Data Science

데이터 분석

데이터 탐색

강사 양석환

데이터의 적재와 저장

데이터 적재

- 데이터 적재(Load) 도구
  - 수집한 데이터는 빅데이터 분석을 위한 저장 시스템에 적재해야 함
  - 관계형 데이터베이스, HDFS를 비롯한 분산 파일 시스템, NoSQL 저장 시스템에 데이터를 적재할 수 있음
  - 데이터 적재 방법
    - 데이터 수집 도구를 이용한 데이터 적재
      - 대표적인 데이터 적재 도구: 플루언티드, 플럼, 스크라이브, 로그스태시 등
    - NoSQL DBMS가 제공하는 도구를 이용한 데이터 적재
      - 수집한 데이터가 CSV 등의 텍스트 데이터라면 mongo import와 같은 적재 도구를 사용하여 수행할 수 있음
    - 관계형 DBMS의 데이터를 NoSQL DBMS에
      - 기존에 운영 중이던 관계형 데이터베이스로부터 데이터를 추출하여 NoSQL 데이터베이스에 적재할 수 있음
      - 데이터 변형 정도에 따라 제공되는 도구를 사용하거나 직접 프로그램을 작성하여 적재할 수 있음

데이터 적재

- **데이터 적재(Load) 도구**
  - **데이터 적재 완료 테스트**
    - 데이터 적재 내용에 따라 체크리스트를 작성
      - 정형 데이터: 테이블 개수, 속성 개수 및 데이터 타입의 일치 여부, 레코드 수 일치 여부가 체크리스트에 포함될 수 있음
      - 반정형 또는 비정형 데이터: 원천 데이터의 테이블이 목적지 저장 시스템에 맞게 생성되었는지, 레코드 수가 일치하는지 등이 체크리스트에 포함될 수 있음
    - 데이터 테스트 케이스 개발
      - 원천 데이터에서 특정 데이터에 대하여 샘플링 한 후, 목적지 저장 시스템에서 조회하는 테스트 케이스를 개발할 수 있음
      - 적재한 대량 데이터의 데이터 타입(특히 한글 문자 등의 ASCII 코드가 아닌 문자들)이 정상적으로 적재되는지 확인 필요
      - 문자열이 숫자로 이루어지면 문자열로 적재되었는지, 또는 숫자인데 문자열로 적재되지는 않았는지 확인 필요
    - 체크리스트 검증 및 데이터 테스트 케이스 실행

데이터 저장

- 데이터 저장 시스템
  - 빅데이터 저장 시스템
    - 대용량 데이터 집합을 저장하고 관리하는 시스템. 사용자에게 데이터 제공 신뢰성과 가용성을 보장함
    - 파일 시스템 저장 방식
      - 빅데이터를 확장 가능한 분산 파일의 형태로 저장하는 방식의 대표적인 예. Apache HDFS, 구글의 GFS 등
      - 파일 시스템 저장 방식은 저사양 서버들을 활용하여 대용량, 분산, 데이터 집중형의 애플리케이션을 지원하며 사용자들에게 고성능 Fault-Tolerance 환경을 제공하도록 구현되어 있음
    - 데이터베이스 저장 방식
      - 빅데이터를 저장하는 방식: 전통적인 관계형 데이터베이스 시스템을 이용, NoSQL 데이터베이스 시스템을 이용
      - NoSQL 데이터베이스: 대용량 데이터의 저장 측면에서 봤을 때, 관계형 데이터베이스보다 수평적 확장성, 데이터 복제, 간편한 API 제공, 일관성 보장 등의 장점이 있음

