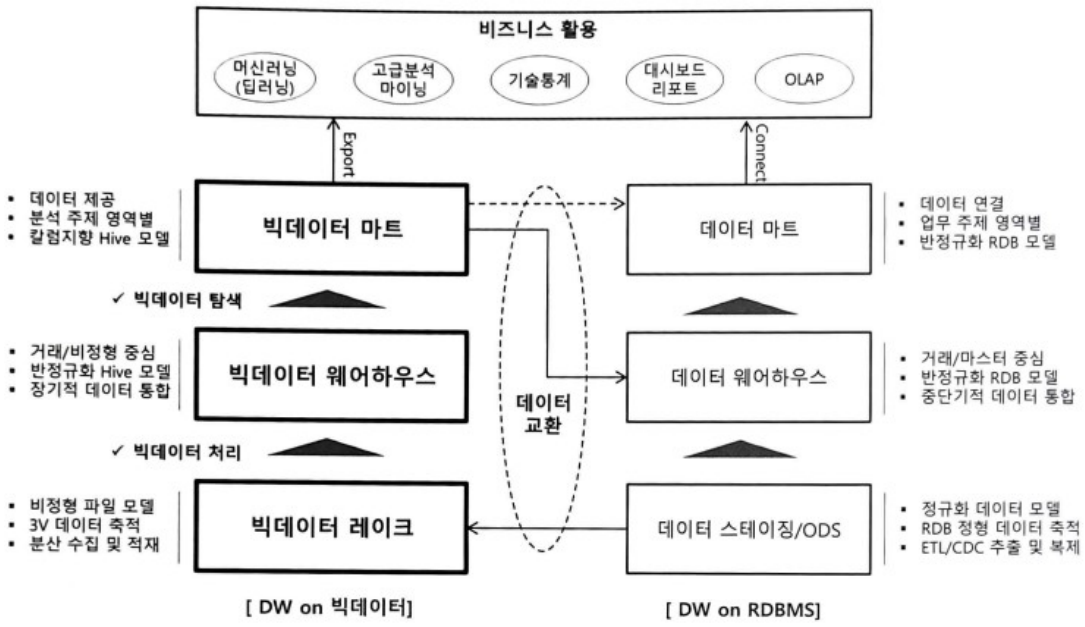
20220427

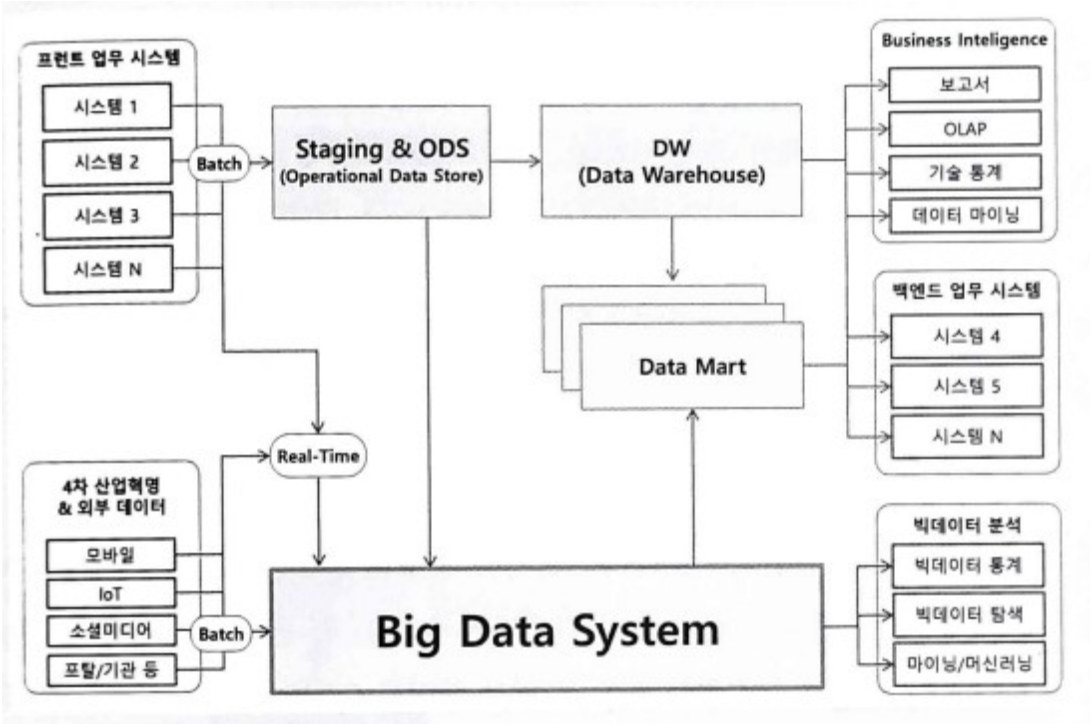
빅데이터 탐색

- **빅데이터 탐색 개요**
  - 빅데이터 처리 및 탐색 영역은 적재된 데이터를 가공하고 이해하는 단계
  - 데이터들의 패턴, 관계, 트렌드 등을 **탐색적 분석**이라고도 한다.
  - 탐색적 분석을 하기 위해서는 **2V(Volume, Variety)**의 비정형 데이터를 정교한 후처리 작업으로 정형화한 저장소가 필요한데 이곳이 **빅데이터 웨어하우스**다.
  - 빅데이터 처리/탐색의 최종 결과물은 빅데이터 웨어하우스 기반의 마트이며, 이를 빅데이터 분석/응용에 활용한다.



