



## 저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

이학석사학위논문

**전력 데이터를 위한 분산 스트림 큐브 기반의  
효율적인 온라인 다차원 분석 시스템 구축**

장 종 원

강원대학교 대학원

컴퓨터과학과

2020년 2월



김 진 호 교수지도

이학석사 학위논문

**전력 데이터를 위한 분산 스트림 큐브 기반의  
효율적인 온라인 다차원 분석 시스템 구축**

**Efficient On-line Multidimensional Analysis System**

**Based on Distributed Stream Cube for Electric Power Data**

강원대학교 대학원

컴퓨터과학과

장 종 원

장종원의 석사 학위논문을  
합격으로 판정함

2019년 12월

심사위원장      문    양    세    인

위      원      김    진    호    인

위      원      임    현    승    인

# 전력 데이터를 위한 분산 스트림 큐브 기반의 효율적인 온라인 다차원 분석 시스템 구축

장 종 원

강원대학교 대학원 컴퓨터학과

오늘날 4차 산업혁명의 주요 기술 중 하나인 IoT(Internet of Things)의 확산과 스마트 그리드(Smart Grid) 등의 발전으로 다양한 기기나 센서에서 생성하는 데이터의 양은 방대하다. 과거에는 배치 데이터를 처리하는 것에 집중하였으나 현재는 IoT나 모니터링 시스템 등에서 스트리밍 데이터 분석이라는 영역이 부각되고 있다. 스트림 데이터의 분석은 즉시성이 요구되는 부분을 포착하여 해결함으로써 스트림 데이터의 가치를 더할 수 있게 한다. 이러한 대용량의 데이터를 효율적으로 저장하고 분석하기 위해서 OLAP과 데이터 큐브가 연구되었으며, 스트림 처리를 위한 *Stream Cube*와 분산 큐브와 같은 연구들이 있었다. 본 논문에서는 하둡 에코시스템 기반의 오픈소스 OLAP 도구 2가지를 이용하여 시스템을 구축하였는데, Apache Kylin를 활용하여 배치 데이터를 그리고 Apache Druid를 활용하여 스트림 데이터를 처리하는 시스템을 구축하였다. 전력 데이터를 전처리하여 스타 스키마를 구축하고 배치 형태의 데이터 큐브를 생성하여 간단한 몇가지 예제를 통해 분석하고 시각화하였다. 그리고 Apache Kafka를 활용하여 스트림 데이터를 전송하고 이를 pull 방식으로

수집하여 원본을 저장하거나 요약하여 저장하도록 한다. 데이터는 수집하는 동시에 실시간으로 질의를 통해 분석할 수 있다. 이후에 Apache Superset을 이용하여 시각화하고 실시간 모니터링을 위한 대시보드를 구성하였다. 시스템을 구축한 결과로 필요에 따라 요약한 데이터가 원본 데이터보다 저장 공간을 절약하는 대신 질의 시에 초기에 설정한 수준으로의 질의가 가능하였다.

#### □ 핵심주제어

빅데이터, 전력 데이터, 데이터 큐브, OLAP, 데이터 웨어하우스

## 감사의 글

연구실에 4학년에 들어오게 되어 석사 졸업까지 하게되었는데, 이 시간동안 제게 많은 도움과 가르침 그리고 격려해주신 모든 분들에게 감사하다고 말씀드리고 싶습니다.

먼저 정말 바쁘신 와중에도 제가 논문을 작성할 수 있도록 관심가져주시고 걱정해주신 저의 지도교수님 김진호 교수님께 감사드립니다. 제가 집중하지 못하는 시기에는 따끔한 충고로 제 정신을 깨워주셨고 교수님의 가르침 덕분에 부족한 논문을 채울 수 있었습니다. 또한, 저의 논문에 심사를 맡아주시고 관심 가져주신 임현승 교수님, 같이 과제하면서 신경 써주시고 심사를 맡아주신 문양세 교수님께도 정말 감사드립니다.

그리고 연구실 생활을 즐겁게할 수 있도록 도와준 사람들에게 감사드립니다. 웃어주시며 방향을 잡아주시고 좋은 말씀해주신 조선화 선배님과 수안이형, 연구실에 들어가도록 도와준 먼저 졸업한 영석이형, 원하는 방향을 잡아보라고 도와준 석이형, 직전까지 함께 석사 생활한 성진이형, 학부생 때 같이 연구실에 있었던 현배형, 연구실 생활 처음부터 끝까지 함께한 민규, 아쉽게 헤어졌지만 같이 오래 생활한 희상이형, 지금 학부생인 범준이, 지형이, 호빈이, 마지막에 함께한 재훈이형까지 모두 감사드립니다. 그리고 석사생활을 함께하며 도움 많이 받았던 12학번 동기들에게도 감사드립니다.



마지막으로 항상 응원하고 걱정해주시는 사랑하는 가족에게 감사드립니다. 항상 제 걱정만 하시는 어머니와 아버지, 외국에 있지만 한국에 오면 가장 반갑게 맞이해주는 누나까지 모두 감사드리고 사랑합니다.

# 목 차

<b>1</b>	<b>서 론</b>	<b>1</b>
<b>2</b>	<b>배경</b>	<b>4</b>
2.1	아파치 하둡(Apache Hadoop) .....	4
2.2	아파치 카프카(Apache Kafka) .....	7
2.3	아파치 기린(Apache Kylin) .....	9
2.4	아파치 드루이드(Apache Druid) .....	11
2.5	아파치 슈퍼셋(Apache Superset) .....	15
<b>3</b>	<b>관련 연구</b>	<b>18</b>
3.1	OLAP(One-Line Analytical Processing) .....	18
3.2	데이터 큐브 .....	20
<b>4</b>	<b>전력 데이터 및 데이터 전처리</b>	<b>23</b>
4.1	전력 데이터 .....	23
4.2	데이터 가공 및 전처리 .....	24
4.2.1	Batch 형태의 입력을 위한 전처리 .....	24
4.2.2	Stream 형태의 입력을 위한 전처리 .....	25
4.3	전력 관련 연구 .....	26
<b>5</b>	<b>Apache Kylin을 이용한 전력 데이터의 다차원 큐브</b>	<b>27</b>
5.1	전력 데이터 다차원 분석 시스템 구성 및 동작 .....	27
5.2	팩트 테이블과 스타 스키마 .....	28
5.2.1	팩트 테이블 .....	28
5.2.2	스타 스키마(Star Schema) .....	28

5.3	Kylin을 이용한 Batch 형태의 큐브 생성 .....	30
5.3.1	가공한 AMI 데이터를 Hive에 테이블 형태로 저장 .....	30
5.3.2	Kylin에서 Hive 데이터와 동기화 .....	32
5.3.3	Kylin에서 데이터 모델 생성 .....	32
5.3.4	Kylin에서 데이터 큐브 생성 .....	33
5.3.5	기타 설정 값 추가 .....	34
<b>6</b>	<b>Apache Druid를 이용한 전력 스트림 데이터의 분산 OLAP</b>	<b>35</b>
6.1	전력 스트림 분석 시스템 구성 및 동작 .....	35
6.2	스트림 데이터 처리 .....	36
6.2.1	스트림 데이터 처리 방식 .....	36
6.3	다른 집계 수준의 데이터 구성 .....	38
6.4	질의 처리 예시 .....	38
<b>7</b>	<b>시스템 구축 및 분석</b>	<b>40</b>
7.1	시스템 구축 환경 .....	40
7.2	Kylin 분석 예시 .....	41
7.2.1	생성한 큐브의 결과 .....	41
7.2.2	다차원 질의 및 시각화 예시 .....	41
7.3	Druid 분석 예시 .....	44
7.3.1	질의 결과 .....	45
7.3.2	Superset을 통한 시각화 .....	47
<b>8</b>	<b>결 론</b>	<b>52</b>
<b>9</b>	<b>참고 문헌</b>	<b>54</b>
	<b>영문 요약</b>	<b>56</b>

## 그림 목차

그림 1. 단어 별 빈도수 출력 맵리듀스 프로세스 .....	5
그림 2. 하둡 에코시스템 구성도 .....	7
그림 3. Kafka의 기본 구성 요소 .....	8
그림 4. 카프카의 동작 과정 .....	8
그림 5. Kylin의 아키텍처[7] .....	10
그림 6. Apache Druid의 아키텍처 .....	12
그림 7. 유저 별 row의 수 집계에 대한 테이블 형태 시각화 .....	16
그림 8. 국가별 row의 수 집계에 대한 pie chart 시각화 .....	16
그림 9. 전력 데이터 다차원 분석 시스템 구성 .....	27
그림 10. 스타 스키마 예시 .....	30
그림 11. 스타 스키마 구조 .....	33
그림 12. 전력 스트림 분석 시스템 구성 .....	35
그림 13. Stream Push 방식 도식화 .....	37
그림 14. Stream Pull 방식 도식화 .....	38
그림 15. Druid 질의 탭 .....	39
그림 16. 일별/업종별 전력 사용량 집계 질의 .....	42
그림 17. 일별/업종별 전력 사용량 집계 결과 .....	42
그림 18. 일별/업종별 전력 사용량 집계 시각화 .....	43
그림 19. 월별 전력 사용량 집계 질의 .....	43
그림 20. 월별 전력 사용량 집계 결과 .....	44
그림 21. 월별 전력 사용량 집계 시각화 .....	44
그림 22. 시간/미터기별 전력 관련 컬럼 집계 질의 .....	46
그림 23. 분 단위의 원본 데이터 질의 결과 .....	46

그림 24. 시간 단위의 데이터 질의 결과 .....	46
그림 25. 일 단위의 데이터 질의 결과 .....	46
그림 26. 15분 단위 구간별 전력 사용량 합 집계 시각화 .....	48
그림 27. 시간 단위 최대 전력 사용량 집계 시각화 .....	49
그림 28. 하루 단위 전력 사용량 Top 5 미터기 시각화 .....	49
그림 29. 월 단위 전력 사용량 대비 무효전력량 시각화 .....	50
그림 30. 분기별 총 전력 사용량 집계 시각화 .....	50
그림 31. 위의 4가지 차트를 포함하는 실시간 대시보드 .....	51

## 표 목차

표 1. 팩트 테이블이 될 AMI 데이터 테이블 .....	31
표 2. Date 상세 차원 테이블이 될 날짜 테이블 .....	31
표 3. Code2Type 상세 차원 테이블이 될 미터기별 업종 테이블 .....	31
표 4. Kylin 큐브 빌드 결과 .....	41

# 1 서 론

오늘날 IoT(Internet-of-Things) 기술을 사용하는 전자기기 사물 개수의 증가 및 스마트 그리드의 발달로 인해 다양한 기기 및 센서에서 생성하는 데이터의 양은 가히 폭발적이라 할 수 있고 점점 증가하는 추세이다. 이러한 데이터 소스에서 생성되는 데이터는 센서 데이터, 웹 트래픽 및 보안장비의 로그 등과 같은 데이터의 값이 시간과 함께 연속성을 지니는 스트림 데이터 혹은 스트리밍 데이터라고 불린다. 수집에서 수백억개에 달하는 센서와 기기가 생성하는 스트림 빅데이터의 실시간 처리 및 분석에 관한 기능이 매우 중요한 요소로 부각되고 있다.

기존의 빅데이터 분석 및 처리 연구의 대부분은 배치 데이터처리 위주였다. 하지만 최근 금융 서비스의 트랜잭션, 의료 서비스의 모니터링, IoT 장치 등 실시간으로 데이터를 배포하는 소스들이 많아지면서 실시간 빅데이터 처리 기술이 요구되고 있고, 이런 식으로 실시간으로 분석하여 시장 상황에 대응하는 것에 대한 수요가 증가하는 추세이다. 이에 따라 스트리밍 데이터나 실시간 데이터 처리를 도와주는 프레임워크가 많이 등장하고 있다. 아직은 위와 같은 프레임워크를 이용한 연구가 적고 본 논문에서 사용하고자 하는 오픈소스 프레임워크가 아직 인큐베이팅 단계이지만 커뮤니티 및 기업에서 사용 등 많은 곳에서 사용 및 연구되고 있다.

대용량의 센서 배치 데이터와 스트리밍 데이터를 데이터 큐브로 만들기 위해서는 많은 저장 공간과 높은 계산 비용, 그리고 이를 분석하기 위한 시스템이 필요하다. 이를 단일

머신에서 처리하기에는 고사양의 서버가 필요하고 데이터의 양은 계속 증가할 것이기 때문에 한계점이 있다. 이를 해결하기 위해 분산 환경에서의 시스템 구축이 필요하고 확장성(Scale-Out)과 고가용성(High-Availability)를 위해 클러스터로 구성하기 위해 많은 BI 도구 및 플랫폼이 생겨나고 연구되고 있다.