데이터 저장

- **데이터 저장 시스템**
  - **분산 파일 시스템**
    - **하둡 분산 파일 시스템(Hadoop Distributed File System, HDFS)**
      - **마스터(Master) 하나와 여러 개의 슬레이브(Slave)로 클러스터링 되어 구성됨**
        - 마스터 노드(Master Node)=네임 노드(Name Node): 슬레이브를 관리하는 메타데이터와 모니터링 시스템을 운영함
        - 슬레이브 노드(Slave Node)=데이터 노드(Data Node): 데이터 블록을 분산 처리함
      - **데이터 손상을 방지하기 위해서 데이터 복제 기법을 사용함**

데이터 저장

- **데이터 저장 시스템**
  - **분산 파일 시스템**
    - **구글 파일 시스템(Google File System, GFS)**
      - **엄청나게 많은 데이터를 보유해야 하는 구글의 핵심 데이터 스토리지와 구글 검색 엔진을 위해 최적화된 분산 파일 시스템**
      - **마스터(Master), 청크 서버(Chunk Server), 클라이언트로 구성됨**
        - 마스터: GFS 전체의 상태를 관리하고 통제함
        - 청크 서버: 물리적인 하드디스크의 실제 입출력을 처리함
        - 클라이언트: 파일을 읽고 쓰는 동작을 요청하는 애플리케이션
      - **파일들은 일반적인 파일 시스템에서의 클러스터, 섹터 들과 비슷하게 64MB로 고정된 크기의 청크로 나누어서 저장됨**
      - **가격이 저렴한 서버에서의 사용, 하드웨어 안정성이나 자료들의 유실을 고려하여 설계되었고, 응답시간이 조금 길더라도 데이터의 높은 처리 성능에 중점을 두고 개발됨**

데이터 저장

- **데이터 저장 시스템**
  - **NoSQL**
    - 전통적인 관계형 데이터베이스보다 유연한 데이터의 저장 및 검색을 위한 매커니즘을 제공함
    - 대규모 데이터를 처리하기 위한 확장성, 가용성 및 높은 성능 제공. 빅데이터 처리와 저장을 위한 플랫폼으로 활용
    - SQL 계열 쿼리를 지원하는 데이터베이스도 있기 때문에 “Not Only SQL”로 불리기도 함

RDBMS와 NoSQL의 장단점 및 특성 비교

구분	장단점		특성
RDBMS	장점	• 데이터 무결성과 정확성 보장 • 정규화된 테이블과 소규모 트랜잭션 지원	• UPDATE, DELETE, JOIN 연산이 가능함 • ACID 트랜잭션 지원 • 고정 스키마 보유
	단점	• 확장성에 한계가 있음 • 클라우드 분산 환경에 부적합	
NoSQL	장점	• 웹 환경의 다양한 정보를 검색, 저장할 수 있음	• 수정, 삭제를 사용하지 않음(입력으로 대체) • 강한 일관성이 불필요함
	단점	• 데이터의 무결성과 정확성을 보장하지 않음	

데이터 저장

- **데이터 저장 시스템**
  - **NoSQL**
    - **기술적 특성**
      - **NoSQL은 고전적인 관계형 데이터베이스의 주요 특성을 보장하는 ACID 특성 중 일부만을 지원하는 대신**
      - **성능과 확장성을 높이는 특성을 강조함**

ACID
Atomicity (원자성), Consistency (일관성),
Isolation (고립성), Durability (지속성,내구성)

특성	내용
無 스키마	• 데이터를 모델링하는 고정된 데이터 스키마 없이 Key-Value를 이용하여 다양한 형태의 데이터 저장 및 접근이 가능함 • 데이터 저장 방식은 크게 열, 값, 문서, 그래프 등의 네 가지를 기반으로 구분함
탄력성 (Elasticity)	• 시스템 일부에 장애가 발생해도 클라이언트가 시스템에 접근 가능 • 응용 시스템의 다운 타임이 없도록 하는 동시에 대용량 데이터의 생성 및 갱신 가능 • 질의에 대응할 수 있도록 시스템 규모와 성능 확장이 용이하며, 입출력의 부하를 분산시키는데 용이한 구조임
질의 (Query) 기능	• 수십 대에서 수천 대 규모로 구성된 시스템에서도 데이터의 특성에 맞게 효율적으로 데이터를 검색, 처리할 수 있는 질의 언어, 관련 처리 기술, API를 제공함
캐싱 (Caching)	• 대규모 질의에도 고성능 응답 속도를 제공할 수 있는 메모리 기반 캐싱 기술을 적용하는 것이 중요함 • 개발 및 운영에도 투명하고 일관되게 적용할 수 있는 구조임