  - 빅데이터 기반 **DW(Data Warehouse)**는 크게 3개의 영역으로 구성, **전통적인 RDBMS 기반 DW 구조**와도 유사하다.
  - **레이크 영역**은 플럼, 스톰 등에서 수집한 3V의 데이터들이 모이는 곳이며, 크고 작은 **비정형(반정형) 파일**들이 축적된다.

- **빅데이터 레이크의 파일**들은 빅데이터 처리 기술을 통해 하이브 모델로 가공되는데, 이때 데이터 **추출/정제/검증/분리/통합** 등의 작업을 거쳐 반정규화된 하이브 테이블로 만들어진다.
- 하이브 기반의 빅데이터 웨어하우스가 만들어지면 SQL기반의 다양한 에드훅 분석으로 EDA를 진행, 그 결과를 집계/요약해서 **빅데이터 마트**를 생성
- **데이터 마트**는 외부 시스템에서 빠르게 조회 및 제공돼야 하므로 칼럼지향형 하이브 테이블로 설계
- **하이브리드 DW 아키텍처**
  - 전통적인 데이터 웨어하우스의 단점으로 데이터의 증가에 따른 확장성 부족과 그로 인한 높은 비용 발생 이슈가 있다.
  - 방안으로 기존 RDBMS 기반의 데이터 웨어하우스는 중단기적인 데이터만 보관하고, 장기적인 보관이 필요한 데이터는 빅데이터 웨어하우스에 저장하게 하는 것이다.
  - 대규모 비정형 및 실시간성 데이터도 빅데이터 웨어하우스에 보관함으로써 RDBMS 대비 운용 비용을 절감시킨다.