본 논문에서는 대용량의 센서 데이터를 저장 및 분석하기 위해 분산 OLAP 도구를 활용한다. 배치 데이터의 경우 Apache Kylin을 활용하고 스트림 데이터의 경우 Apache Druid를 활용한다. 2개의 OLAP 도구를 이용하여 대용량의 데이터를 분석하고 시각화하는 시스템을 구축한다. Apache Kylin을 활용하기 위해서 하둡 에코시스템이 필요한데 Apache Hadoop과 Apache HBase, Apache Hive가 필요하다. 이를 위해서 다음 장에서 하둡 에코 시스템에 대해 살펴본다.

본 논문의 구성은 다음과 같다. 제2장에서는 하둡 에코시스템들에 대해 살펴본다. 먼저 베이스가 되는 Apache Hadoop을 살펴보고 스트리밍 처리를 위한 Apache Kafka를 살펴보고 OLAP 도구인 Apache Kylin과 Apache Druid에 대해 살펴본다. 그리고 이를 시각화하고 모니터링하기 위해 Apache Superset에 대해 살펴본다. 제3장에서는 OLAP과 데이터 큐브에 대해 살펴보고 관련 연구들을 살펴본다. 제4장에서는 전력 데이터에 대해 설명하고, 데이터 전처리에 대해 다룬다. 그리고 전력 데이터에 관한 관련 연구에 대해 살펴본다. 제5장에서는 Apache Kylin을 이용한 데이터 큐브에 관하여 설명한다. 제6장에서는 Apache Druid를 이용한 스트림 데이터의 분산 OLAP에 대해 설명한다. 제7장에서는 시스템 구축 및 분석에 관해



살펴보고 각 시스템의 예시에 대해 살펴본다. 마지막으로 제8장에서는 논문의 결론으로 구성되어 있다.

## 2 배경

본 장에서는 본 논문에서 사용하는 플랫폼들에 대해 설명한다. 제2.1절에서는 Apache Hadoop에 대해 설명하고 제2.2절에서는 Apache Kafka에 대해 설명하고 제2.3절에서는 Apache Kylin에 대해 설명한다. 제2.4절에서는 Apache Druid에 대해 설명하고, 마지막으로 제2.5절에서는 Apache Superset에 대해 설명한다.

### 2.1 아파치 하둡(Apache Hadoop)

아파치 하둡은 더그 커팅과 마이크 캐퍼렐라에 의해 2006년에 개발되었고 이후에 아파치 재단으로 넘어가 오픈소스 소프트웨어로 현재까지 개발 및 사용되고 있다[1]. 하둡의 등장배경은 과거에 대량의 데이터를 처리하기 위해서는 고성능 서버가 필요하였고 비용이 매우 컸다. 이를 위해 여러 대의 저렴한 PC로 고비용의 서버를 대체하기 위하여 구상되었다. 하둡은 HDFS(Hadoop Distributed File System, 하둡 분산 파일 시스템)[2]와 맵리듀스(MapReduce) 프레임워크[3]로 구성된다. HDFS란 여러 대의 서버를 하나의 저장소로 생각하여 대용량의 데이터를 하둡 네트워크에 연결된 기기에 분산으로 저장하기 위한 분산 확장 파일 시스템이다. 블록 구조의 파일 시스템으로서, 파일을 특정 사이즈의 블록으로 나눈 뒤 클러스터 서버에 저장하게 된다. 하둡 분산 파일 시스템에는 마스터-슬레이브 구조의 마스터 역할을 하는 네임노드 서버 1대와 슬레이브 역할을 하는 여러 대의 데이터노드 서버로 구성된다. 데이터를 여러 서버에 복제하여 중복 저장함으로써 데이터 안정성 및 결함

내성(Fault-tolerance)을 얻을 수 있다. 맵리듀스 프레임워크는 하둡 분산 파일 시스템에 저장된 대용량의 데이터를 맵(Map)과 리듀스(Reduce) 과정으로 나누어 병렬로 처리하기 위한 오픈소스 분산 프로그래밍 모델이다. 맵이란 흩어져 있는 데이터를 <key, value>의 형태로 관련 있는 데이터들로 분류하는 과정이고, 리듀스는 맵에서 출력된 데이터에서 중복을 제거하고 원하는 데이터를 추출하는 작업이다. 이 맵리듀스 과정은 프로그래머가 직접 작성하여 메소드를 구성할 수 있다. 그림 1은 맵리듀스의 대표적인 예제로 일반 문자열 데이터에서 단어 별로 빈도수를 구하는 문제를 처리할 때의 과정과 단계별 설명이다.

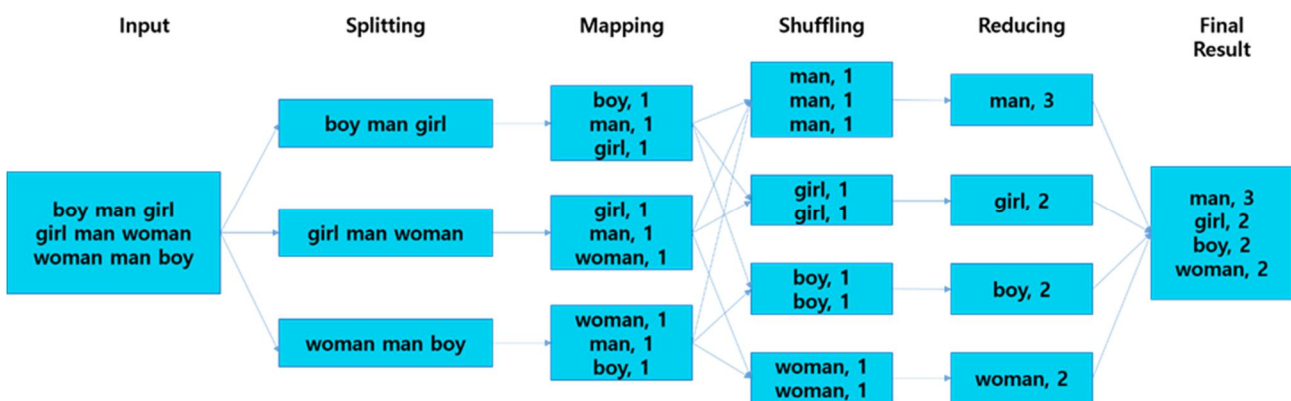


그림 1. 단어 별 빈도수 출력 맵리듀스 프로세스

- (1)Input 단계 : HDFS 내의 정해진 위치에 있는 데이터를 읽어들인다. 각 노드는 자신이 가지고 있는 블록의 데이터를 읽는다.
- (2)Splitting 단계 : 각 노드가 가지고 있는 문자열 데이터를 라인별로 나눈다.
- (3)Mapping 단계 : 라인별로 문자열을 입력받아서 문자열을 key로, 빈도수를 value로 하는 (key, value) 형태로 출력을 하게되는데, 여기서는 빈도수를 계산하진 않고 단순히

문자열과 value 값으로 1을 주어 예를 들어 <boy, 1>와 같은 형태로 다음 단계에 전달한다.

(4)Shuffling 단계 : Mapping 단계에서 받은 데이터를 같은 key를 가지는 데이터끼리 모아서 분류한다. 이 때까지는 <boy, 1>, <boy, 1>과 같은 형태로 빈도수 계산이 이뤄지지 않은 상태이다.

(5)Reducing 단계 : 각 key별로 value 값을 합산하여 빈도수를 구한다. 이 과정에서 <boy, 2>와 같은 형태가 된다.

(6)Final Result 단계 : 각 리듀스 메소드의 출력 데이터를 모아서 HDFS에 저장한다.

오늘날 많은 하둡 기반의 오픈소스 분산 처리 플랫폼이 개발되고 있고 실제 기업에서도 사용되고 있다. 이러한 하둡 기반 주요 빅데이터 오픈소스 플랫폼들을 하둡 에코시스템(Hadoop Ecosystem)이라 한다. 가장 많이 쓰이는 기술로서는 하둡에서도 사용되는 분산 자원 관리 툴인 Apache Zookeeper, 분산 컬럼 기반 데이터베이스 Apache HBase[4], 맵리듀스를 통한 분석을 지원하는 데이터 웨어하우스인 Apache Hive, 범용적 목적의 병렬분산 인메모리 기반의 고성능 클러스터링 플랫폼인 Apache Spark, 분산 메시징 시스템인 Apache Kafka 등이 있다. 한번 하둡 에코시스템을 구축하면 새로운 기술은 그 위에서 동작하여 쉽게 확장할 수 있기 때문에 최근 이런 시스템을 많이 사용하여 구축하고 있다[5]. 그림 2는 전반적인 하둡 에코시스템 구성도이며, 각각의 목적을 가지고 플랫폼들이 개발된 것을 알 수 있다.

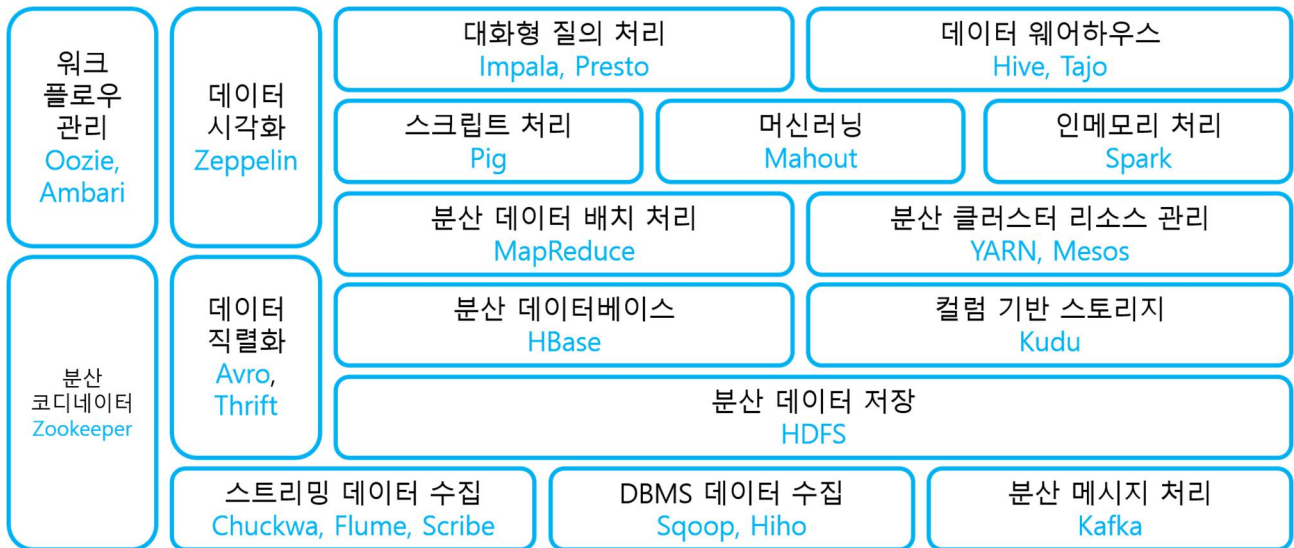


그림 2. 하둡 에코시스템 구성도

## 2.2 아파치 카프카(Apache Kafka)

아파치 카프카는 LinkedIn에서 개발된 분산 메시징 시스템으로 2011년에 오픈소스가 되었다[6]. 대용량의 실시간 로그처리에 특화된 아키텍처로 기존에 존재하던 메시징 시스템보다 우수하여 많이 사용되고 있다. 카프카는 기본적으로 발행-구독(publish-subscribe) 모델을 기반으로 동작하고 모델의 형태는 그림 3과 같으며, 크게 생산자(Producer), 소비자(Consumer), 브로커(Broker) 3가지로 구성된다. 생산자는 특정 토픽(topic)의 메시지를 생성하며 브로커에 전달하는 역할을 한다. 소비자는 생산자가 생성한 메시지 중에서 구독하는 토픽의 메시지를 가져오는 역할을 하며, 브로커는 토픽을 기준으로 메시지를 관리하는 역할을 한다.

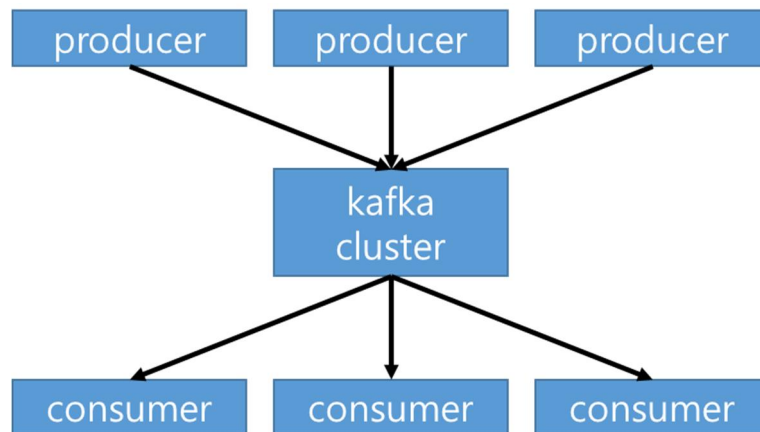


그림 3. Kafka의 기본 구성 요소

아래 그림 4는 카프카의 동작 과정이다. 카프카의 브로커는 확장성과 고가용성을 위해 클러스터로 구성되어 동작하도록 설계되었다. 클러스터 내에서 브로커에 대한 분산 처리 및 연결 관리는 Apache ZooKeeper가 담당한다.

