2장 이 책에서 다루는 도구와 데이터

3강 시스템

▼ 1. PostgreSQL

- 오픈소스 RDB (Relational Database)
- 다양한 플랫폼을 지원하며 GUI 인스톨러가 제공되어 쉽게 컴퓨터에 설치 가능
- 표준 SQL을 잘 준수하며 윈도함수, CTE (WITH 구문) 등 분석에 필수적으로 사용하는 구문을 모두 구현함.

▼ 2. Apache Hive

- 디스크 I/O는 속도가 매우 느림 → 보틀넥 발생 → 고속으로 데이터를 처리하기 위한 아키텍처로 **분산 파일 시스템(HDFS)**이 고안됨.
- 분산 파일 시스템을 통해 **거대한 데이터를 작게 분할해서 여러 개의 디스크에 분산해서 저장**하고 **각 디스크에서 동시에 데이터를 읽어 들여** 고속으로 대량의 데이터 처리를 함.
- `Apache Hive`는 분산 파일 시스템 위에 데이터를 SQL스러운 인터페이스로 간단하게 처리해주는 시스템
- **분산 파일 시스템 위의 데이터의 순서를 맞추는 방법**으로 `MapReduce`라는 알고리즘이 고안됨.
- HDFS와 MapReduce 아키텍처를 구현한 시스템이 초기의 `Apache Hadoop`이고, Hive는 Hadoop 생태계의 일부분

▼ 장점

- **대량의 데이터를 한번에 처리**
- HiveQL로 작성한 쿼리를 자동적으로 MapReduce 잡으로 변환해서 간단하게 병렬 분산 처리를 할 수 있음.
- 쿼리 실행 때 동적으로 데이터를 정의할 수 있음.
- 데이터 분석을 위한 풍부한 UDF(User-Defined Function)을 활용해서 SQL만으로는 구현하기 어려운 문자열 처리 등을 간단하게 할 수 있음.

▼ 단점

- 파일 기반 시스템이므로 특정 레코드 하나를 변경하거나 제거하는 것이 어렵고 인덱스도 디폴트로 존재하지 않아 쿼리 실행 때 **파일 전체를 조작해야 함**.
- Hive는 처리율(throughput)을 높이기 위한 아키텍처를 가지고 있어 **리액턴시가 낮은 처리(실시간 처리)를 요구하는 경우에는 적합X**
  - `리액턴시` : 특정 작업을 요청한 후 그 작업이 완료되기까지 걸리는 시간

    → 시스템의 응답 속도를 나타내는 지표
- 쿼리를 실행할 때 HiveQL로 작성된 처리를 자바 코드로 변환하고 생성된 jar를 각각의 연산 노드에 배치하고 처리를 시작하는 복잡한 과정을 거치므로 간단한 쿼리라도 결과를 얻는 데까지 많은 시간(수 초~수 시간)이 걸리는 경우가 꽤 있음.

- 최근에는 실행 엔진으로 MapReduce 대신 Spark 또는 Tez를 사용하거나 데이터 형식을 최적화해서 리액턴시를 낮추는 방법도 쓰임.

▼ 3. Amazon Redshift

- Amazon Web Service(AWS)에서 제공하는 **분산 병렬 RDB**
- Hive는 파일 기반의 배치 처리를 SQL스러운 인터페이스로 구현할 수 있는 시스템이지만, Redshift는 RDB
- 레코드를 업데이트하거나 제거할 수도 있으며 트랜잭션 등도 지원함.

  *`Transaction` : 데이터베이스의 상태를 변화시키기 해서 수행하는 작업의 단위
- Redshift의 접속 인터페이스는 PostgreSQL과 호환성을 가지므로 PostgreSQL 전용 ODBC/JDBC 드라이버 또는 psql 클라이언트에서 곧바로 Redshift에 접속할 수도 있음.
- **컬럼 기반 스토리지**를 사용함. → 테이블 설계, 쿼리 실행 때 일반적인 RDB와는 다른 발상이 필요함.