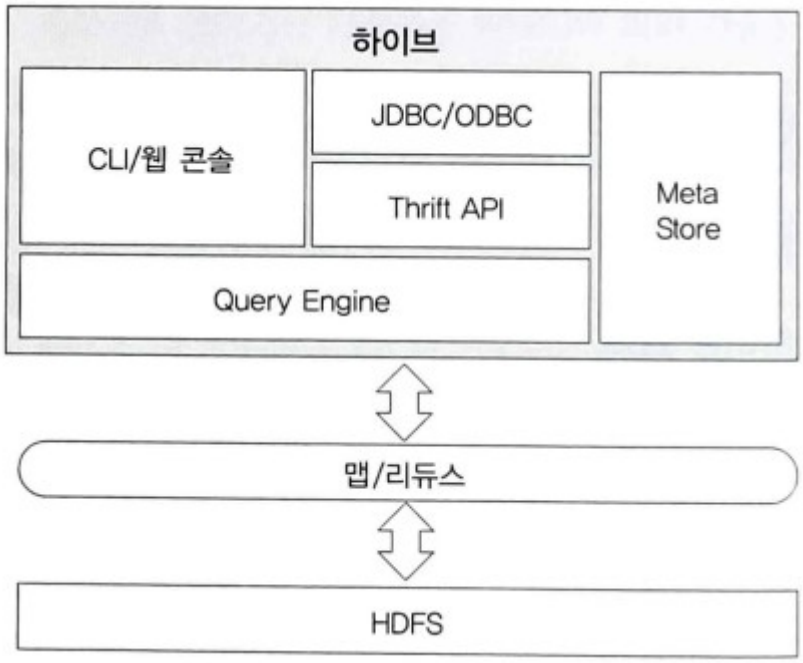


- 빅데이터 탐색에 활용되는 기술
  - **하이브**

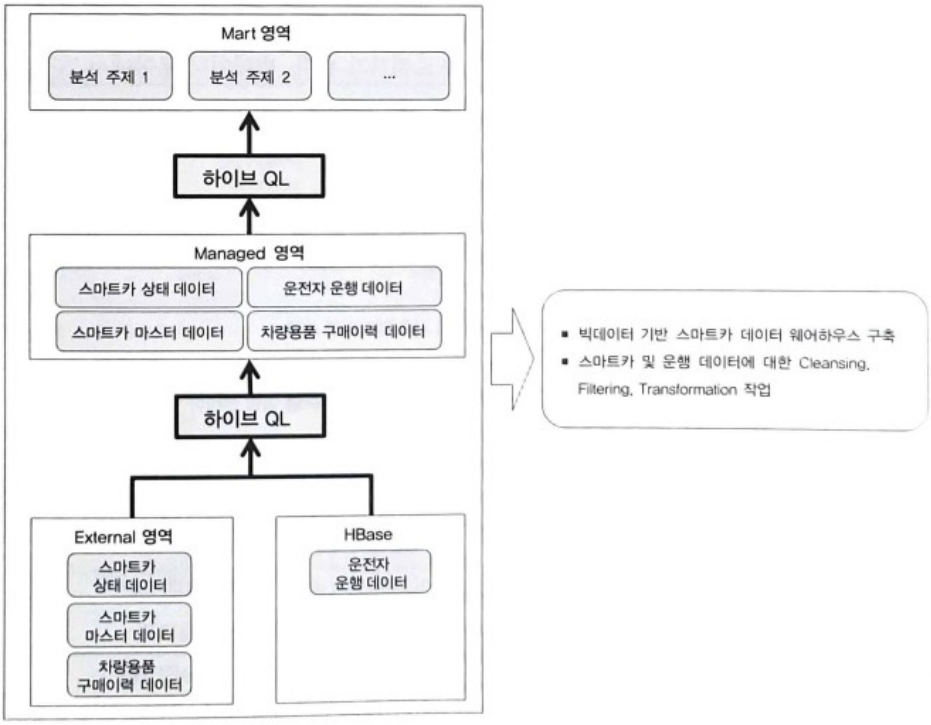
- 하둡 초창기에는 적재된 데이터를 탐색/분석하기 위한 도구로 **맵리듀스(MapReduce)**를 주로 이용
- **맵리듀스**는 복잡도가 높은 프로그래밍 기법이 필요, 접근을 어렵게 만듬
- **SQL과 매우 유사한 방식**으로 하둡 데이터에 접근성을 높인 하이브 개발

주요 구성 요소	CLI	사용자가 하이브 쿼리를 입력하고 실행할 수 있는 인터페이스(Hive Server1 기반의 CLI와 Hive Server2 기반의 Beeline이 있음)
	JDBC/ODBC Driver	하이브의 쿼리를 다양한 데이터베이스와 연결하기 위한 드라이버를 제공
	Query Engine	사용자가 입력한 하이브 쿼리를 분석해 실행 계획을 수립하고 하이브 QL(Query Language)을 맵리듀스 코드로 변환 및 실행
주요 구성 요소	MetaStore	하이브에서 사용하는 테이블의 스키마 정보를 저장 및 관리하며, 기본적으로 더비 DB(Derby DB)가 사용되나 다른 DBMS(MySQL, PostgreSQL 등)로 변경 가능

- **하이브 아키텍처**
  - 가장 큰 특징은 하이브 클라이언트에서 작성한 QL(Query Language)이 맵리듀스 프로그램으로 변환되어 실행



- **하이브 활용 방안**
  - 수집 및 적재한 데이터가 하이브의 External(빅데이터 레이크) 영역에 적재돼있는데, 이를 정제해서 Managed(빅데이터 웨어하우스) 영역으로 옮기고 주제 영역별 Mart를 구성하기 위해 사용



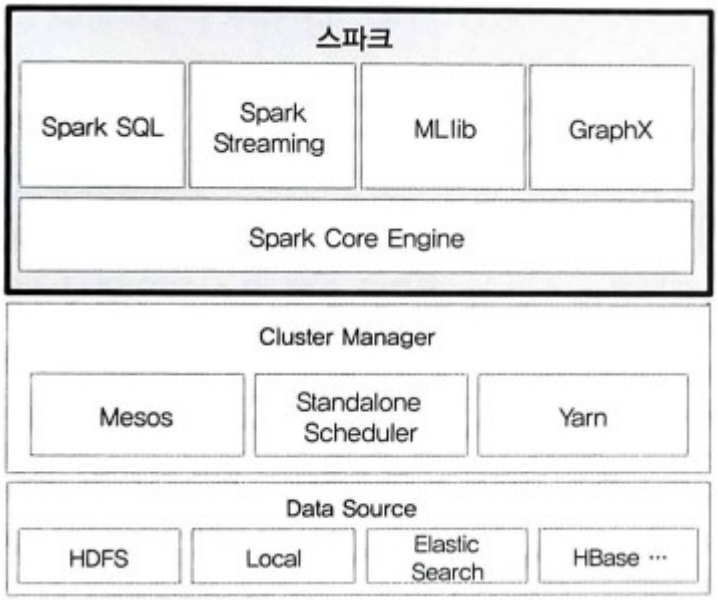
- **피그**
  - 하둡 에코시스템 가운데 하이브와 유사한 목적으로 맵리듀스의 복잡성을 해결하기 위한 피그(Pig)라는 프로젝트가 있다.
  - SQL대신 피그 라틴(Pig Latin)이라는 언어를 제공해서 하이브보다는 절차적인 요소가 많이 사용되는 특징이 있다.