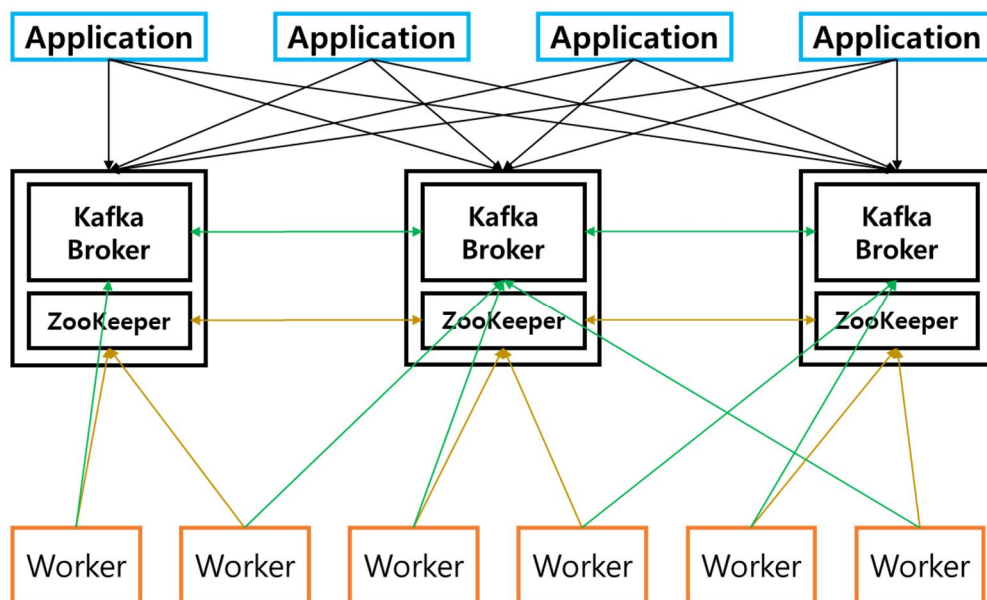


그림 4. 카프카의 동작 과정

## 2.3 아파치 기린(Apache Kylin)

아파치 기린은 2013년 중국 상하이의 eBay R&D에서 시작되었고, 2014년 10월에 KylinOLAP이라는 이름의 오픈소스로 공개되었다[7]. 이후 같은 해 11월에 아파치 소프트웨어 재단의 인큐베이터에 합류하였고, 2015년 12월에 최상위 프로젝트로 상승되었다. 지금은 eBay의 트랜잭션 OLAP에서 핵심이 되었다.

대규모 데이터에 대한 OLAP은 보통 두 가지 방식으로 접근한다. (1) 여러 질의 유형별로 해당되는 데이터 셋을 미리 만들어 놓는 데이터 큐브 방식, (2) 질의가 들어올 때 동적으로 계산하는 방식이 있다. Kylin은 하둡 에코시스템 기반의 OLAP 플랫폼으로서 대규모 데이터를 SQL 기반으로 다차원 분석한다는 특징이 있다. Kylin은 대규모 데이터를 처리하기 위해 데이터 큐브 방식을 채택하여 질의 결과를 빠르게 제공하기 위해 데이터를 미리 만들어 놓는 방식을 사용한다.

Kylin은 하둡 에코시스템 기반 플랫폼이기 때문에 동작하기 위해 필요한 플랫폼이 있다. 하둡의 분산 파일 시스템 및 매퍼듀스와 Apache Hive, Apache HBase, Apache Calcite가 필요하다. 각각의 역할로는 하둡 분산 파일 시스템에 큐브를 생성하는 과정에서 생성되는 중간 파일들을 저장하고, 매퍼듀스로 큐브를 생성하는 과정 중의 수치 값들을 집계한다. Apache Hive는 큐브를 생성하기 위한 입력 소스 역할과 사전 결합된 스타 스키마 형태를 가지고 있는 역할을 한다. Apache HBase는 데이터 큐브를 데이터베이스 형태로 저장하고 있고, 해당하는

질의의 결과를 반환한다. Apache Calcite는 데이터 관리 시스템 구축을 위한 프레임워크로서 SQL 파서를 내장하고 있어 질의 분석 및 최적화와 실행 계획을 생성하는 역할을 한다.

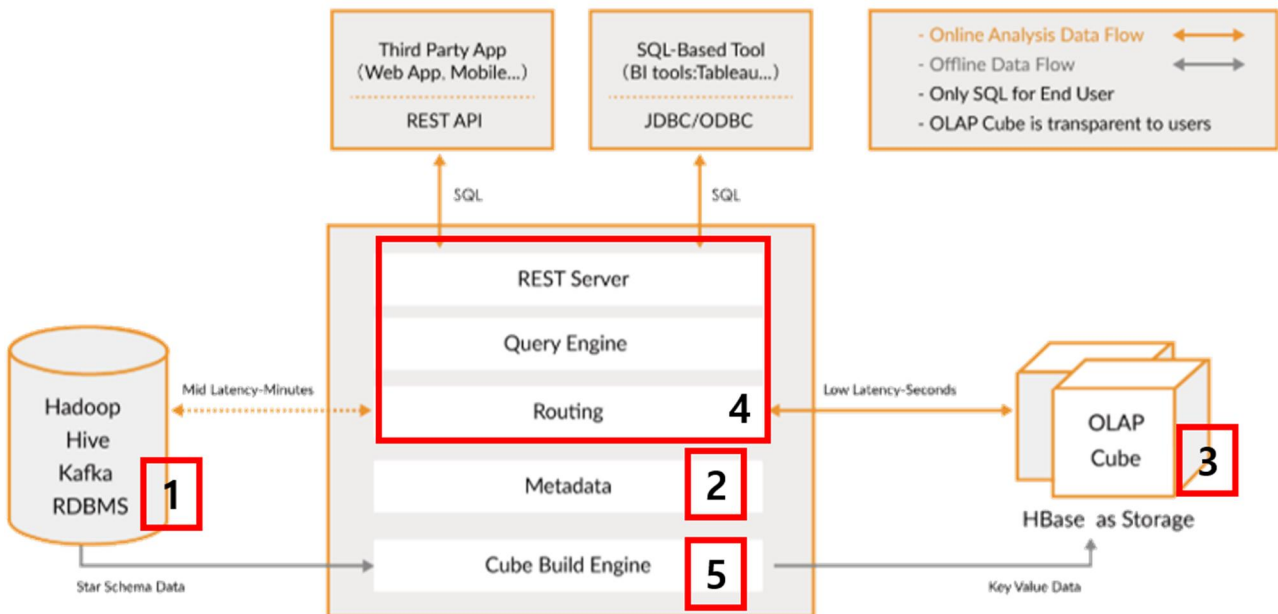


그림 5. Kylin의 아키텍처[7]

위의 그림 5는 Kylin의 아키텍처이다. Kylin은 위와 같이 5가지의 컴포넌트로 구성된다.

- (1) 데이터 저장소 : 원본 데이터가 주로 Hive에 스타 스키마 형태로 저장된다.
- (2) 메타데이터 저장소 : 큐브 디자인에 관한 메타 데이터, 스타 스키마의 정보 등 전체적인 메타 데이터가 저장된다. 저장되는 위치는 HBase 내에 kylin\_metadata라는 htable로 저장되고 HBase 내에서 htable을 스캔하여 확인할 수 있다.
- (3) 큐브 저장소 : 큐브 빌드 엔진인 MapReduce로 빌드한 OLAP 큐브가 큐브 저장소로 사용된 HBase에 데이터베이스 형태로 저장된다.



(4) 질의 엔진 : 사용자는 RESTful API, SQL 등으로 질의를 실행할 수 있다. 사용자의 질의를 분석해서 질의 결과를 HBase에 저장된 큐브에서 가져올 수 있는지, 가능하지 않다면 Hive의 원본 데이터에서 가져오는 것을 판단한다. 질의 결과를 큐브에서 가져오게 되면 low latency로 seconds 단위로 가져오고, 원본 데이터에서 가져오게 되면 MapReduce 작업이 필요하기 때문에 지연이 발생한다 mid latency로 minutes 단위의 시간이 필요하게 된다. 사용자를 위한 웹 GUI 애플리케이션을 실행하는 Tomcat, 질의 분석과 실행 계획을 생성하는 Calcite로 구성된다.

(5) 큐브 빌드 엔진 : 원본 데이터를 하둡의 MapReduce 작업을 통해 큐브로 빌드한다.

## 2.4 아파치 드루이드(Apache Druid)

아파치 드루이드는 아파치 소프트웨어 재단의 오픈소스 고성능 실시간 분석을 위한 분산 데이터 스토어(Datastore)이고, 대규모 데이터셋에서 slicing이나 dicing 등 빠른 OLAP 질의 및 실시간 분석을 위해 설계되었으며, 많은 양의 실시간 스트리밍 및 배치 데이터 수집, 빠른 질의 성능, 높은 가동시간이 중요한 케이스에 사용되고 있다[8]. 이러한 이유로 아직 인큐베이팅 단계이지만 Event 기반의 데이터에 잘 동작하기 때문에 eBay나 Yahoo, Airbnb, Netflix 등에서 채택하여 사용되고 있다. Druid를 클러스터 모드로 동작시키기 위해서는 여러 대의 서버를 각각의 특수 목적을 가지는 특정 노드를 실행시켜 동작하도록 해야 한다.

Druid의 특징으로는 (1) 컬럼 기반의 저장소 포맷이라는 것이다. 특정 질의 요청 시에 필요한 열만 로드하여 질의 속도가 향상된다. (2) 확장 가능한 분산 시스템이다. 수집에서

수백개의 서버까지의 클러스터에 배포가 가능하고 필요한 성능을 유지할 수 있다. (3) 실시간 처리가 가능하다. 실시간으로 수집된 데이터를 즉시 질의할 수 있다. (4) 웹 기반으로 Cloud-native하게 구현되었기 때문에 내결함성을 지닌 아키텍처이다. (5) 빠른 필터링을 위한 인덱스를 지원한다. (6) timestamp 기반으로 데이터가 파티셔닝되기 때문에 기존 전통 데이터베이스보다 빠르다. (7) 데이터를 수집하는 동안 특정 컬럼의 정해진 형태의 Roll-up을 지원하여 데이터를 요약한다. (8) SQL을 지원한다. 예전 버전에서는 JSON 기반의 질의 언어를 작성해야만 하는 불편함이 있었지만, 그 대안으로 이제는 Apache Calcite 기반의 파서와 플래너를 제공하는 Druid SQL이 추가되어 SQL 질의가 가능하다.

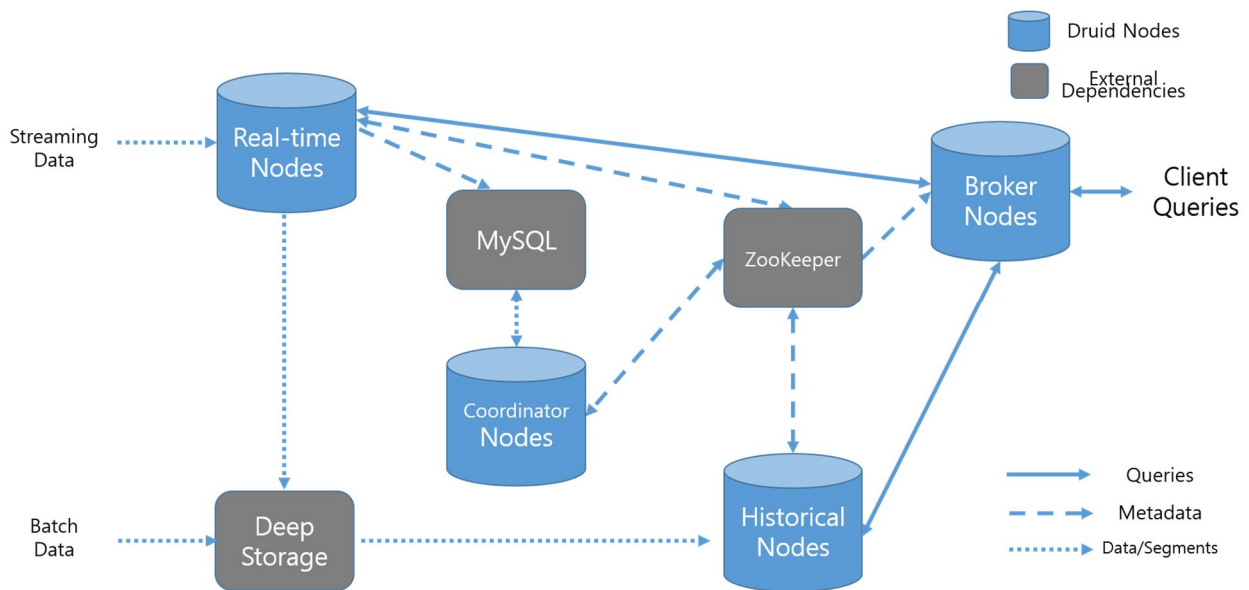


그림 6. Apache Druid의 아키텍처

위 그림 6은 Druid의 아키텍처이다. 실시간 스트리밍 데이터와 배치 데이터를 처리하기 위한 데이터 수집 방법이 존재하며, 서버는 보통 크게 3가지로 분류한다. 먼저 데이터의 수집

및 가용성을 관리하는 Master 서버, 사용자와 클라이언트 응용 프로그램이 상호작용하여 질의를 데이터 서버나 다른 Query 서버로 라우팅하는 엔드 포인트를 제공하는 Query 서버, 그리고 데이터 수집 작업을 실행하고 질의가 가능한 데이터를 저장하는 Data 서버가 있다. 세부적으로는 Master 서버는 Coordinator 노드와 Overload로 나누어지고 Query 서버는 Broker 노드와 Router로 나누어지고, 그리고 Data 서버는 Historical 노드와 Middle Manager로 나누어진다. 각 노드의 자세한 역할에 대해 살펴보면 다음과 같다.