데이터 저장

- **데이터 저장 시스템**
  - **NoSQL**
    - **데이터 모델: 데이터 저장 방식에 따라 Key-Value 구조, Column 기반 구조, Document 기반 구조로 구분**
      - **Key-Value 데이터베이스**
        - 데이터를 키(Key)와 그에 해당하는 값(Value)의 쌍으로 저장하는 데이터 모델에 기반을 둠
        - 단순한 데이터 모델에 기반을 두기 때문에 관계형 데이터베이스보다 확장성이 뛰어나고 질의 응답시간이 빠름
        - 아마존의 Dynamo DB가 효시. Redis와 같은 인메모리(In- Memory) 방식의 오픈소스 DB가 대표적임
      - **Column 기반 데이터베이스**
        - 데이터를 Row가 아닌 Column 기반으로 저장하고 처리하는 데이터베이스
        - Column과 Row는 확장성을 보장하기 위하여 여러 개의 노드로 분할되어 저장, 관리됨
        - 구글의 Bigtable이 Column 기반 DB의 효시. 여기서 파생된 Cassandra, HBase, HyperTable 등이 대표적임
      - **Document 기반 데이터베이스**

데이터 저장

- **데이터 저장 시스템**
  - **NoSQL**
    - **데이터 모델: 데이터 저장 방식에 따라 Key-Value 구조, Column 기반 구조, Document 기반 구조로 구분**
      - **Document 기반 데이터베이스**
        - 문서 형식의 정보를 저장, 검색, 관리하기 위한 데이터베이스
        - Key-Value 데이터베이스보다 문서의 내부 구조에 기반을 둔 복잡한 형태의 데이터 저장을 지원하고 이에 따른 최적화가 가능함
        - 대표적으로 MongoDB, SimpleDB, CouchDB가 있음

데이터 저장

- **데이터 저장 시스템**
  - **NoSQL**

데이터 모델	설명	제품 예
Key-Value 저장구조	• 가장 간단한 데이터 모델 • 범위 질의는 사용이 어려움(DB에서 지원하는 경우에는 사용 가능) • 응용 프로그램 모델링이 복잡함	• DynamoDB(아마존) • Redis
Column 기반 저장구조	• 연관된 데이터 위조로 읽는데 유리한 구조 • 하나의 레코드를 변경하려면 여러 곳을 수정해야 함 • 동일 도메인의 열 값이 연속되므로 압축 효율이 좋음 • 범위 질의에 유리함	• Bigtable(구글) • Cassandra(아파치) • HBase • HyperTable
Document 기반 저장구조	• 문서마다 다른 스키마가 있음 • 레코드 간의 관계 설명이 가능함 • 개념적으로 RDBMS와 비슷함	• SimpleDB(아마존) • CouchDB(아파치) • MongoDB

데이터 저장

- 빅데이터 저장 시스템 선정을 위한 분석
  - 기능성 비교 분석
    - 데이터 모델
      - 데이터를 테이블로 정리하고 사용하는데 무리가 없다면 RDBMS가 필요하며, NoSQL을 사용할 필요는 없음
      - MongoDB는 RDBMS보다 한층 유연한 스키마를 활용하는 것이 가능하므로 RDBMS에서 문서 중심의 데이터 모델로 전환하는데 도움이 됨
      - Apache CouchDB는 데이터 저장소에 대한 인터페이스로 RESTful HTTP를 지원하기 때문에 웹기반 시스템을 구축하는데 있어서 상대적으로 장점을 가지는 Document 데이터베이스 시스템임
      - 단순 Key-Value 쌍을 저장하여 대규모 사용자와부하 분산을 위한 안정적인 분산 저장소가 필요한 경우, Dynamo 또는 Redis를 선택하면 됨
      - Cassandra, HyperTable, HBase는 Column 기반의 데이터베이스 시스템으로 극단적인 확장성을 보장할 수 있다는 장점을 가짐