스파크

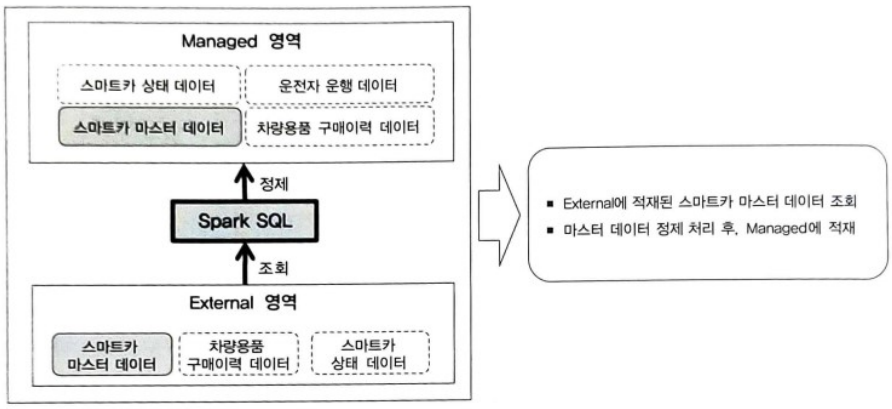
- 복잡한 맵리듀스를 하이브 QL로 래핑해 접근성을 높일 수 있었지만 맵리듀스 코어를 그래도 사용함으로써 성능면에서는 만족스럽지 못함.
- 반복적인 대화형 연산 작업에서는 하이브가 적합하지 않았다
- 이런 단점을 극복하기 위해 **스파크**가 개발

주요 구성 요소	Spark RDD	스파크 프로그래밍의 기초 데이터셋 모델
	Spark Driver / Executors	Driver는 RDD 프로그램을 분산 노드에서 실행하기 위한 Task의 구성, 할당, 계획 등을 수립하고, Executor는 Task를 실행 관리하며, 분산 노드의 스토리지 및 메모리를 참조
	Spark Cluster Manager	스파크 실행 환경을 구성하는 클러스터 관리자로 Mesos, YARN, Spark Standalone이 있음
	Spark SQL	SQL 방식으로 스파크 RDD 프로그래밍을 지원
	Spark Streaming	스트리밍 데이터를 마이크로타임의 배치로 나누어 실시간 처리
	Spark MLib	스파크에서 머신러닝 프로그래밍(군집, 분류, 추천 등)을 지원
	Spark GraphX	다양한 유형의 네트워크(SNS, 하이퍼링크 등) 구조 분석을 지원

- **스파크 아키텍처**
  - 스파크의 가장 큰 특징은 **고성능 인메모리 분석**이다
  - 대량의 데이터를 로드하고 생성함으로써 **높은 IO(Input,Output)발생**과 그로 인한 **레이턴시**(lactency)를 피할 수 없다는 단점을 극복하기 위해서 데이터 가공 처리를 인메모리에서 빠르게 처리한다.
  - **스파크 SQL. 스파크 스트리밍, 스파크 머신러닝** 등의 기능을 제공하고 있어 활요성이 높고 다양한 클라이언트 언어(파이썬, 자바, 스칼라 등)와 라이브러리를 지원해 범용성도 뛰어나다.
  - 데이터소스 영역은 높은 호환성을 보장함으로써 **HDFS, HBase,카산드라(Cassandra), 일렉스틱 서치(Elastic Search)** 등을 연결해 이용할 수 있다.



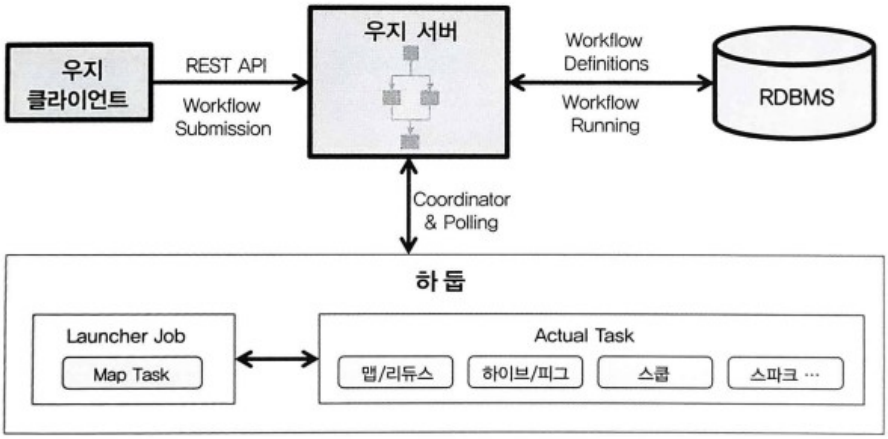
  - 스파크 활용 방안
    - 다양한 클라이언트 프로그래밍 언어(파이썬, 자바, 스칼라 등)를 지원하고, SQL을 이용해 데이터에 엑세스할 수도 있다.