- Coordinator Node : Historical Node에 보관하는 데이터 세그먼트 관리 및 데이터 세그먼트의 이동, 삭제, 생성을 관리한다.
- Overload : Middle Manager에게 일을 분배하고 관리한다.
- Broker Node : 사용자 클라이언트의 질의를 받아 Data 서버로 전달하고, 서브쿼리들의 결과를 모아 병합하여 사용자에게 결과를 제공하는 노드이며 어떤 데이터 세그먼트가 정상 및 비정상인지, 어느 노드에 저장되어 있는지와 같은 메타 데이터를 가지고 있다.
- Router : Broker와 Overload, Coordinator 앞단에서 통합 API를 제공하는 프로세스로서 데이터 소스, 세그먼트 현황, 작업 현황, 서버 현황 등을 보여주는 관리 유저 인터페이스를 실행한다.

- Historical Node : Real-time Node로부터 받은 스트리밍 데이터 및 배치 데이터에 대한 저장소이고 Deep storage에서 데이터 세그먼트를 다운받아 저장한다. 그리고, 각 Historical Node 파일 시스템에 저장된 데이터 세그먼트에 대한 질의를 제공한다.
- Middle Manager : 클러스터로의 새로운 데이터의 수집을 관리한다. 외부 데이터 소스를 읽어서 Druid의 세그먼트를 발행한다.
- Real-time Node : 데이터를 수집함과 동시에 indexing을 하기 때문에 수집 중인 데이터에 대해 질의 요청시에 서비스를 제공한다. 이후 정해진 시간 단위로 데이터 세그먼트를 생성하여 저장하기 위해 최종으로 Historical Node에 전달하게 된다.

위와 같은 노드를 이용한 클러스터를 구축하기 위해서는 3가지 외부 의존성이 존재하는데, 클러스터 조율로는 Apache ZooKeeper가 사용되고 메타 데이터 저장소로는 Apache MySQL이 사용되며 딥 스토리지(Deep Storage)로는 하둡의 분산 파일 시스템이 사용된다. 자세한 역할은 다음과 같다.

- 딥 스토리지 : 모든 Druid 서버가 접근하는 공유 파일 스토리지로서 일반적으로 Amazon S3나 하둡의 HDFS가 사용된다. 데이터 백업이나 Druid 프로세스 간의 데이터를 전송하는 방법으로 사용된다. 질의에 응답하는 것은 딥 스토리지를 거치지 않고 Historical 프로세스만 사용하기 때문에 조금 더 빠른 질의 응답속도를 제공할 수 있다.

- 메타데이터 저장소 : 세그먼트 가용성 정보 및 작업 정보와 같은 여러 공유 시스템 관련 메타 데이터를 가진다. 기본 설치로는 Apache Derby가 실행되나 클러스터로 구축 시에는 적합하지 않고 MySQL이나 PostgreSQL과 같은 일반 RDB가 적합하다. 메타데이터는 잃어버릴 시 복원이 힘들기 때문에 고가용성 환경이 요구된다.
- 분산 코디네이터(Distributed Coordination) : Druid 클러스터를 대규모에서도 운영할 수 있게 하기 위해 필요하고 Apache ZooKeeper를 사용함. 내부의 새로운 서비스 노드 발견 및 조정, 리더 선출 등의 역할을 한다.

## 2.5 아파치 슈퍼셋(Apache Superset)

아파치 슈퍼셋은 웹 기반의 데이터 시각화 도구로서 Airbnb에서 오픈소스로 공개하여 현재 아파치 재단의 incubating 단계에 있는 BI 도구이다[9]. 데이터 시각화를 위한 많은 종류의 차트를 지원하고, 대화형 대시보드를 생성하여 모니터링 시스템으로 사용이 가능하다. 그리고 SQLAlchemy를 통하여 RDBMS와 SQL을 통해 연동할 수 있고, Drill-Down이나 slice, dice와 같은 OLAP 질의도 가능하다. 여러가지 일반 RDBMS와도 결합하여 시각화가 가능하고 특히 본 논문에서 활용한 Druid와의 연동성으로 실시간 상태를 유지하면서 대용량의 실시간 데이터를 OLAP 질의할 수 있는 등의 이유 때문에 시각화 도구로 선정하였다.

Superset은 WEB GUI 메뉴를 통해 CSV 데이터나 RDBMS의 데이터를 가져와서 시각화할 수 있고, 특히 본 논문에서 필요한 Druid 클러스터와 연동할 수 있고 Druid의 데이터 소스를 불러올 수 있는 기능을 제공한다. 그 이후에 원하는 형태로 데이터를 탐색하고 시각화할 수

있으며 모니터링 하고자 하는 데이터들을 실시간 상태를 유지하는 대시보드를 작성할 수 있다.

아래 그림들은 Superset을 이용한 기본 제공 파일인 ‘Wikipedia’ 데이터의 시각화 예제이다.

user	count
TuanminhBot	3.39k
ThitvongkhoiAWB	2.04k
TuanUt-Bot	1.79k
TuanUt	984
Ximik1991Bot	982
Cheersl-bot	721
Anligng-bot	379
TuHan-Bot	354
Krdbot	301
GHA-WDAS	279

그림 7. 유저 별 row의 수 집계에 대한 테이블 형태 시각화

위의 그림 7 Superset 예제는 테이블 형태로 유저의 목록과 유저별 row의 수를 카운트한 것을 시각화한 것이다.

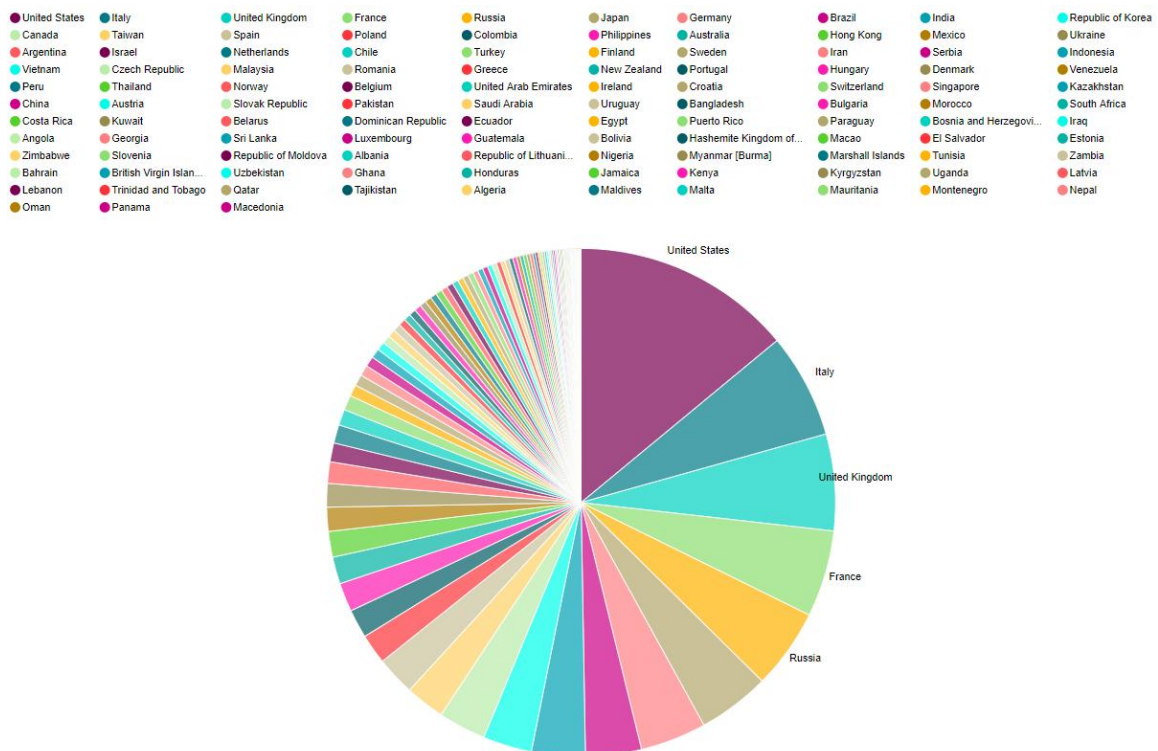


그림 8. 국가별 row의 수 집계에 대한 pie chart 시각화

위의 그림 8 Supsert 예제는 pie chart의 형태로 국가별 row의 수를 집계한 것이고 Druid의 NoSQL 성격 때문에 null 값이 허용되어 필터링하여 제거하였다.

### 3 관련 연구

본 장에서는 OLAP과 데이터 큐브의 개념에 대해 설명한다. 그리고 데이터 큐브에 관한 연구에 대해서 살펴본다. 제3.1절에서는 OLAP에 대해 설명하고 제3.2절에서는 데이터 큐브에 대하여 설명한다.

#### 3.1 OLAP(One-Line Analytical Processing)

OLAP은 데이터웨어하우스 활용 수단으로서 기업에서 효율적인 의사결정을 할 수 있게 도와주는 방법을 연구하는 BI(Business Intelligence)의 한 분야로 최종 사용자가 다차원 정보에 접근해 대화식(Interactive)으로 분석하고 의사결정에 활용하는 과정을 의미한다[10][11][12]. OLAP의 주요 특징은 분석을 위해 활용되는 정보의 형태가 다차원(Multi-Dimensional)이라는 것이다. 그리고 최종 사용자는 정보에 대해 잘 아는 담당자나 보고서 등의 매개체 없이 온라인에서 직접 데이터에 접근한다는 점이고, 대화식으로 정보를 분석할 때 시스템은 신속한 질의 결과를 제시해야 한다. 다차원 분석을 위한 OLAP의 주요 기능은 다음과 같다.

- Drill-Down/Roll-up : 가장 요약된 수준으로부터 가장 상세한 수준까지 단계적으로 분석에 필요한 요약 수준을 바꾸는 기능이다. 이 기능을 활용하여 계층별 분석을 함으로써 의미있는 분석을 도출할 수 있다. 일반적인 예로 지역 데이터의 경우에



대한민국-시/도-구/군-읍/면/동 으로 데이터 정렬을 바꿀 수 있다면 대한민국에  
가까워질수록 Roll-up이고 읍/면/동별에 가까워질수록 Drill-Down 연산이다.

- Pivot : Rotate라고도 하며, 사용자에게 최종적으로 보여지는 결과 화면에서 보여지는  
축인 차원(Dimension)을 바꾸는 기능이다. 이 기능을 통해 분석가는 여러 패턴으로  
데이터를 분석할 수 있다.
- Slice : 다차원 배열에서 한 차원 이상의 멤버를 선택하여 한 값을 선택하여 나타나도록  
하는 그 부분 집합(데이터 단편화)이다. 예를 들어, 제품, 기간, 지역 3차원의  
데이터에서 분석하고자 하는 값이 판매량일 때 특정 제품의 전 지역, 전 기간의  
판매량을 보는 것을 말한다.
- Dice : 사용자가 Slice한 특정한 항목에 대해 Pivot이나 Drill-Down 등의 기능을  
이용해서 대화식으로 화면을 바꿔가며 분석하는 프로세스이다. 다양한 상황에 맞춰  
사용자는 다차원 데이터에서 Slice와 Dice 기능을 이용해 필요한 정보를 분석할 수  
있다.