데이터 저장

- 빅데이터 저장 시스템 선정을 위한 분석
  - 기능성 비교 분석
    - 확장성
      - 확장성에서는 HBase, Cassandra, HyperTable과 같은 Column 기반 데이터베이스가 가장 뛰어난 것으로 알려져 있음
      - Redis와 같은 인메모리 방식의 데이터베이스와 MongoDB, CouchDB와 같은 Document 데이터베이스가 약간 뒤처지는 것으로 알려져 있음
    - 트랜잭션 일관성
      - 트랜잭션의 일관성은 데이터 수정, 삭제 등의 작업이 빈번하게 일어나는 환경에서는 중요도가 높지만 배치 중심의 하둡 기반 분석환경에서는 중요도가 그리 높지 않음
      - 트랜잭션의 일관성이 중요한 분야에서는 RDBMS를 선택해야 하며, 그렇지 않은 때에만 NoSQL을 선택해야 함

데이터 저장

- 빅데이터 저장 시스템 선정을 위한 분석
  - 기능성 비교 분석
    - 질의 지원
      - MongoDB는 SQL과 유사한 문법에 기반을 두어 쉽게 학습할 수 있는 우수한 질의 인터페이스를 지원함
      - CouchDB도 MongoDB에 비해 뒤처지지 않으며, 뷰 개념을 이해하고 활용하면 간편하게 사용할 수 있음
      - Key-Value기반 데이터베이스의 대표격인 Redis도 풍부한 질의기능을 제공함
      - HBase나 HyperTable은 자체 질의지원 기능은 제공하지 않으나 Hive를 통해 SQL과 유사한 형태의 질의기능 사용 가능
    - 접근성
      - MongoDB를 접근하기 위한 드라이버는 아주 다양하며, 현존하는 주류 라이브러리용 드라이버를 대부분 지원하고 있음
      - CouchDB는 웹 통신을 지원하는 프로그래밍 언어라면 모두 사용 가능함
      - 기타 Redis, HBase, HyperTable, Cassandra는 대부분의 프로그래밍 언어에서 연결 가능하도록 언어 바인딩을 지원

데이터 저장

- 빅데이터 저장 시스템 선정을 위한 분석
  - 분석 방식 및 환경
    - 빅데이터 저장 방식
      - 빅데이터를 파일 시스템으로 저장하는 방식
      - NoSQL 저장 시스템을 사용하는 방식
      - RDBMS에 기반을 둔 데이터 웨어하우스 방식
    - 필요로하는 분석 및 검색결과가 상시로 온라인 형식으로 필요한지, 분석가를 통해 별도의 프로세스를 거쳐 제공받는지 등을 구분하여 저장 방식과 환경을 선택함

데이터 저장

- 빅데이터 저장 시스템 선정을 위한 분석
  - 분석대상 데이터 유형
    - 분석대상이 되는 데이터 유형이
      - 기업 내·외부에서 발생하는 기업 데이터인가?
      - IoT 환경에서 발생하는 데이터인가?
      - 기타 다양한 과학, 바이오, 의학 분야에서 취급되는 데이터인가?