  *`컬럼 기반 스토리지` : 테이블의 데이터를 물리적으로 저장할 때 레코드별로 저장하는 것이 아니라 컬럼별로 저장. 데이터의 압축률을 높일 수 있고, 쿼리 실행 때 디스크 I/O를 줄일 수 있음.
- 분석에 필요한 데이터를 모두 하나의 테이블 컬럼에 추가하는 형태로 진행함. 쿼리 실행 때도 'SELECT *'처럼 모든 컬럼을 추출하는 쿼리는 성능이 낮게 나오므로 필요한 컬럼만 추출하는 쿼리를 실행해야 함.

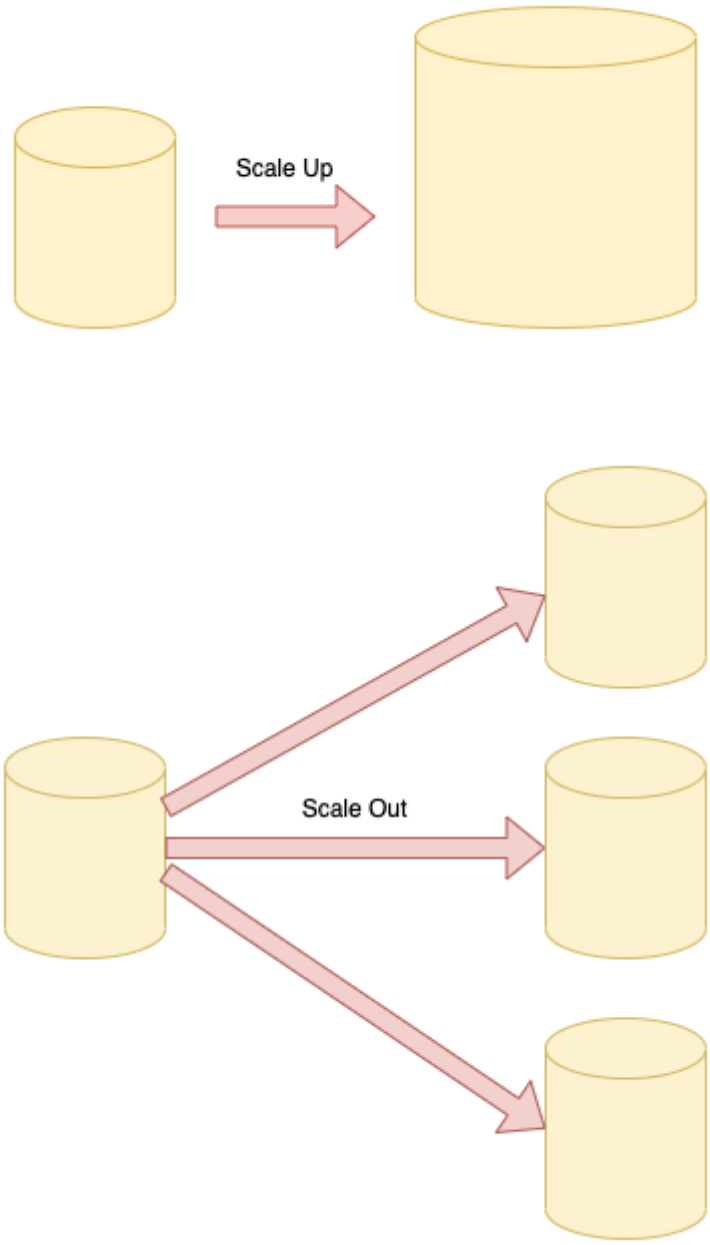
▼ 장점

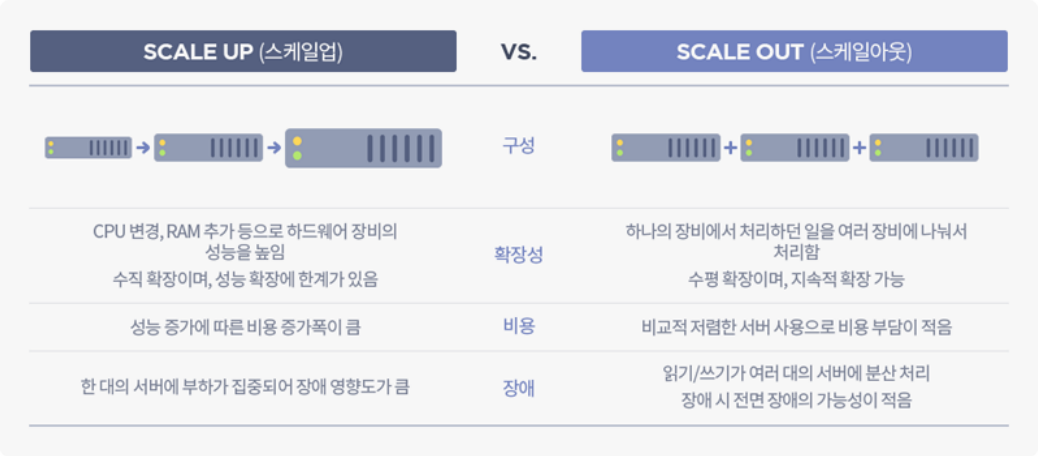
- 일반적인 RDB에서 다룰 수 없는 대량의 데이터와 상호 작용하는 쿼리를 실행하고 싶을 때 효과적임. → 매우 빠름.
- AWS에서 제공하는 클라우드 서비스이므로 **scale-up/scale-out이 쉬움**.

▼ Scale-up vs. Scale-out

`Scale-up` : 기존 서버의 사양을 업그레이드해 시스템을 확장하는 것

`Scale-out` : 서버를 여러 대 추가하여 시스템을 확장하는 것





▼ 단점

- PostgreSQL, Hive와 달리, 사용 시간에 따라 비용 발생
- 성능 튜닝을 하거나 비용을 줄이려면 최적의 노드 수와 스펙을 예측해서 인스턴스의 실행과 종료를 관리해야 함.

▼ 4. Google BigQuery