OLAP의 종류는 MOLAP(Multidimensional OLAP : 다차원 OLAP), ROLAP(Relational  
OLAP : 관계형 OLAP), DOLAP(Desktop OLAP), HOLAP(Hybrid OLAP) 이 4가지로 구분할 수  
있고, 크게 보면 ROLAP과 MOLAP으로 나뉘어지는데 이 둘의 차이는 DBMS라고 볼 수 있다.  
다차원 데이터베이스(MDB)를 기반으로 하면 MOLAP이고, 관계형 데이터베이스(RDB)를  
기반으로 하면 ROLAP이다. MOLAP의 경우 기존 원본 데이터를 큐브 형태의 다차원 데이터로

변환하여 미리 계산하여 저장해두고 해당하는 질의 요청시에 저장된 값을 반환하게 된다. 데이터가 정형화되어 있으므로 사용자 질의에 일정하게 빠른 속도를 보장하는 것이 장점이다. 그러나 모든 데이터를 미리 계산해두려면 비용이 많이 들기 때문에 대용량 데이터 처리시 초기 계산 비용이 많이 든다. 반면 ROLAP의 경우는 기존 사용하던 RDBMS에 바로 OLAP 도구를 연결해서 사용할 수 있고, 확장성 있는 데이터 구조와 비정형화된 질의에도 응답이 가능하다는 장점이 있다. 하지만, 다차원 모델링을 위해서 스타 스키마와 같은 별도의 스키마가 필요하다.

데이터 분석을 제공하기 위해 의사결정지원 시스템에서 OLAP 기술이 점점 더 많이 사용되고 필요함에 따라 많은 OLAP 도구들이 생겨나고 있다. 대표적인 도구로는 위에서 설명한 아파치 재단의 Kylin이나 Druid가 있고 Oracle의 OBIEE나 IBM의 Cognos 등 많은 BI 도구가 생겨나고 연구되고 있다. Kylin을 이용하여 빅데이터 시스템 구축하는 것을 다룬 연구가 있었고[13], ROLAP과 MOLAP을 모두 다루는 시스템 개발 및 성능평가를 한 연구가 있었다[14]. Druid를 새로운 인덱스 체계를 사용하여 분석 질의의 응답 시간을 향상시키는 연구가 있었고[15], Druid의 여러가지 변수를 조절하는 것을 다루고 다른 Hadoop 기반 시스템들과 성능 비교를 실험한 연구가 있었다[16].

### 3.2 데이터 큐브

데이터 큐브는 대용량의 데이터를 다차원으로 분석 및 집계한 결과를 사용자에게 빠르게 제공하기 위한 자료 구조이다[17]. 데이터 큐브는 분석 관점인 차원(Dimension) 속성과 여러

차원과 관련하여 원하는 집계 연산을 적용하여 계산된 측정값(Measure)으로 구성된다. 측정값은 수치값들의 최소(MIN), 최대(MAX), 평균(AVG), 합계(SUM), 집계된 개수(COUNT) 등과 같은 값으로 표현한다. 일반적으로 데이터 큐브는 이러한 방식에 따라 계산 가능한 모든 집계값을 모두 저장하는 자료구조이다. 이후에 사용자가 분석하고자 하는 관점에 따라 OLAP 연산을 적용하여 원하는 결과를 빠르게 얻어냄으로써 의사결정에 도움을 줄 수 있다.

데이터 큐브는 여러 분야에 걸쳐서 다차원 데이터 분석을 위해 활용되고 있는 추세이다. 하지만 데이터 큐브의 계산은 보통 대용량의 데이터를 다루기 때문에 계산에 사용되는 집계 연산에 높은 비용이 요구된다. 이 문제를 효율적으로 해결하기 위해서 많은 방법들이 연구되었다. 먼저 데이터의 큐브의 개념과 큐브에 적용가능한 연산자, SQL을 통한 Roll-up 등을 소개한 연구가 있었고[17], 아주 큰 차원의 데이터셋에서 효율적으로 OLAP 연산을 수행하기 위한 데이터 큐브에 관한 연구가 있었다[18]. 그리고 데이터 큐브 전체를 집계하는 것이 아니라 필요한 일부 차원의 조합만을 집계하는 빙산 큐브가 있었다[19].

일반적인 데이터 큐브 이외에도 여러가지 연구가 있었는데, 분산 병렬 처리 방법을 이용한 연구들이 있었다. 데이터 큐브를 구축하거나 계산하는 비용이 컸기 때문에 병렬로 데이터 큐브를 구축하는 연구가 있었고[20], 확장 가능한 큐브에 대한 연구도 있었다[21]. 고성능 서버를 대신하여 여러 대의 저렴한 컴퓨터로 분산 병렬 처리가 가능한 맵리듀스를 이용하여 데이터 큐브를 분산 병렬 처리하는 연구가 있었고[22], 맵리듀스를 이용하여 대규모 데이터셋에서 큐브를 계산하는 효율적인 알고리즘을 제시하는 연구가 있었다[23]. 그리고

스트림 데이터에 대해 온라인 다차원 분석 처리를 수행하는 Stream cube를 제안한 연구가 있었다[24].

## 4 전력 데이터 및 데이터 전처리

이 장에서는 본 논문에서 구성하는 시스템에 사용된 데이터에 대한 설명과 다차원 모델 생성에 관해 살펴보고자 한다. 제4.1절에서는 전력 데이터에 대해 설명하고, 제4.2절에서는 모델 구성에 필요한 데이터 가공에 대해 살펴보고, 마지막으로 제4.3절에서는 전력 데이터를 다루는 연구에 대해 살펴본다.

### 4.1 전력 데이터

오랫동안 많은 국가는 화석연료를 통해 전력을 공급받아왔지만, 오늘날 전 세계적으로 기후의 변화와 에너지 고갈에 관한 문제가 제기되면서 미래의 전력 공급에 대한 불안정 이슈가 나오기 시작하였다. 이를 해결하기 위해 기존 전력망에 IT 기술을 활용하여 전력 생산량 및 전력 소비량을 파악하여 전력 공급을 함으로써 효율적으로 문제를 해결할 수 있는 스마트 그리드 시스템이 개발되기 시작하였다[25]. 그러한 기술 중에 스마트그리드 시스템을 더 효율적으로 사용하도록 하는 핵심 기술로 AMI(Advanced Metering Infrastructure) 라고 하는 원격검침 인프라가 각광받기 시작했는데 본 논문에서 사용할 전력 데이터가 이 AMI 데이터이다[26].

AMI는 스마트 미터기에서 측정한 데이터를 원격 검침기로 측정하여 전력 사용 분석을 자동화 하는 기술로서 스마트 미터기가 각 집의 전력 사용량을 자동으로 검침하여 검침값을

통신망으로 전달하는 형태이다. 이 형태로 얻은 데이터를 활용하여 전력회사들은 소비자의 전력 사용량에 해당하는 전기요금을 부과하게 된다. 오늘날에는 이러한 데이터를 활용하여 고객별 전기 사용 패턴을 파악하여 적당량의 전력을 공급하여 전기요금 절약 및 전력의 낭비를 예방하려고 하는 연구들이 진행되고 실제 분야에 적용되고 있다. 머신러닝이 발전함에 따라 머신러닝 기법에 적용하여 정확도를 높이려는 시도도 있다. AMI의 또 다른 장점으로는 AMI가 보급되기 전에는 사람이 직접 돌아다니면서 검침해야 했고, 사람이 직접 검침하기 때문에 발생하는 오차 등의 불편함이 있었는데 AMI를 통해 이를 해소하고 정확한 빅데이터를 획득하여 효율적인 전력 생산 관리가 이루어질 것으로 예상하고 있다.

## 4.2 데이터 가공 및 전처리

데이터는 한전에서 연구를 위해 제공한 AMI-LP(Load Profile) 데이터의 일부를 사용하였다. 데이터의 총 크기는 저압 LP 데이터 39GB, 고압 LP 데이터 12.8GB였으며, 저압 LP 데이터에는 총 26개의 컬럼이 있고, 고압 LP 데이터에는 총 19개의 컬럼이 있다. 모델 생성 시 필요에 따라 가공 및 전처리하여 데이터의 양을 줄여서 사용하였다.

### 4.2.1 Batch 형태의 입력을 위한 전처리

초기에 AMI 데이터를 이용하여 배치 형태의 데이터 큐브를 구축하고자 했다. 데이터를 먼저 간단히 분석하고자 (1) 데이터의 컬럼 중 필요한 것을 제외한 나머지 컬럼 제거 (2) 이상치로 판단되는 데이터 제거 작업을 진행하였다. (1)에서 저압과 고압 LP 데이터 모두 여러 종류의 컬럼 중 미터기\_ID, DCU\_ID, 전력 사용량 컬럼만을 제외하고 제거하였고 컬럼 중에

날짜 관련 컬럼 2개를 조합하여 Kylin에 적재하기 위하여 'yyyy-MM-dd' 형태의 컬럼을 1개 생성하여 총 4개의 컬럼으로 데이터를 구성하였다. (2)에서는 위의 방법으로 확인한 이상값을 필터링하였다. 그리고 미터기 별 정상적인 데이터 측정 cycletime이 정해져 있는데 정해진 시간을 벗어난 값도 제거하였다. 이후에 분석을 위하여 Python의 데이터 분석 및 처리를 위한 라이브러리인 Pandas를 활용하였다. CSV 형태의 데이터를 Pandas로 읽어 데이터 프레임 형태로 만든 뒤 .describe() 메소드를 이용하여 데이터의 개수, 평균값, 표준 편차, 최솟값, 25%, 50%, 75%, 최댓값에 대한 정보를 확인하였다. 특히 여기서 최솟값, 최댓값 수치를 통해 이상값의 존재를 파악하고 필터링하였다.

#### 4.2.2 Stream 형태의 입력을 위한 전처리

Stream 형태로 데이터를 수집하는 방법의 경우에는 데이터 원본 그대로 수집된다고 가정하고 배치 형태의 경우와 다르게 데이터를 필터링이나 추출하지 않고 모두 저장되도록 하였다. Stream 형태로 데이터를 저장하기 위해서는 timestamp 컬럼이 필수적인데 Druid에서 사용 가능한 timestamp의 형태로 만들기 위해서 날짜 관련 2개의 컬럼을 합쳐 'yyyy-MM-dd HH:mm:ss'의 형태의 컬럼 1개를 만들었다. 그리고 Druid 시스템의 오류로 csv 파일의 Column header를 인식하는데 문제가 있으므로 모두 제거해주었고 스트림으로 데이터를 읽을 때 컬럼의 이름을 명시하는 방식으로 진행하였다. 마지막으로 Druid의 timestamp 컬럼이 AMI 데이터에는 존재하는 24시 정각의 데이터를 인식하지 못하여 다음날 00시로 전환해주었다.

### 4.3 전력 관련 연구

전력 관련 데이터를 다룬 연구로는 주로 전력 사용 패턴을 분석하는 연구가 많았고, 스마트 그리드 기술이 발전함에 따라 AMI 기술이 주목을 받아서 그 데이터를 이용하는 연구들이 늘고 있었다. AMI 데이터를 군집화하여 패턴을 분석하고 AMI 데이터에서 발생할 수 있는 오류와 보정 방법에 대해 다룬 연구[27]가 있었고, 대학 캠퍼스 내의 건물을 대상으로 실시간 전력 사용 데이터를 수집하고 연구에 활용하여 전력 사용량의 감소 효과 및 사용자의 만족도 향상을 위한 목적으로 서비스 개발을 하고 있는 연구[28]가 있었다. 전력 수요량을 예측하는 연구도 많았는데 시계열 데이터의 장, 단기적 특성을 반영하는 방법론을 제시하는 연구[29]가 최근에 있었다.



## 5 Apache Kylin을 이용한 전력 데이터의 다차원 큐브

이 장에서는 Apache Kylin을 이용하여 배치 전력 데이터의 다차원 큐브 생성에 관한 연구를 살펴본다. 제5.1절에서는 전력 데이터 다차원 분석 시스템 구성 및 동작에 대해서 살펴보고, 제5.2절에서는 큐브 구축을 위한 개념인 팩트 테이블과 스타 스키마에 대해서 살펴본다. 마지막으로 제5.3절에서는 Kylin을 이용한 Batch 형태의 큐브 생성 과정에 대해서 살펴본다.