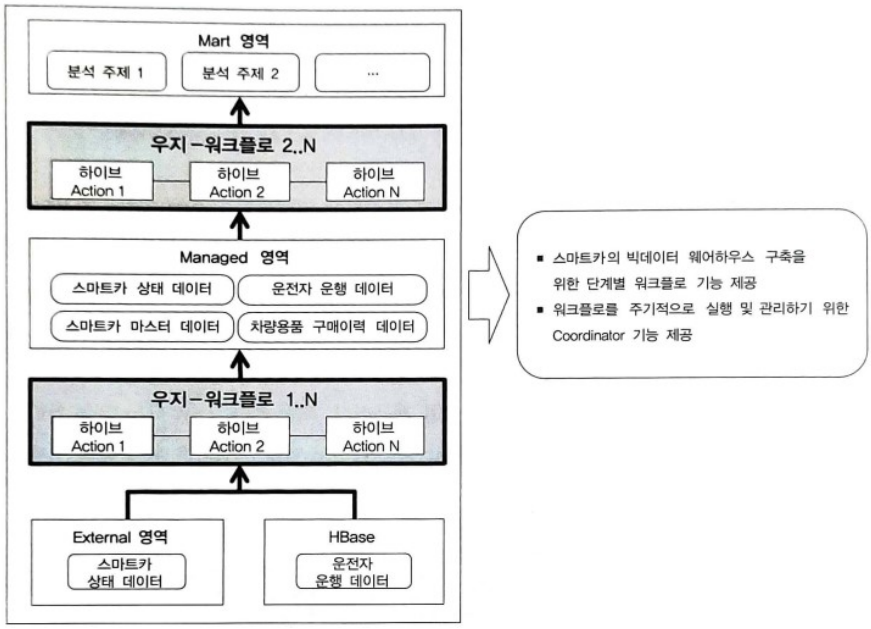
- **우지**
  - 빅데이터의 처리, 탐색, 분석하는 과정은 복잡한 선후행 관계를 맺고 반복적으로 진행된다.
  - 복잡한 데이터 파이프라인 작업을 위해 방향성 있는 **비순환 그래프(DAG : Direct Acyclic Graph)**로 잡의 시작, 처리, 분기, 종료점 등의 액션(Action) 등을 정의하는 워크플로가 **아파치 우지(Apache Oozi)**다.

주요 구성 요소	Oozie Workflow	주요 액션에 대한 작업 규칙과 플로우를 정의
	Oozie Client	워크플로를 Server에 전송하고 관리하기 위한 환경
주요 구성 요소	Oozie Server	워크플로 정보가 잡으로 등록되어 잡의 실행, 중지, 모니터링 등을 관리
	Control 노드	워크플로의 흐름을 제어하기 위한 Start, End, Decision 노드 등의 기능을 제공
	Action 노드	잡의 실제 수행 태스크를 정의하는 노드로서 하이브, 피그, 맵리듀스 등의 액션으로 구성
	Coordinator	워크플로 잡을 실행하기 위한 스케줄 정책을 관리

  - **우지 아키텍처**
    - 우지 클라이언트에서 작성한 워크플로는 우지 서버로 전송되어 메타화되고 RDBMS에 저장된다.
    - 우지 서버에 있는 **Coordinator**는 우지에 등록된 워크플로를 스케줄링해준다.
    - 실행 중인 테스크의 라이프 사이클을 우지 서버가 시작부터 종료까지 추적하면서 모니터링 정보를 제공한다.



- **우지 활용 방안**
  - 적재된 데이터를 External → Managed → Mart로 이동시키기 위해 다양한 하이브 QL들이 이용되고, 이를 **약속된 시간**에 따라 스케줄링 해서 실행해야 하는데, 이때 우지의 워크플로를 활용



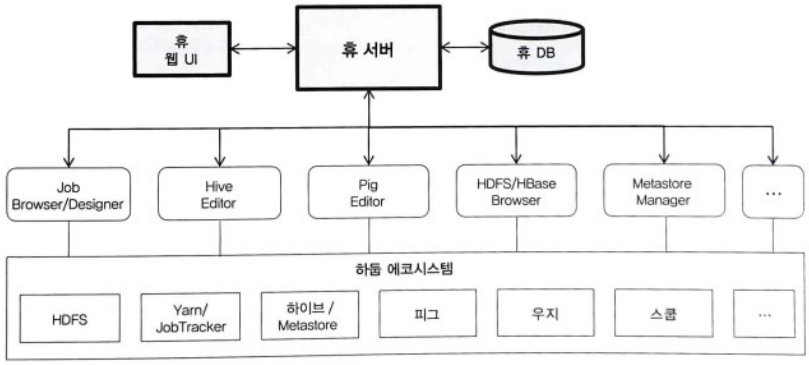
▪ 휴

- 빅데이터 탐색/분석은 장기간의 반복작업이면서 그 과정에 있어 많은 도구들이 활용된다.
- 이를 일반 분석가 또는 업무 담당자들이 각 서버에 직접 접속해 사용하기 어려움이 많다.
- 이러한 기술의 복잡도를 숨기고 접근성과 편의성을 높인 소프트웨어 중 하나가 **휴(Hue)**이다.

