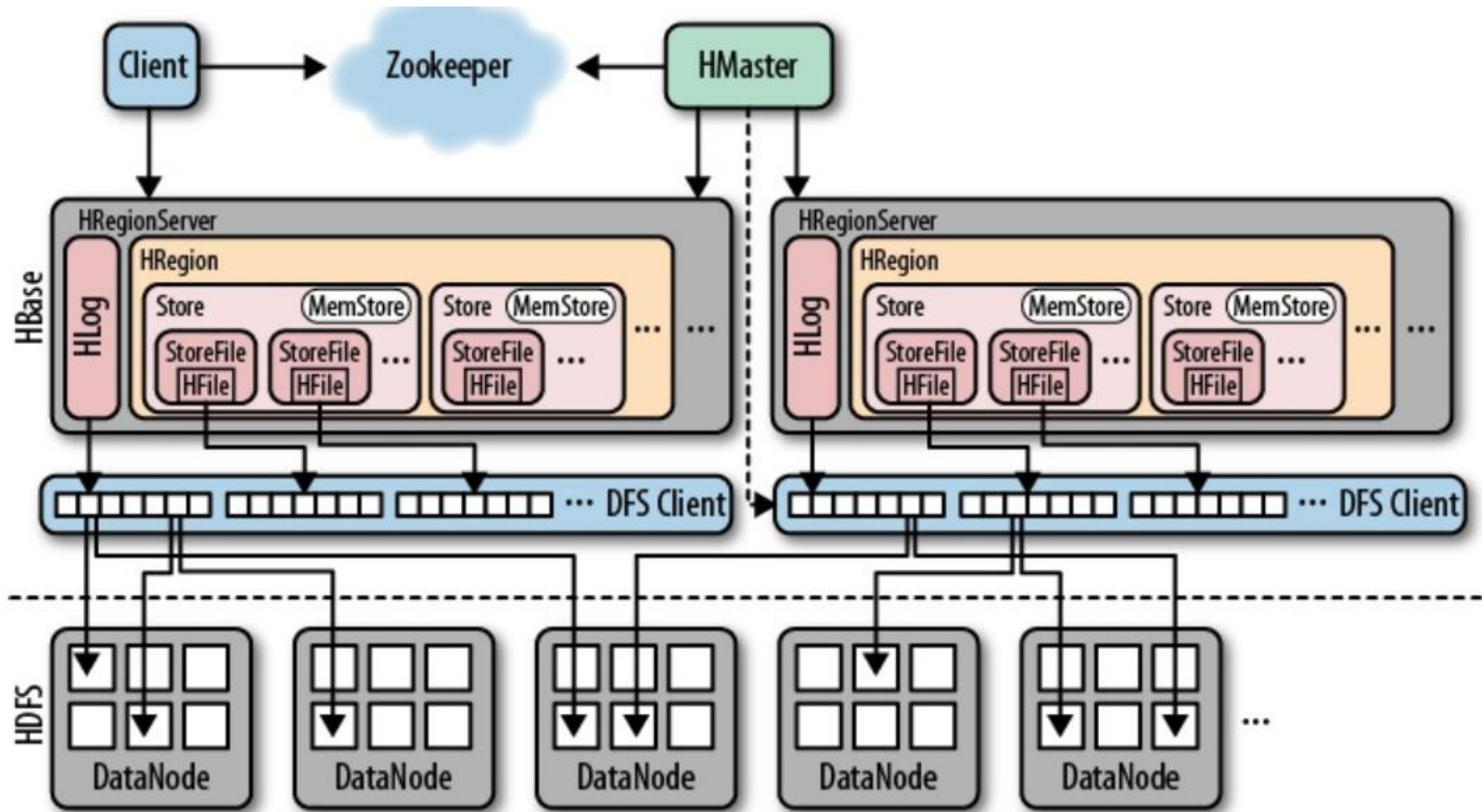
- ❖ Not Only SQL
- ❖ 고정형 Table Schema 불필요
- ❖ 적재와 조회에 성능 최적화
- ❖ 테이블 간 관계형 설정 지양, Join 경우 성능 저하
- ❖ Scale out을 통한 수평적 규모 확장

Not Only SQL

<http://blog.sqlauthority.com>

13. NoSQL

13.3 Hbase 아키텍처



13. NoSQL

13.4 NoSQL DB 종류



13. NoSQL

13.5 NoSQL DB 현황

❖ <http://nosql-database.org/>



Your Ultimate Guide to the
Non-Relational Universe!