### 5.1 전력 데이터 다차원 분석 시스템 구성 및 동작

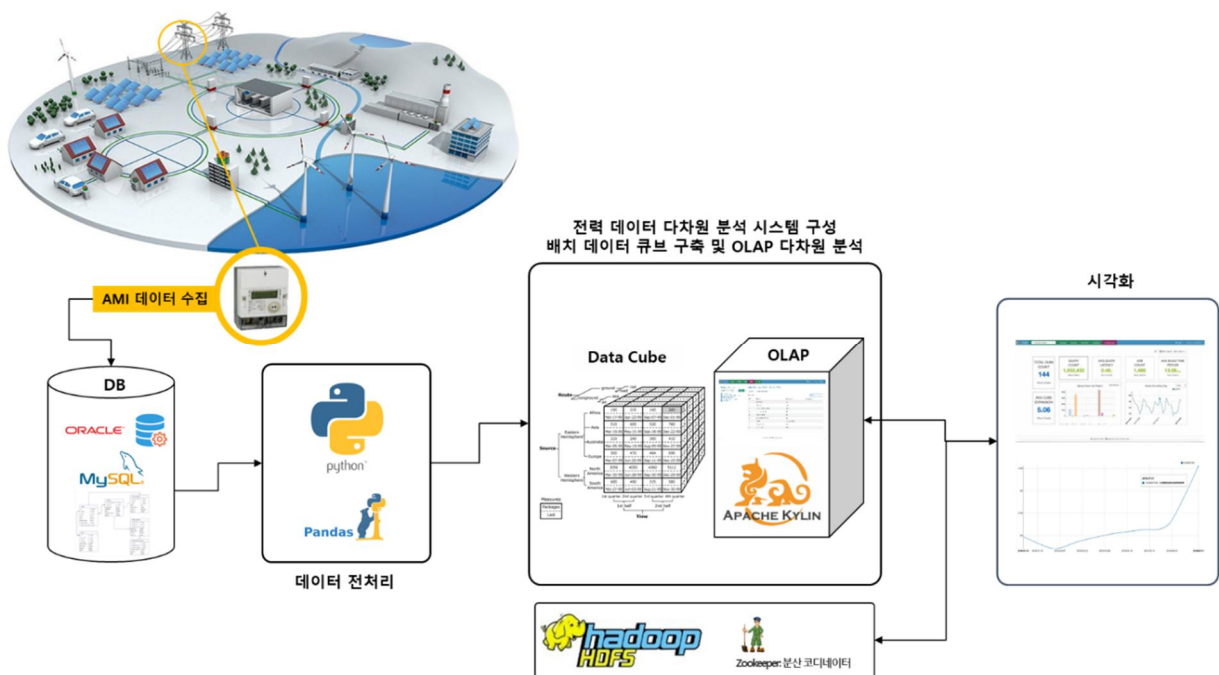


그림 9. 전력 데이터 다차원 분석 시스템 구성

위 그림 9는 Kylin을 활용하여 전력 데이터 다차원 분석 시스템을 구성한 것이다. AMI 미터기 혹은 DCU에서 모인 데이터들이 저장된 데이터베이스에서 데이터를 수집하여 데이터

큐브를 생성하기 위한 전처리 과정을 거친다. 이 과정에서 Python의 데이터 분석 및 처리를 위한 라이브러리인 Pandas를 사용하였다. 이후 전처리된 데이터를 여러 하둡 에코시스템이 베이스된 Kylin을 이용하여 데이터 큐브를 생성하였다. Kylin의 기능을 이용하여 데이터가 추가되면 점진적으로 업데이트 되도록 하여 데이터 추가시 큐브에 반영할 수 있도록 하였다. 이후에 SQL을 통해 다차원 분석을 하고 시각화하여 결과를 확인해볼 수 있다.

## 5.2 팩트 테이블과 스타 스키마

### 5.2.1 팩트 테이블

팩트 테이블은 어떤 사실에 관한 테이블이며 분석의 시작점이다. 예를 들어 미터기의 종류별, 날짜별 전력사용량 분석 같은 경우 팩트 테이블에서 시작된다. 팩트 테이블의 칼럼은 디멘션(Dimension)과 지표(Measure) 2가지 타입으로 구성된다. 디멘션은 내가 보고자 하는 축이다. 위의 예시에서 미터기의 종류, 특정 미터기, 날짜는 모두 디멘션에 해당한다. 일반적으로 팩트 테이블에서 디멘션 칼럼은 값을 가지고 있지 않고 상세 차원 테이블을 가리키는 외래키를 가지고 있다. Measure는 전력 사용량, 역률과 같은 디멘션의 지표 값이다. 일반 SQL에서 Aggregation 함수(COUNT, SUM 등)의 개념과 같다.

### 5.2.2 스타 스키마(Star Schema)

조인 스키마(Join Schema)라고도 불리는 스타 스키마는 데이터 웨어하우스 스키마 중 가장 단순한 종류의 스키마로서, 한 개의 팩트 테이블과 팩트 테이블의 특정 요소에 대한 정보를 가지고 있는 여러 개의 '상세 차원 테이블' 혹은 'Look-up table'이라고 불리는 것으로

구성된다. 이름의 유래는 팩트 테이블과 상세 차원 테이블이 연결된 스키마 다이어그램이 마치 별 모양과 같아서 스타 스키마라고 불리게 됐다. 팩트 테이블은 위에서 설명한 것과 같이 상세 차원 테이블을 참조하는 외래키를 가지고 있고, 이 외래키와 상세 차원 테이블의 기본키와 조인하여 상세 정보를 최종 분석 결과에 포함시킨다. 그러나 상세 차원 테이블끼리는 조인할 수 없다. 아래 그림 10은 스타 스키마의 예시로 많이 쓰이는 간단한 판매 예제이다. 중앙에는 판매 정보에 관한 팩트 테이블이 있는데, 이 테이블에는 3개의 상세 차원 테이블의 외래키와 판매대수에 대한 Measure 값을 가지고 있다. Dim\_Date 테이블은 날짜에 관한 상세 정보를 가진 상세 차원 테이블이고, Dim\_Store는 판매점에 관한 상세 차원 테이블, Dim\_Product 테이블은 상품에 관한 상세 차원 테이블이다. 이 데이터를 SQL을 이용하여 분석하게 되는데, 팩트 테이블로 시작하여 상세 차원 테이블로 조인을 하여 분석 결과를 가져오게 된다.

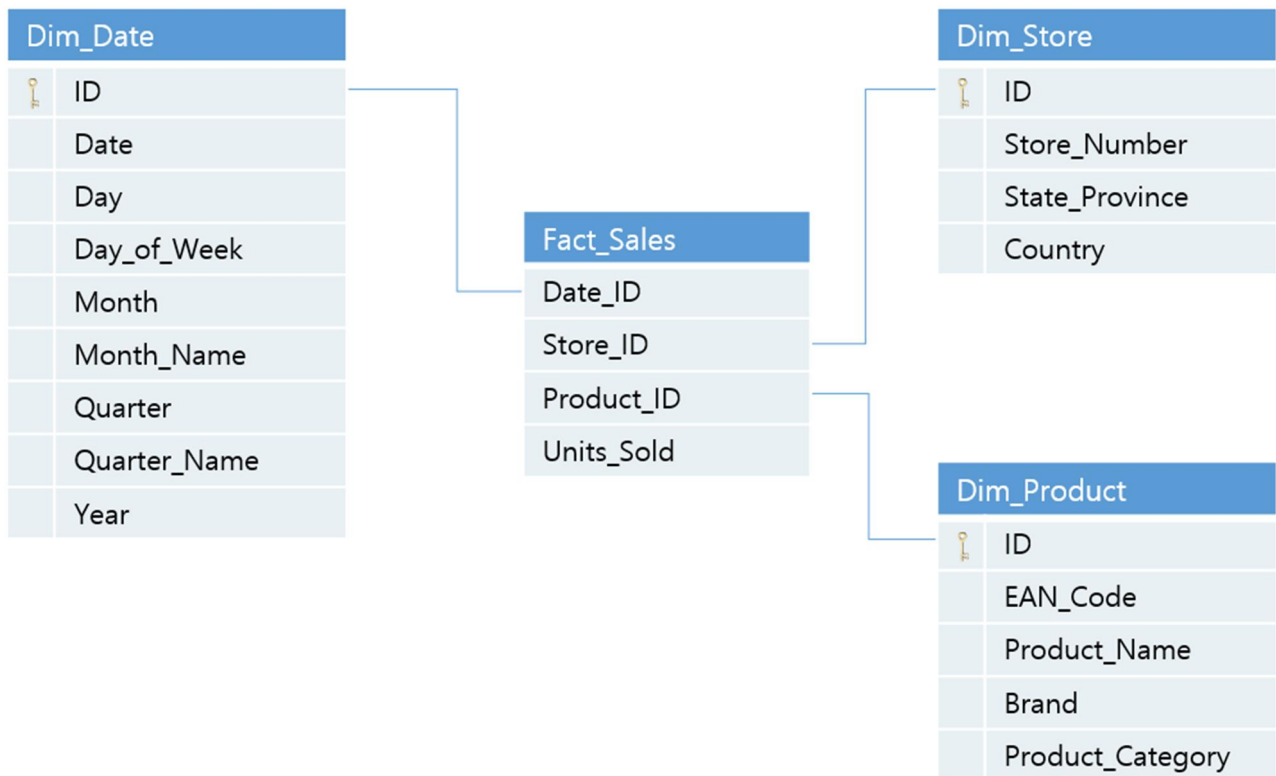


그림 10. 스타 스키마 예시

## 5.3 Kylin을 이용한 Batch 형태의 큐브 생성

본 장에서는 Kylin을 이용하여 배치 데이터 형태의 큐브를 빌드하는 과정을 설명한다.

### 5.3.1 가공한 AMI 데이터를 Hive에 테이블 형태로 저장

먼저 로컬에 보관하고 있는 가공된 데이터 csv파일을 분산 데이터 웨어하우스인 Hive에 적재하기 위하여 존재하는 컬럼에 맞추어 table을 생성한다. 이 테이블이 큐브를 만들기 위한 소스 테이블이 된다. 테이블을 생성한 이후에 Hive CLI 환경에서 LOAD DATA 명령어를 이용하여 로컬에 있는 데이터를 Hive 소스 테이블에 적재한다(데이터가 로컬 파일 시스템에

있다고 가정한다). 아래 표는 본 논문에서 구축한 데이터 테이블의 형태이다. 위에서 설명한 바와 같이 가공하였으며, 구축 실험을 위하여 적은 양의 컬럼을 사용하였다.

**표 1. 팩트 테이블이 될 AMI 데이터 테이블**

컬럼명	데이터 타입
Meter_ID	String
Date	String
DCU_ID	String
Power	Double

**표 2. Date 상세 차원 테이블이 될 날짜 테이블**

컬럼명	데이터 타입
Day_DT	String
Year_DT	String
Month_DT	String
By10_DT	String
By5_DT	String

**표 3. Code2Type 상세 차원 테이블이 될 미터기별 업종 테이블**

컬럼명	데이터 타입
Meter_ID	String
Business_type	Int

### 5.3.2 Kylin에서 Hive 데이터와 동기화

Kylin Web UI에서 Hive에 저장된 데이터를 불러올 수 있다. Data Source 탭에서 Load Table 기능에서 Hive의 Table 이름을 동일하게 적어 동기화시킬 수 있다. 이후 Tables 탭에서 동기화시킨 테이블이 추가된 것을 확인할 수 있고 테이블의 내용을 볼 수 있다. 백그라운드에서 MapReduce 작업을 통해 동기화된 테이블의 대략적인 카디널리티를 계산하여 테이블 정보에 표시한다.

### 5.3.3 Kylin에서 데이터 모델 생성

Kylin Web UI에서 데이터 큐브를 제작하기 전 만들고자 하는 데이터 모델을 설정할 수 있다. 이것이 스타 스키마를 의미하며 다른 큐브에서도 재사용 가능하다. 구축하고자 하는 스타스키마의 Fact Table을 정의하고 큐브에서 사용할 Dimension과 Measure를 선택할 수 있고 상세 차원 테이블인 Lookup Table을 추가 및 조인 유형을 선택하여 구성할 수 있다. 추가 세팅 값으로는 데이터의 분할 기준 컬럼을 원하는 경우 일별/시간별로 지정할 수 있고 필터 조건으로 제외하고 싶은 데이터를 정할 수 있다.

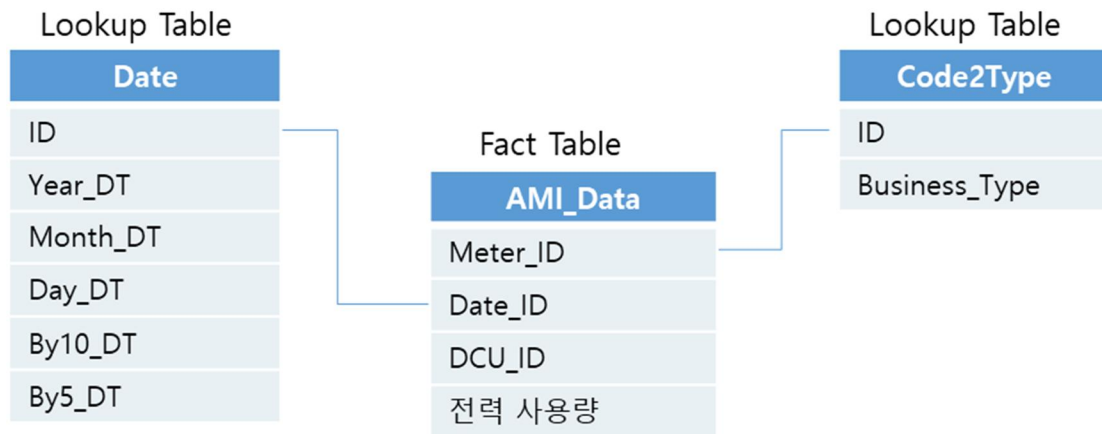


그림 11. 스타 스키마 구조

팩트 테이블은 Date 상세 차원 테이블과 Code2Type 상세 차원 테이블과 조인하기 위한 외래키 ID 값을 가지고 전력 사용량 수치값을 가진다. Date 테이블은 날짜 데이터로서 데이터를 년, 월, 일 혹은 10일단위, 5일단위로 검색하기 위한 상세 차원 테이블이다. Code2Type 테이블은 미터기별로 어떤 종류의 전기에 사용되는지에 대한 정보를 가지고 있는 상세 차원 테이블이다.