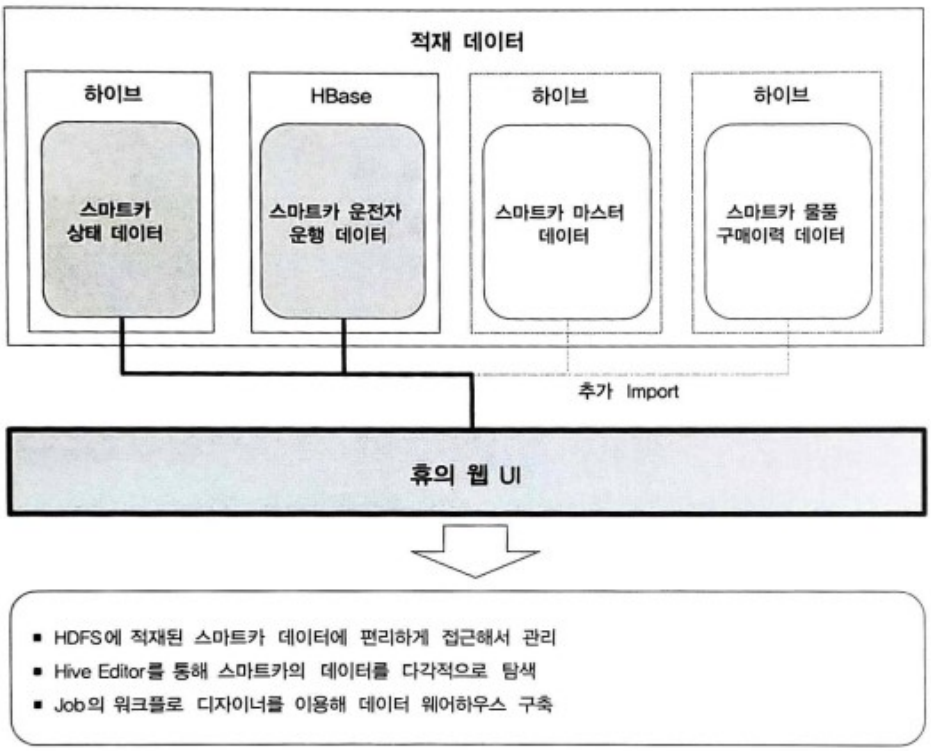
- **휴**는 다양한 하둡의 에코시스템의 기능들을 웹 UI로 통합 제공한다.

주요 구성 요소	Job Designer	우지의 워크플로 및 Coordinator를 웹 UI에서 디자인
	Job Browser	등록한 잡의 리스트 및 진행 상황과 결과 등을 조회
	Hive Editor	하이브 QL을 웹 UI에서 작성, 실행, 관리
	Pig Editor	피그 스크립트를 웹 UI에서 작성, 실행, 관리
	HDFS Browser	하둡의 파일시스템을 웹 UI에서 탐색 및 관리
	HBase Browser	HBase의 HTable을 웹 UI에서 탐색 및 관리

- **휴 아키텍처**
  - 하둡 에코시스템들을 통합하기 위해 자체 플러그인을 설치하거나 API를 연동해서 에코시스템들의 주요 기능들을 웹 UI로 제공한다.



- **휴 활용 방안**
  - 휴에서는 HDFS, HBase, 하이브, 임팔라를 편리하게 사용하기 위한 웹 에디터를 제공
  - 휴의 Job Designer를 이용해 우지의 워크플로를 영역별로 작성하고 실행