[including a historic [Archive](#) 2009-2011]
News Feed covering some changes [here](#) !

NOSQL DEFINITION:Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable.

The original intention has been modern web-scale databases. The movement began early 2009 and is growing rapidly. Often more characteristics apply such as: schema-free, easy replication support, simple API, eventually consistent / BASE (not ACID), a huge amount of data and more. So the misleading term "*nosql*" (the community now translates it mostly with "not only sql") should be seen as an alias to something like the definition above. [based on 7 sources, 15 constructive feedback emails (thanks!) and 1 disliking comment. Agree / Disagree? [Tell](#) me so! By the way: this is a strong definition and it is out there here since 2009!]

LIST OF NOSQL DATABASES [currently >225]

Core NOSQL Systems: [Mostly originated out of a Web 2.0 need]

Wide Column Store / Column Families

NoSQL RELATED EVENTS:

- June 26-27 2018 MongoDB World [»](#)

Register your event 4free: [»](#)

NoSQL ARCHIVE



 **ArangoDB**
the *multi-model* NoSQL DB

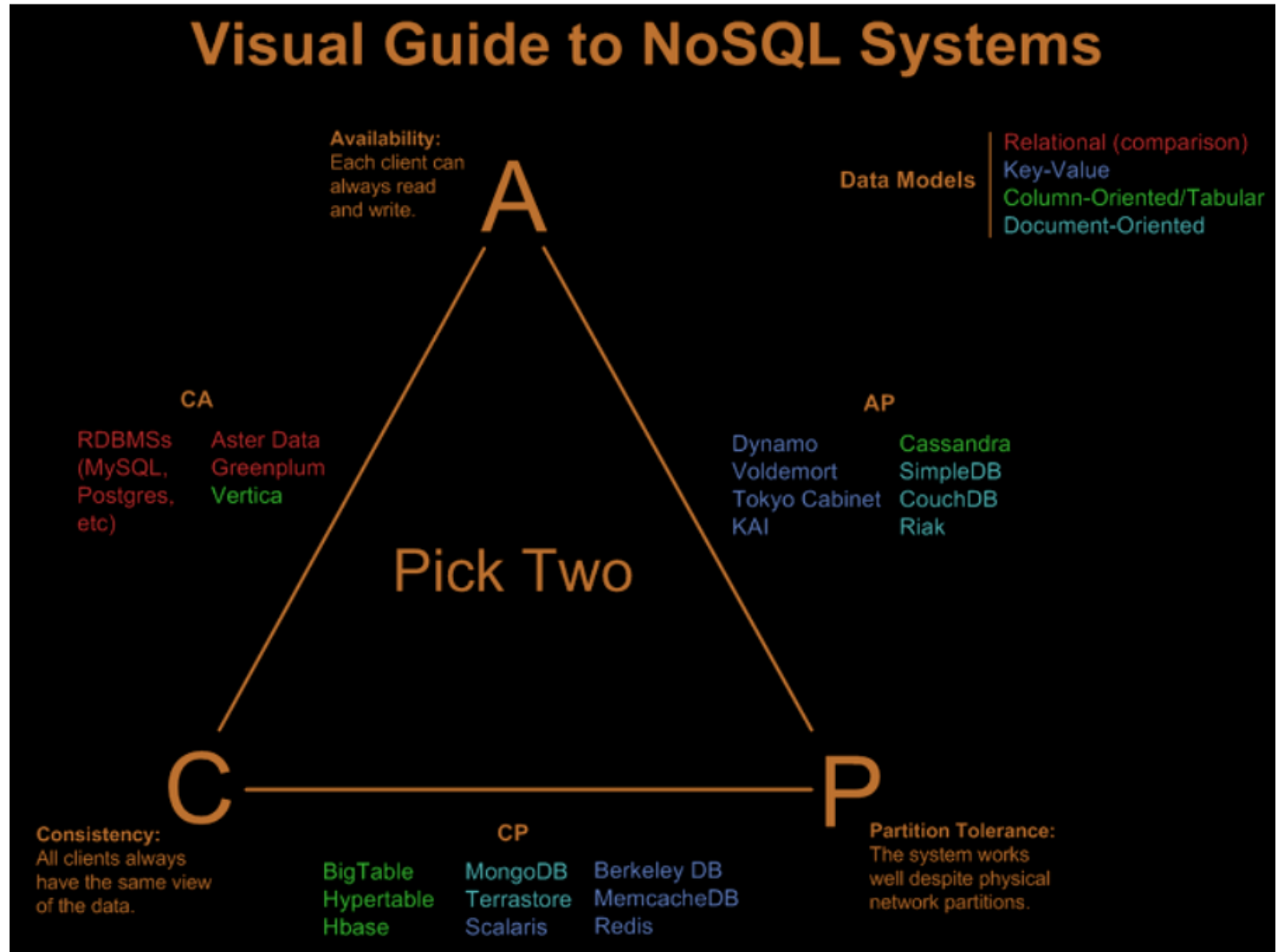
13. NoSQL

13.6 CAP 이론

- ❖ 분산시스템에서 세 가지 속성을 모두 만족할 수 없다는 이론
- ❖ 일관성 (Consistency)
 - 모든 사용자가 같은 순간에 같은 데이터를 볼 수 있다
- ❖ 가용성 (Availability)
 - 모든 클라이언트는 항상 읽기 / 쓰기가 가능하다
- ❖ 분할성 (Partition Tolerance)
 - 시스템이 물리적 네트워크로 분할된 상태에서 정상 동작한다

13. NoSQL

13.6 CAP 이론



13. NoSQL

13.7 ACID, 관계형 데이터베이스의 대표 특성

❖ ACID : 트랜잭션이 안전하게 수행되는 것을 보장하기 위해 관계형 데이터베이스가 가져야 하는 특성