#### 5.3.4 Kylin에서 데이터 큐브 생성

데이터 모델을 생성하고 나면 큐브를 생성할 수 있는데, 스타 스키마를 토대로 빌드되고 한번에 큐브가 빌드되는 것이 아니라 먼저 N차원의 큐보이드(cuboid)를 생성한다. 큐보이드는 큐브의 한 축을 말하는데, N차원의 큐보이드에서 N-1차원의 큐보이드를 유도하는 방법으로 전체 큐브를 빌드한다. 먼저 큐브의 이름과 큐브에 사용될 Dimension과 Measure 값을 설정하게 되는데, 사용 가능한 Measure 값으로는 SUM, MAX, MIN, COUNT, COUNT\_DISTINCT, TOP\_N가 있다. 이들 중에 COUNT\_DISTINCT는 매우 무거운 유형이라

Measure로 추가할 시 빌드 및 질의 속도가 많이 느려져 제외하였다. 마지막으로 Refresh 세팅에서 incremental(증분) 큐브 빌드를 설계할 수 있다. 세그먼트의 크기를 조절하여 작은 큐브의 데이터에서 중간/큰 세그먼트의 크기로 자동 병합시켜 큰 큐브를 생성하도록 할 수 있다. 또, 데이터의 보존 기간을 설정하여 데이터가 과거 몇 일간의 데이터만 유지하도록 설정할 수 있고, 큐브의 시작 날짜를 설정할 수 있다.

### 5.3.5 기타 설정 값 추가

추가적으로 큐브 빌드를 최적화시킬 수 있는 설정 세팅이 있다. Aggregation 그룹을 설정하여 부분 큐브를 정의할 수 있고 Mandatory 차원을 설정하여 필수 포함 차원을 설정할 수 있다. 그리고 Hierarchy(계층) 차원을 설정하여 계층 구조 차원의 경우 모든 조합을 빌드하지 않고 필요한 부분만 빌드한다. Joint 차원을 설정하여 같이 나와야 하는 차원을 정의하는 등 여러 가지 옵션값이 있다. 본 논문에서는 간단한 시스템 구축을 위해 추가로 설정값을 추가하지는 않았다.



## 6 Apache Druid를 이용한 전력 스트림 데이터의 분산 OLAP

이 장에서는 Apache Druid를 이용하여 전력 스트림 데이터의 다차원 OLAP 분석에 관한 연구를 살펴본다. 제6.1절에서는 시스템 구성 및 동작에 대해서 살펴보고, 제6.2절에서는 스트림 데이터 처리 방식 2가지에 대해서 살펴본다. 제6.3절에서는 다른 수준의 집계 모델의 비교에 대해 살펴보고, 마지막으로 제6.4절에서는 간단한 분석 질의 처리와 질의 GUI 화면을 살펴본다.

### 6.1 전력 스트림 분석 시스템 구성 및 동작

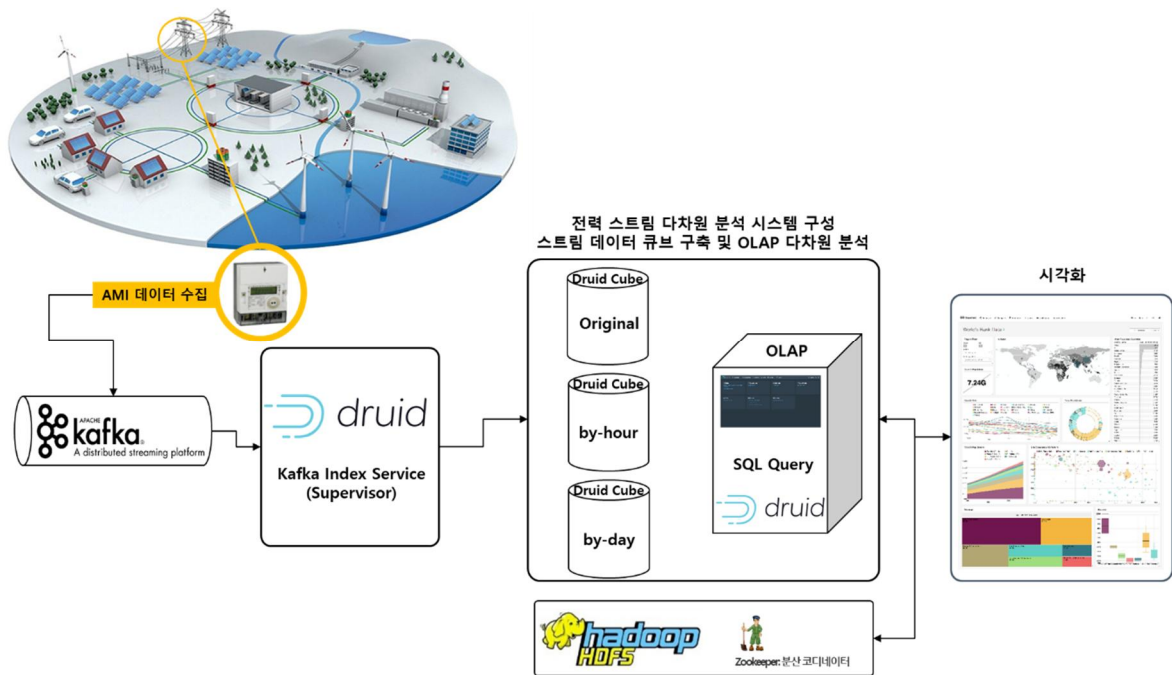


그림 12. 전력 스트림 분석 시스템 구성

위 그림 12는 Druid를 활용하여 전력 스트림 다차원 분석 시스템을 구성한 것이다. AMI 미터기 혹은 DCU에서 데이터를 전송하면 Kafka의 토픽으로 전송하도록 한 이후에 Druid의

Kafka Index Service를 이용하여 토픽에 있는 데이터를 수집한다. 이후에 SQL 질의를 이용한 다차원 분석을 할 수 있고 Superset을 통하여 차트의 형태로 시각화하고 대시보드를 구성하였다. Superset을 통하여 직접 질의를 할 수도 있으며 여러 time granularity로 변환하며 데이터를 시각화하여 분석할 수 있다.

## 6.2 스트림 데이터 처리

### 6.2.1 스트림 데이터 처리 방식

Druid에서 스트림 데이터를 수집하는 방식에는 2가지가 있는데 (1) 라이브러리 형태로 Druid의 API를 이용하여 Kafka 및 다른 스트리밍 처리 도구들과 연동하는 Tranquility를 사용하는 Stream Push 방식, (2) Real-Time Node의 수집 방식으로 Kafka와 연동하여 일정 시간마다 해당 topic의 데이터를 specFile에 지정한 형태로 가져오는 Supervisor를 두는 Stream Pull 방식이 있다. 각 방식에 대해 자세히 살펴본다.

#### 6.2.1.1 Stream Push 방식(Tranquility)

Druid는 Tranquility라는 스트림 클라이언트 API를 제공하는데 Apache Storm이나 Spark Streaming과 같은 스트리밍 엔진과 연동이 가능하다. 이러한 API를 이용하여 데이터를 직접 Druid에 적재하는 방식을 Push 방식이라 한다. Druid의 실시간 indexing task를 생성하고 파티셔닝 복제 등을 다룬다. 아래 그림 13은 Kafka와 스트리밍 엔진과 연동하여 Push 방식의

실시간 스트림 처리를 도식화한 것이다. 데이터 수집 시에는 데이터 전달의 역할만 하고 스트리밍 엔진 부분에서 데이터를 필요에 따라 가공하는 형태로 구현된다.

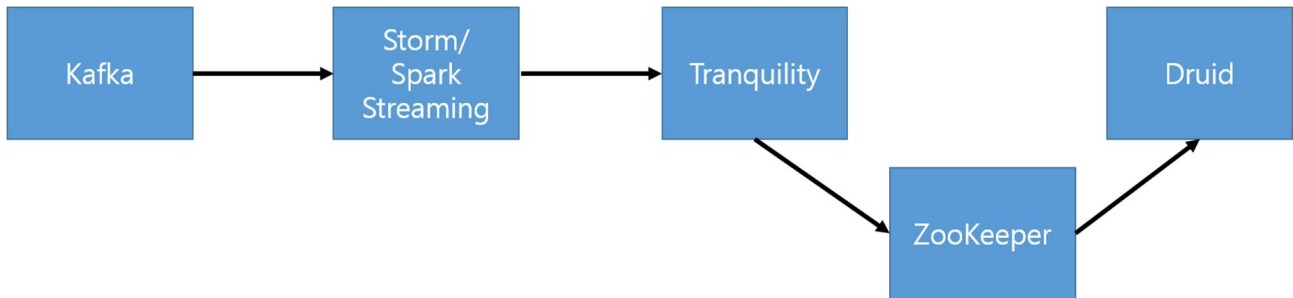


그림 13. Stream Push 방식 도식화

#### 6.2.1.2 Stream Pull 방식(Supervisor, Kafka-index-service)

Druid 클러스터 내에 존재하는 Real-Time Node의 수집 방식으로 수집 데이터의 포맷(ingestion specFile)을 지정하여 수집하는 방식이다. 포맷은 json 형식으로 작성되며 dimension, measure(druid에서는 metric라고 표현함) 등에 관해 정의한다. 아래 그림 14은 Kafka로부터 Pull 방식의 수집 형태를 도식화한 것이다. 데이터의 수집 순서는 다음과 같다. (1) Kafka의 특정 topic으로 데이터를 정제한 뒤 보낸다. (2) Druid의 Overload에서 특정 topic의 메시지를 구독하도록 consumer Task를 생성한다. (3) Real-Time Node에 수집하여 적재 (4) 지정한 세그먼트 주기 단위로 Druid 내의 딥 스토리지로 데이터 전달하여 저장됨 이렇게 하여 Real-Time Node를 Kafka의 consumer로서 역할을 수행하도록 하는 것이다.