- 빅데이터 분석을 위해 구글이 제공하는 **클라우드 서비스**
- 레거시 SQL(default)과 스탠다드 SQL이 있음.
- **컬럼 지향 스토리지 아키텍처**

▼ 장점

- **노드 인스턴스를 직접 관리할 필요가 없음**.
- **읽어들인 데이터의 양**으로 비용이 발생함.

  → 다루는 데이터가 적으면 적은 비용으로 운용 가능

  → 쿼리 실행 때 데이터 로드를 줄이도록 자주 읽어 들이는 데이터만 모아서 별도의 테이블로 분할하거나 필요한 컬럼만 선택하는 SELECT 구문을 활용하는 등의 테크닉이 필요함.
- 구글이 제공하는 GA(Google Analytics), 다른 클라우드 서비스와 쉽게 연동이 가능

▼ 단점

- 데이터양을 기반으로 비용이 발생하므로 사용 요금을 예측하기 어려움.

▼ 5. SparkSQL

- Apache Spark의 기능 중에서 SQL 인터페이스와 관련된 기능

  `Spark` : 빅데이터 처리를 위한 오픈 소스 고속 분산처리 엔진

▼ 장점

- 인터페이스로 SQL, 파이썬, 스칼라, 자바, R 등의 **다양한 프로그래밍 언어를 지원**하여 **빅데이터 활용과 관련된 대부분의 처리를 한 번에 구현할 수 있음.**
- DataFrames API를 이용하여 SQL스러운 선언적인 구문으로 데이터를 조작할 수 있고, 절차적인 프로그래밍과 같은 방법으로 프로그램을 구현할 수 있음.