- 원자성 (Atomicity)
 - 한 트랜잭션 안의 모든 작업은 모두 성공하거나 실패해야 함
- 일관성 (Consistency)
 - 한 트랜잭션이 성공하면 모든 데이터는 동시에 일관된 상태로 유지
- 고립성 (Isolation)
 - 한 트랜잭션이 수행되는 동안 다른 트랜잭션이 수행될 수 없음
- 지속성 (Durability)
 - 트랜잭션이 성공하면 해당 최종 상태는 영원히 반영되어야 함

13. NoSQL

13.8 BASE, NoSQL의 특성

❖ Basically Available

- 데이터는 어떠한 경우에도 항상 활용될 수 있는 특성
 - ACID의 고립성(Isolation)과 대립되는 특성임

❖ Soft-State

- 분산 노드 간 업데이트는 데이터가 노드에 도달한 시점에 갱신

❖ Eventual Consistency

- 일관성을 제공하되 모든 노드에서 동시에 보장되지는 않음
- 시간이 지나면 일관성이 보장이 되는 '지연된 일관성' 보장

13. NoSQL

13.9 RDB vs NoSQL DB 테이블 조회 실체

```
mysql> select * from student;
```

roll_no	name	specialization	dob
11	Subhransu Patra	cse	1983-06-03
12	Sudhansu Patra	etc	0000-00-00
13	Suvransu	ee	0000-00-00
14	Jonny	etc	1982-06-02
15	Missy	ee	1981-05-04
16	Jenny	cse	1982-05-07
18	Kyle	cse	1983-07-06
19	Nathan	ee	1982-02-05
20	Abby	cse	1984-09-08

9 rows in set (0.00 sec)

```
hbase(main):012:0> scan 'Contacts'
```

ROW	COLUMN+CELL
1000	column=Office:Address, timestamp=1439233899349, value=1111 San Gabriel Dr.
1000	column=Office:Phone, timestamp=1439233891147, value=1-425- 000-0002
1000	column=Personal:Name, timestamp=1439233796882, value=John Dole
1000	column=Personal:Phone, timestamp=1439233884195, value=1-42 5-000-0001

1 row(s) in 0.0490 seconds

13. NoSQL

13.10 NoSQL DB 언제 사용? 주의점은?

❖ 언제 활용?

- 실시간 대용량 데이터 적재 필요시
- 테이블 간 Join 등의 처리가 거의 없는 경우
- 추후 DB 확장이 유연해야 하는 경우

❖ 주의점

- NoSQL 특성에 맞는 Table 설계가 매우 중요 → RDB와 전혀 다름
- 테이블간 Join을 할 경우 성능 저하 심함
- 유지보수에 따른 운영 인력 확인 필수



용도가 명확할 경우 강력한 NoSQL DB !!!