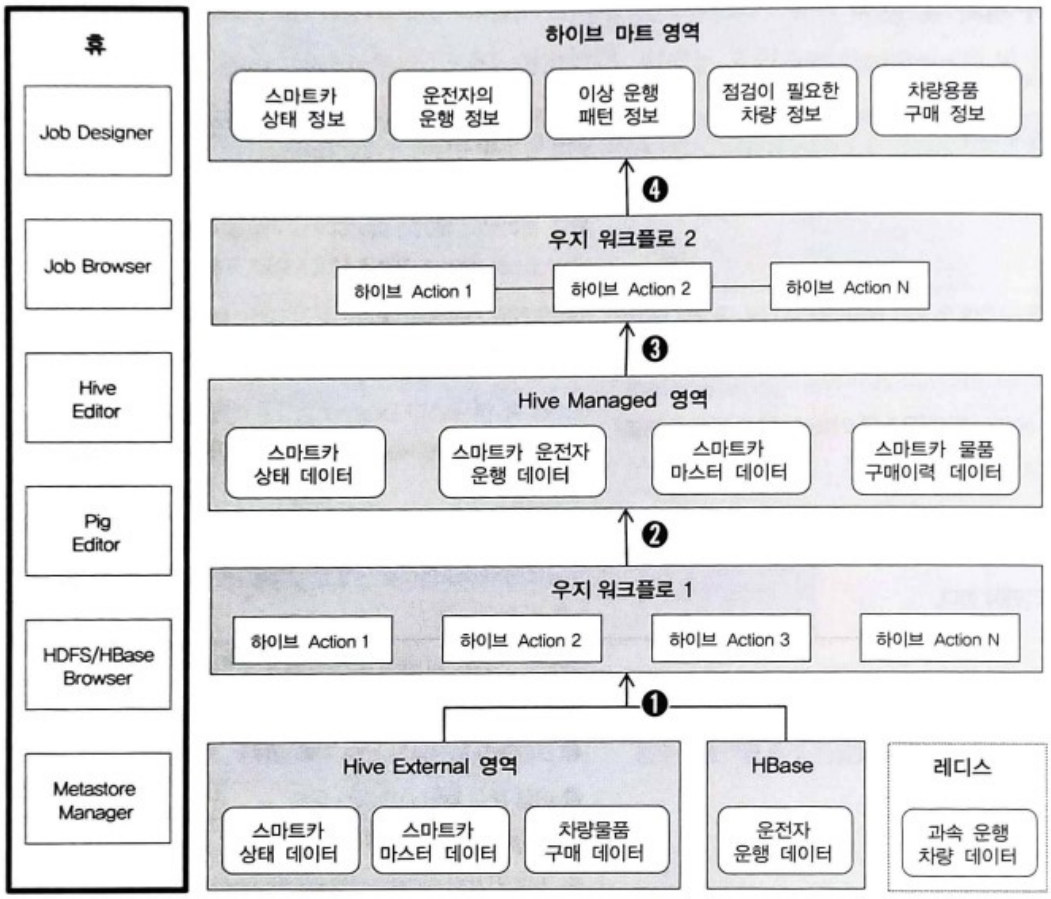
- 탐색 파일럿 실행 1단계 - 탐색 아키텍쳐

  ▼ 요구사항 1 : 다양한 장치로부터 발생하는 로그 파일을 수집해서 기능별 상태를 점검

  ▼ 요구사항 2 : 정보가 담긴 로그를 실시간으로 수집해서 분석

탐색 요구사항 구체화	분석 및 해결 방안
1. 적재된 데이터는 하이브의 데이터 웨어하우스로 관리돼야 한다.	초기 HDFS에 적재된 영역을 하이브의 External 영역으로 정의하고, 하이브의 데이터 웨어하우스 기능을 이용해 External → Managed → Mart 영역을 단계적으로 구성
2. 데이터 마트 구축에 필요한 데이터를 추가로 구성할 수 있어야 한다.	스마트카의 기본정보 데이터, 운전자의 차량용품 구매 이력 데이터셋을 HDFS 명령어로 External 영역으로 추가 적재
3. 하이브의 데이터 웨어하우스의 이력성 데이터들은 일자별로 관리돼야 한다.	데이터 웨어하우스의 External 영역은 작업 처리일을 기준으로 파티션을 구성하며, Managed 영역은 데이터 생성일 기준으로 파티셔닝
4. 분석 마트가 만들어지는 일련의 과정들은 워크플로로 만들어져 관리돼야 한다.	데이터 웨어하우스를 만들기 위한 하이브 QL을 Job Designer에 등록해서 워크플로로 만들고 완성된 워크플로는 스케줄러에 등록 및 관리
5. 최종 마트로 만들어질 데이터셋들은 주제 영역별로 구성돼야 한다.	스마트카 빅데이터 분석 마트의 주제 영역을 4+1개로 확장 ❶ 스마트카의 상태 모니터링 정보 ❷ 스마트카 운전자의 운행 기록 정보 ❸ 이상 운전 패턴 스마트카 정보 ❹ 운전자의 차량용품 구매 이력 정보 ❺ 긴급 점검이 필요한 스마트카 정보

- **탐색 아키텍처**



        - External에 적재된 데이터를 휴에서 제공하는 Hive Editor를 이용해 SQL과 유사한 방식으로 조회.
        - External과 HBase에 적재된 데이터를 작업일자 기준으로 후처리 작업
        - Managed에 만들어진 데이터는 곧바로 탐색과 분석에 활용
        - 데이터 엔지니어 또는 분석가들이 많은 시간과 노력을 들여 다양한 애드훅 분석 작업들을 반복 수행하며 가치 있는 정보들이 발견
- 탐색 파일럿 실행 2단계 - 탐색 환경 구성
    - **하이브 설치**
        - HBase가 포함된 항목 선택
        - WebHCat Server를 제외한 모든 역할을 Server02에 지정
    - **우지 설치**
        - HBase가 포함된 항목 선택
        - 역할을 Server02에 지정
        - 구성에서 Launcher Memory에 기본 메모리 값을 2GB에서 1GB로 수정
    - **휴 설치**
        - 휴 설치하기 위해서는 Python 2.7이 설치돼 있어야 한다.
        - python -V 를 통한 파이썬 버전 확인
        - **파이썬 2.7 설치**
            - yum install centos-release-scl
            - yum install scl-utils (Nothing to do가 뜰 시 바로 파이썬 설치)
            - yum install python27
            - source /opt/rh/python27/enable 재등록
            - 설치 에러 날 시 참조

              https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e523ba47-8ade-45ba-a04a-2f11cda06c54/centos6_yum_backup.txt