4강 데이터

▼ 1. 데이터의 종류

업무 데이터

- **서비스와 시스템을 운용하기 위한** 목적으로 구축된 데이터베이스에 존재하는 데이터
- 업무 데이터의 대부분은 **갱신형 데이터**
  - 상품을 추가할 때는 레코드 하나 삽입
  - 가격 변경이 있으면 기존의 데이터 갱신
- **트랜잭션 데이터**
  - **서비스와 시스템을 통해 사용자의 행동을 기록한 데이터**
  - ex) 구매 데이터, 리뷰 데이터, 게임 플레이 데이터

    - 날짜, 시각, 마스터 데이터의 회원 ID, 상품 ID, 수량, 가격 등이 포함되는 경우가 많음.
    - 트랜잭션 데이터를 기반으로 리포트를 만드는 경우가 많음.

- **마스터 데이터**
    - **서비스와 시스템이 정의하고 있는 데이터**
    - ex) 카테고리 마스터, 상품 마스터
    - 트랜잭션 데이터의 상품 ID와 마스터 데이터를 결합해서 상품 이름, 상품 카테고리, 발매일 등을 명확하게 만들어야 함.

로그 데이터

- **통계 또는 분석**을 주 용도로 설계된 데이터
- 특정 태그를 포함해서 전송된 데이터
- 특정 행동을 서버 측에 출력한 데이터
- **누적형 데이터**
    - 출력 시점의 정보를 축적해놓음.
    - 로그 출력 이후에 가격이 변경되거나 사용자 정보가 변경되더라도 기존의 데이터를 수정X

▼ 2. 업무 데이터

업무 데이터의 특징

- **데이터의 정밀도가 높다**
    - 업무 데이터는 여러 데이터 처리를 하는 중에 문제가 발생하면, 트랜잭션과 콜백이라는 기능을 사용해 문제를 제거함.
- **갱신형 데이터**
    - 업무 데이터는 매일 다양한 데이터 추가, 갱신, 제거 등이 실행됨.
    - 데이터 갱신이 일어나는 경우
        - 사용자가 탈퇴하는 경우, 데이터를 물리적으로 제거

        - 주문을 취소하는 경우, 플래그를 통해 상태를 변경해서 논리적으로 제거
        - 이사 등으로 주소가 변경된 경우, 사용자 정보를 갱신
- **다뤄야 하는 테이블의 수가 많다**
    - 대부분의 서비스는 데이터베이스로 **RDB** 이용
    - 데이터의 확장성을 배제하고 데이터의 **정합성을 쉽게 유지**하며 데이터를 저장하기 위함 (정규화)
    - ER 다이어그램 (데이터 구조를 나타낸 설계 문서)를 파악하고 여러 테이블을 결합해야 데이터 전체 내용을 파악할 수 있음.

업무 데이터 축적 방법

- 모든 데이터 변경하기
    - 날짜를 기반으로 데이터가 계속 누적되는 경우가 아니라면, 데이터 전체를 한꺼번에 바꾸어 최신 상태로 만듦. (ex. 우편번호 마스터, 상품 카테고리 마스터)
- 모든 레코드의 스냅샷을 날짜별로 저장하기
- 어제와의 변경 사항만 누적하기
    - 데이터 전송량과 처리 시간을 줄일 수 있음.

업무 데이터 다루기

- 매출액, 사용자 수처럼 **정확한 값을 요구할 경우** 활용하기
    - 업무 데이터는 트랜잭션 기능으로 인해 데이터의 정합성이 보장되므로 추출 결과를 신뢰할 수 있음.
    - 로그 데이터는 전송 방법에 따라서 중간 손실이 발생할 수 있으므로, 정확한 값을 요구할 때는 업무 데이터를 활용함.
- **서비스의 방문 횟수, 페이지뷰, 사용자 유도** 등의 데이터 분석에는 사용할 수 없음
    - 로그 데이터를 사용해야 함.
- 데이터 변경이 발생할 수 있으므로 추**출 시점에 따라 결과가 변화**할 수 있음
    - 리포트를 만들 때 추출 시점의 정보를 기반으로 작성된 리포트라고 명시 필요