    에 따라 데이터의 Volume, Velocity, Variety, Veracity 등을 고려하여 빅데이터 저장 시스템을 선택함

데이터 저장

- 빅데이터 저장 시스템 선정을 위한 분석
  - 기존 시스템과의 연계
    - 빅데이터 저장 시스템을 선정할 때는 기존 시스템과의 연계성을 반드시 고려하여야 함
      - 저장 대상 데이터의 유형이 대부분 테이블로 정의될 수 있는 형태이고 기존에 RDBMS 기반의 데이터 웨어하우스가 도입된 형태라면 기존 시스템을 그대로 활용하여 저장함
      - 기존에 HDFS만을 활용하여 빅데이터 저장 시스템이 구축되어 있으나 SQL- like 분석 환경을 구축하고자 한다면 HBase를 추가 도입하는 것을 권장함
      - 빅데이터 분석 애플리케이션이 Key-Value 쌍 위주로 처리하는 시스템이 구현되어 있다면 Redis 등을 도입하는 것이 좋음
      - IoT 데이터처럼 다양한 데이터가 지속해서 실시간 발생하는 것을 수집하는 시스템 환경이라면 Key-Value 기반의 데이터베이스를 선정하는 것이 좋음
      - 데이터베이스나 확장성이 중요한 요소라면 Cassandra와 같이 확장성이 보장된 Column 기반의 데이터베이스를 선정하는 것이 좋음

데이터 저장

- **데이터 발생 유형 및 특성**
  - **대용량 실시간 서비스 데이터**
    - 대상 데이터의 용량, 실시간 여부, 정형, 비정형 등 유형 및 요건을 파악하여 빅데이터 저장 계획 수립에 반영함
    - 일반적으로 실시간으로 처리해야 하는 데이터를 스트리밍(Streaming) 데이터라고 부르는데, 대용량의 특성과 무중단 서비스를 보장하는 저장 체계를 구축해야 함
    - 실시간으로 데이터를 처리하기 위해서 사용되는 시스템으로는 스파크(Spark), 스톰(Storm) 등이 있으며, 배치 기반의 대용량 데이터 처리에 특화된 하둡 시스템보다 실시간 대용량 데이터 처리에 특화되어 있음
    - IoT에서 발생하는 센서 데이터, 네트워크 모니터링 데이터, 에너지 관리 분야 데이터, 통신 데이터, 웹 로그, 주식 데이터나 생산 현장에서 발생하는 데이터들이 이에 해당함

데이터 저장

- **데이터 발생 유형 및 특성**
  - **대용량 실시간 서비스 데이터 저장**
    - 실시간 빅데이터 처리를 위해 스톰(Storm)을 사용한다고 가정하면, 스톰은 저장소가 없으므로 외부 저장 시스템과의 연동이 필수적임
    - 다양한 소스의 로그로부터 데이터가 발생하는 환경에서는 스톰을 통하여 데이터를 전처리한 후, HDFS나 MongoDB, Cassandra, HBase와 같은 NoSQL을 저장소로 사용하거나, 데이터를 정규화하여 일반 RDBMS를 저장소로 사용할 수도 있음
    - 대표적인 실시간 데이터 처리 시스템인 스파크(Spark) 역시 내장된 저장소를 제공하지 않기 때문에 외부 빅데이터 저장 시스템과의 연계가 필수적임
    - 실시간 서비스를 웹페이지로 제공하는 것이 필요한 환경에서는 Redis와 같은 메인 메모리 저장 시스템을 저장소로 사용하기도 함

데이터 저장

- **안정성과 신뢰성 확보 및 접근성 제어 계획 수립**
  - **빅데이터 저장 시스템 안정성 및 신뢰성 확보**
    - 안정성 및 신뢰성을 확보하고 보장하기 위해 저장 계획 수립단에서 용량산정이 필요함
    - 조직의 빅데이터 활용목적에 부합하는 현재와 향후 증가 추세를 추정 반영함
  - **접근성 제어계획 수립**
    - 저장 시스템의 사용자와 관리자 유형, 역할 및 기능을 정의하고 각각에 해당하는 제어 계획을 수립함