            - 이후 python —version으로 버전확인

- **파이썬 패키지 psycopg2를 설치 (휴가 동작하기 위한 패키지)**
    - yum install python-pip
    - yum install postgresql-devel
    - bash -c “source /opt/python27/enable; pip install psycopg2==2.6.2 —ignore-installed”
  - **클러스터에 서비스 추가로 휴 설치**
    - 모든 컴포넌트를 포함한 항목 선택
    - Hue Server를 Server02로 지정하고 Load Balncer를 선택 해제
    - 휴 → 구성에서 “시간대”를 입력후 Asia/Seoul로 변경
    - 휴 → 구성에서 “HBase Thrift 서버”를 입력후 server01로 변경(없을시 냅둠)
- **스파크 설치**
  - HBase가 포함된 항목 선택
  - Sever02에 역할 지정
  - 스파크를 YARN에서 작동하도록 구성했으므로 YARN 서비스와 스파크를 재시작
  - 스파크 히스토리 서버 → http://server02.hadoop.com:18088 로 접근하여 동작 확인 (모니터링 할 수 있다.)
- 탐색 파일럿 실행 3단계 - 휴를 이용한 데이터 탐색
  - **휴 접속**
    - 방법 1 : http://server02.hadoop.com:8888/
    - 방법 2 : CM의 홈 → Hue → 상단의 Hue 웹 UI
    - 아이디와 패스워드 모두 “admin”으로 지정
  - **HDFS에 적재된 데이터 확인**
    - 파일 메뉴 선택
    - root 경로(”/”)를 클릭해 /pilot-pjt/collect/car-batch-log/wrk_date=날짜 경로까지 이동
    - 로그 파일 클릭하면 적재된 파일의 내용을 볼 수 있다.
    - Hbase선택 후 테이블 클릭

        - 로우키를 기준으로 적재된 정보를 확인 할 수 있다.
        - HBase Browser를 이용하면 HBase에 적재된 컬럼 기반 데이터를 직관적으로 조회 및 관리할 수 있다.
- 탐색 파일럿 실행 4단계 - 데이터 탐색 기능 구현 및 테스트
    - **하이브를 이용한 External 데이터 탐색**
        - http://server02.hadoop.com:8888/ 에서 쿼리 → 편집기 → Hive를 선택
        - sql쿼리를 작성하거나 파일을 드래그&드랍
        - Alter table을 이용한 파티션 생성 (간혹 데이터만 적재한 후 Add Partition을 수행하지 않고 데이터를 조회해하면 데이터가 조회되지 않는다.)
        - select 절을 이용하여 조회 (”limit” 절을 이용해 부하를 최소화하고 빠르게 데이터를 조회해 볼 수 있다.)
        - 하이브 특징
            1. 하이브 쿼리는 맵리듀스로 변환되어 실행
                a. 하나의 하이브 쿼리는 복잡도에 따라 여러 개의 잡이 순차적으로 만들어지고,
                b. 다시 잡 안에서는 데이터의 크기와 조건절에 따라 여러 개의 맵과 리듀스 작업이 생성되어 실행된다.
            2. 대화형 온라인 쿼리 사용에 부적합
                a. 하이브 쿼리는 맵리듀스 변환되어 실행되기까지는 최소 10 ~ 30초 이상의 준비시간이 필요하며 처리량이 높은 대규모 배치 작업이 최적화되어 있다.
            3. 데이터의 부분적인 수정 불가
                a. HDFS의 특징으로 HDFS를 기반으로 작동하는 하이브는 그 특징을 그대로 계승했다고 할 수 있다. 그로 인해 하이브 테이블에서는 부분 수정 및 삭제 처리를 할 수 없다.
            4. 대규모 병렬분산 처리가 불가능한 경우
                a. 하이브는 처리량이 높은 대규모 병렬분산 처리에 최적화돼 있지만 일부 요건에 따라 대규모 분산처리가 어려울 수 있다.
            5. 트랜잭션 관리 기능이 없어 롤백 처리 불가
                a. 하나의 하이브 쿼리는 여러 개의 잡과 맵리듀스 프로그램으로 실행되며, 로컬디스크에 중간 파일들을 만들어낸다.

b. 이때 특정 맵리듀스 작업하나가 실패하면 이미 성공한 잡들이 롤백 처리되지 않는다.

- **하이브를 이용한 HBase 데이터 탐색**
  - http://server02.hadoop.com:8888/ 에서 쿼리 → 편집기 → Hive를 선택
  - sql 쿼리를 이용하여 External 테이블을 생성