▼ 3. 로그 데이터

로그 데이터의 특징

- **시간, 사용자 엔드 포인트, IP, URL, 레퍼러, Cookie** 등의 정보 저장하기
- 로그 데이터는 추출 방법에 따라 데이터의 정밀도가 달라짐
- 계속 기록을 추가하는 것 뿐이므로 과거의 데이터가 변경되지는 않음

로그 데이터 축적 방법

- **태그, SDK를 통해 사용자 장치에서 데이터를 전송하고 출력하기 (비컨 형태)**
  - GA처럼 HTML에 특정 태그를 집어넣고 데이터를 전송하는 형식
  - 웹사이트에서 자바스크립트를 통해 로그 데이터를 전송하는 경우, 자바스크립트를 해석할 수 없는 **크롤러 또는 브라우저의 데이터는 로그로 출력되지 않음**.
- **서버에서 데이터를 추출하고 출력하기 (서버 형태)**
  - 클라이언트 쪽에서 별도의 처리를 하지 않고 서버에서 로그를 출력하는 방법
  - 서버에 요청이 있을 때 출력하므로, 따로 크롤러의 접근을 확인하고 조건을 걸지 않는 이상 **크롤러의 접근도 출력**됨.
  - 의도하지 않은 로그는 반드시 제거하고 사용자의 행동을 집계/분석해야 함.

로그 데이터 다루기

- **사이트 방문 횟수, 페이지뷰, 사용자 유도 상황**을 집계하고 분석할 때 주로 사용
  - 업무 데이터로 관리할 수 없는 열람 페이지, 레퍼러, 사용자 에이전트 등을 저장할 수 있음.
- 최신 상태를 고려한 분석에는 적합하지 않음
- 계속 기록을 누적하는 형태이므로 추출 결과가 변할 가능성이 적음
- **데이터의 정확도는** 업무 데이터에 비해 **낮음**
  - 로그 추출 방법에 따라 사용자가 누락될 수 있음.
  - 크롤러의 로그가 함께 포함되어 집계될 수 있음.

▼ 4. 두 데이터를 사용해서 생성되는 가치

업무 데이터와 로그 데이터의 가치

- **업무 데이터**
  - 매출액의 추이, 인기 있는 상품 등을 파악하여 사용자의 구매를 유도할 수 있음.
  - 어떤 상품이 계절성을 가지는지, 특정 시간에 많이 팔리는지 등 과거의 경향을 파악하여 계획을 세울 수 있음.
  - 다양한 이벤트를 통해 상품을 더 많이 노출시켜 더 많은 구매를 유도할 수 있음.
- **로그 데이터**
  - 대부분의 접근 분석 도구는 웹사이트에 비컨 형식으로 탑재되어 로그를 전송하며 이 로그를 기반으로 리포트를 제공해줌.
  - 페이지뷰, 액션, 해당 데이터에 포함된 값(레퍼러, 사용자 에이전트, 사용자 정의 변수 등)을 집계하고 출력해줌.

두 데이터를 사용했을 때 발생하는 새로운 가치

- 업무 데이터는 **오프라인**에서의 데이터도 사용할 수 있음.
- 로그 데이터는 주로 **웹사이트**에서의 행동만을 기록함.
- 두 데이터를 활용해 웹사이트에서의 행동이 오프라인의 행동에 어떠한 영향을 미치는지 등을 조사할 수 있음.

데이터 사용 가치 (활용 사례)

- 목표를 관리하고, 설계하고, 서비스/조직의 성장에 기여하기 (**목표 관리**)
- 사용자 행동을 기반으로 경향을 발견하고, 매출과 서비스 개선에 기여하기 (**서비스 개선**)
- 과거의 경향을 기반으로 미래의 행동 예측하기 (**미래 예측**)

▼ 참고문헌

데이터 분석을 위한 SQL 레시피 2장

Scale-up과 Scale-out에 대해 알아보자! (techcourse.co.kr)