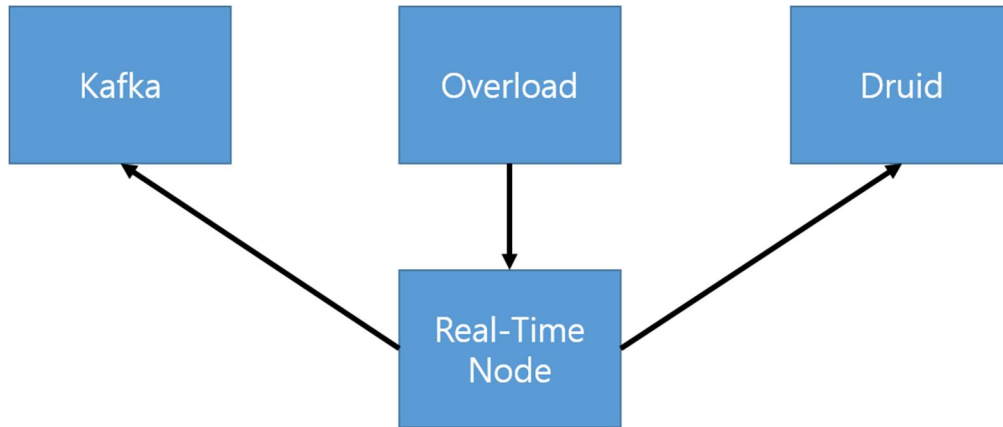


그림 14. Stream Pull 방식 도식화

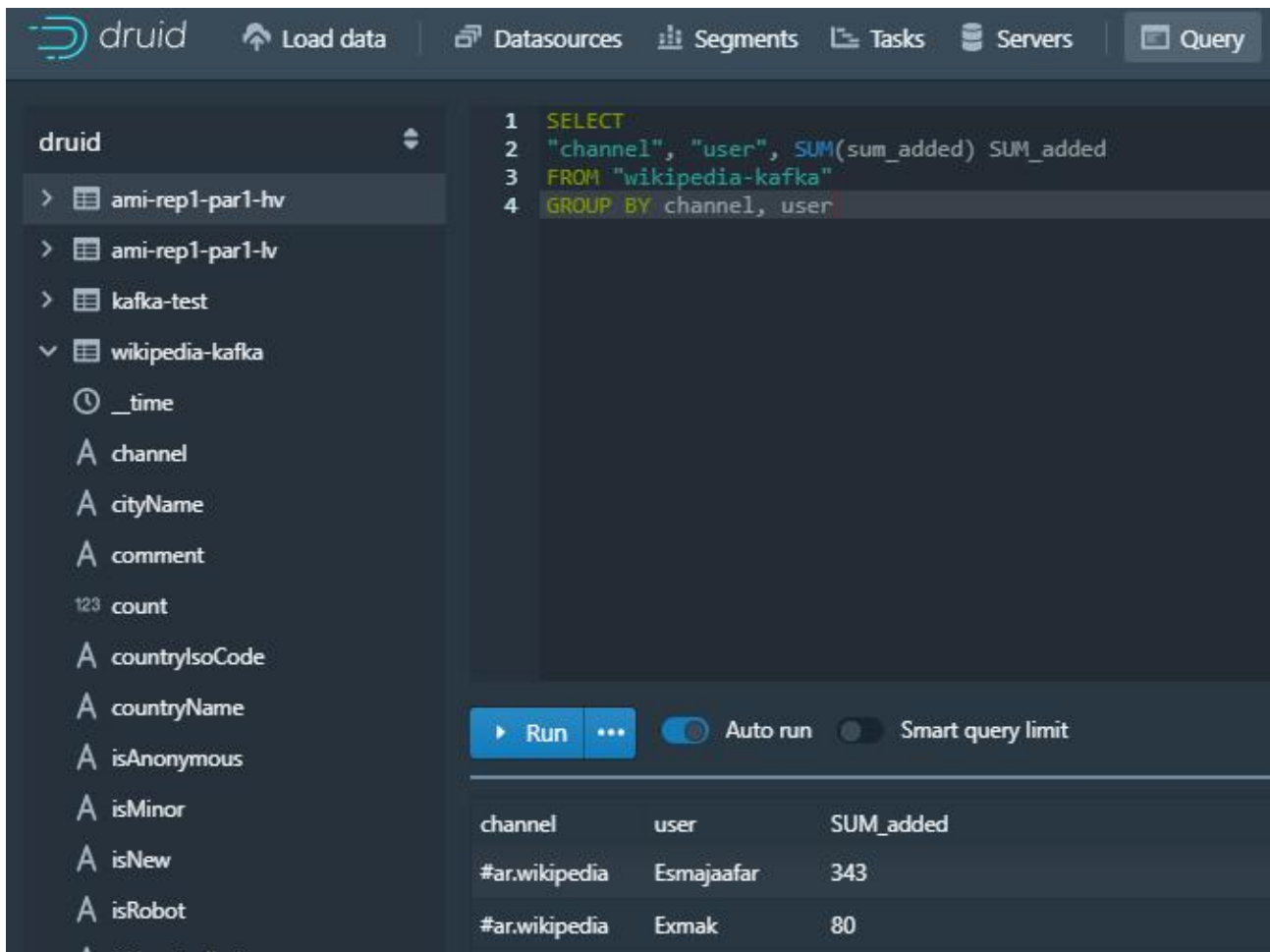
### 6.3 다른 집계 수준의 데이터 구성

Druid에는 몇 가지의 기능이 있으며, 많은 기능 중 하나인 질의 query granularity를 이용하여 데이터의 Roll-up 수준을 다르게 저장하여 저장 공간을 절약하고 질의 속도를 빠르게 할 수 있다. 집계 수준을 분 단위, 시간 단위, 일 단위 이렇게 3가지로 저장하여 데이터 소스를 구성하였다. 이후에 각각 다른 데이터 소스에 질의하였을 시 데이터의 요약 수준이 다른 것을 확인할 수 있고 요약되었기 때문에 저장 공간 상에서도 시간 단위가 분 단위보다, 일 단위가 시간 단위보다 용량을 절약할 수 있는 것을 확인하였다.

### 6.4 질의 처리 예시

Druid를 이용한 간단한 분석 질의 예제에 대해 살펴보고 SQL 질의 화면을 살펴본다. 아래 그림 15는 Druid의 질의 탭의 일부이다. 왼편에서 Druid 클러스터가 수집한 데이터 소스의 목록을 보여주고 세부 정보를 확인할 수 있다. 오른편에서는 일반적인 SQL 질의를 사용할 수 있고, 아래에서 테이블 형태로 질의 결과를 확인할 수 있다. 여기서는 “Wikipedia-

kafka" 데이터 소스에서 여러 컬럼으로 그룹화되어 미리 계산된 다차원의 데이터에서 특정 컬럼(sum\_added 수치 값)을 채널, user 2가지의 컬럼으로 그룹화하여 총합을 집계하는 간단한 예제이다.



The screenshot shows the Druid Query Console interface. On the left, a tree view shows the database structure with 'wikipedia-kafka' selected. The main area displays a SQL query:

```
1 SELECT
2   "channel", "user", SUM(sum_added) SUM_added
3 FROM "wikipedia-kafka"
4 GROUP BY channel, user
```

Below the query, there are buttons for 'Run' and 'Auto run', and a 'Smart query limit' toggle. The results are displayed in a table:

channel	user	SUM_added
#ar.wikipedia	Esmajaafar	343
#ar.wikipedia	Exmak	80

그림 15. Druid 질의 탭

## 7 시스템 구축 및 분석

이 장에서는 본 논문에 사용된 모델들을 구축한 환경과 2개의 시스템 별로 예시를 들어 설명하고 질의 결과 및 시각화하는 것에 대해 살펴본다. 제7.1절에서는 시스템 구축 환경에 대해서 살펴보고, 제7.2절에서는 Kylin 분석 예시를 살펴본다. 마지막으로 제7.3절에서는 Druid 분석 예시를 살펴본다.

### 7.1 시스템 구축 환경

본 논문에서 구축하고자 하는 시스템을 위해서 마스터 노드 1대, 슬레이브 노드 10대의 총 11대의 클러스터를 구축하였다. 모든 노드는 CentOS 7.7 버전의 운영체제를 사용했고, 각 서버는 Intel i7-4770 @ 3.40 GHz 4 Core의 CPU, 노드 1-5번은 4GB RAM, 마스터 및 6-10번 노드는 12GB RAM, 노드당 1TB HDD의 사양을 사용했다. 각 노드들은 ~bps의 네트워크에 연결되어 있다. 분산 클러스터 구축을 위해 Hadoop 2.9.0을 사용하였고 Kylin 2.6.1, Druid(incubating) 0.16.0, Kafka 2.3.0을 사용하였다. Kylin 구축 시에는 11대의 클러스터를 모두 사용하였고, Druid 구축 시에는 마스터와 RAM 사양이 높은 6-10번 노드 총 6대의 클러스터로 구축하였다.

## 7.2 Kylin 분석 예시

본 논문에서 다루고자 하는 전력 데이터를 4장에서 설명했던 Kylin 클러스터 환경을 이용하여 의미 있는 질의를 하고 시각화하는 과정을 설명한다.

### 7.2.1 생성한 큐브의 결과

표 4는 위에서 구성한 스타 스키마로 실제 생성한 Kylin 내의 큐브의 크기 및 측정된 큐브 빌드에 걸리는 시간이다. 기간별로 데이터를 나눠서 용량별로 빌드했을 때 큐브 사이즈의 변화율 및 빌드 시간을 측정하고자 했다.

표 4. Kylin 큐브 빌드 결과

원본 테이블의 크기	큐브의 크기	확장 비율	빌드 시간
320 KB	650 KB	약 203 %	2.98 mins
640 KB	960 KB	약 150 %	3.70 mins
960 KB	1.24 MB	약 130 %	3.05 mins
1.26 MB	1.54 MB	약 120 %	2.97 mins
88.68 MB	163.66 MB	약 185 %	6.33 mins

### 7.2.2 다차원 질의 및 시각화 예시

아래 그림들은 빌드한 큐브의 의미있는 다차원 질의 및 Kylin에서 제공하는 시각화 차트에 대한 간단한 2가지 예시와 설명이다.

```

SELECT
  D.Day_DT AS PART_DT,
  C.Business_Type AS METER_KIND
  MAX(A.전력 사용량)
  SUM(A.전력 사용량)
FROM AMI_Data A
INNER JOIN Date D ON A.Date_ID = D.ID
INNER JOIN Code2Type C ON A.Meter_ID = C.ID
GROUP BY
  D.Day_DT, C.Business_Type

```

그림 16. 일별/업종별 전력 사용량 집계 질의

그림 16은 일별/업종별 전력 사용량 집계 값을 출력하는 질의 예제이다. 여기에 SUM 값을 내림차순으로 정렬하여 가장 많이 사용한 날과 업종을 찾고자 했다. 질의 결과로 아래 그림 17과 같이 2018년 7월 29일에 100번의 업종이 가장 많이 사용한 것을 테이블 형태의 결과로 확인할 수 있다. 그리고 아래 그림 17은 Kylin에서 자체 제공하는 시각화로 일별 전력 사용 총합을 보여준다. 아래 그림 18의 그래프로 확인할 시 특정 날짜의 전력사용량을 바로 확인할 수 있다.

PART_DT ▾	METER_KIND ▾	MAX_ACT ▾	SUM_ACT ▾
2018-07-29	100	56.969999999...	89549.219999...
2018-07-28	100	77.634999999...	84385.670999...
2018-07-30	100	51.961999999...	81644.451000...
2018-07-31	100	55.662000000...	81598.2
2018-07-26	100	57.567000000...	78252.162000...
2018-07-27	100	63.500999999...	77365.446000...
2018-07-21	100	61.552000000...	75569.912000...
2018-07-25	100	55.936000000...	74644.684000...

그림 17. 일별/업종별 전력 사용량 집계 결과



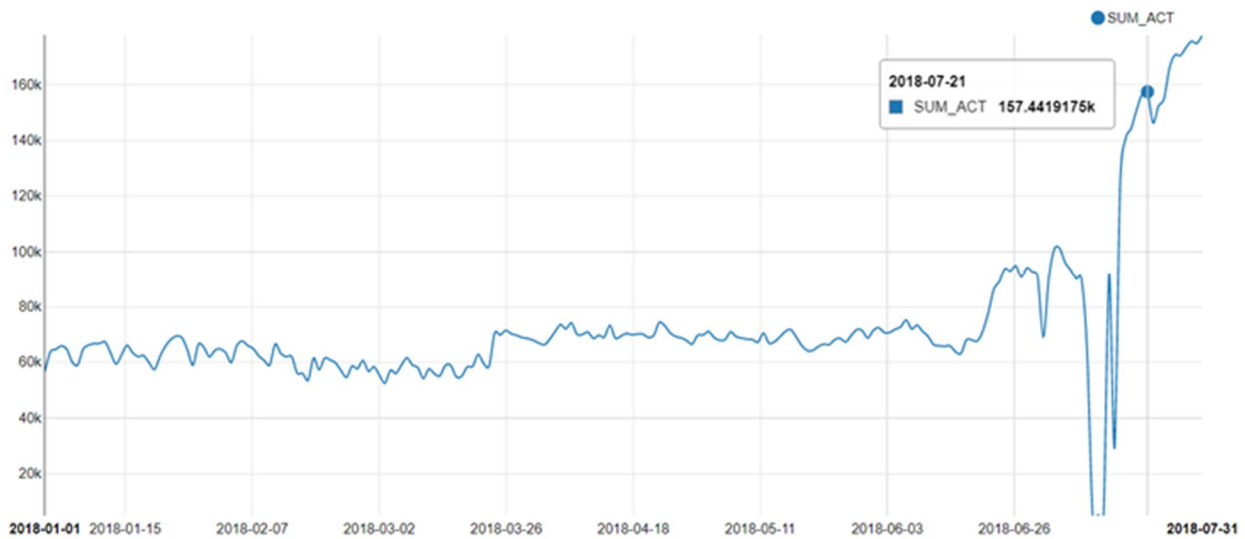


그림 18. 일별/업종별 전력 사용량 집계 시각화

그림 19는 월별 집계 값을 출력하는 예제이다. 결과로는 월 단위 가장 많이 사용한 미터기의 전기 사용량(MAX)과 월별 전기 사용량 총합(SUM)이다. 아래 그림 20에서와 같이 SUM 값이 가장 높은 달은 2018년 7월인 것을 확인할 수 있다.

```
SELECT
  D.Month_DT AS MONTH_DT,
  MAX(A.전력 사용량)
  SUM(A.전력 사용량)
FROM AMI_Data A
INNER JOIN Date D ON A.Date_ID = D.ID
GROUP BY
  D.Month_DT
```

그림 19. 월별 전력 사용량 집계 질의

	MONTH_DT ▾	SUMATION ▾	MAXIMUM ▾
	2018-01-01	1976262.2799...	3218.4900000...
	2018-02-01	1709044.0689...	3118.11999999...
	2018-03-01	1880593.3859...	2751.75
	2018-04-02	2035022.9569...	2785.8099999...
	2018-05-01	2123626.9605	2833.5650000...
	2018-06-01	2270806.5905	2863.6450000...
	2018-07-01	3556034.9344...	2909.4550000...

그림 20. 월별 전력 사용량 집계 결과

위의 경우와 같이 아래 그림 21은 월 단위 전력 사용 총합을 시각화한 것이다. 특정 달의 전력 사용량을 확인 및 가장 많이 사용한 달을 한눈에 확인할 수 있다.

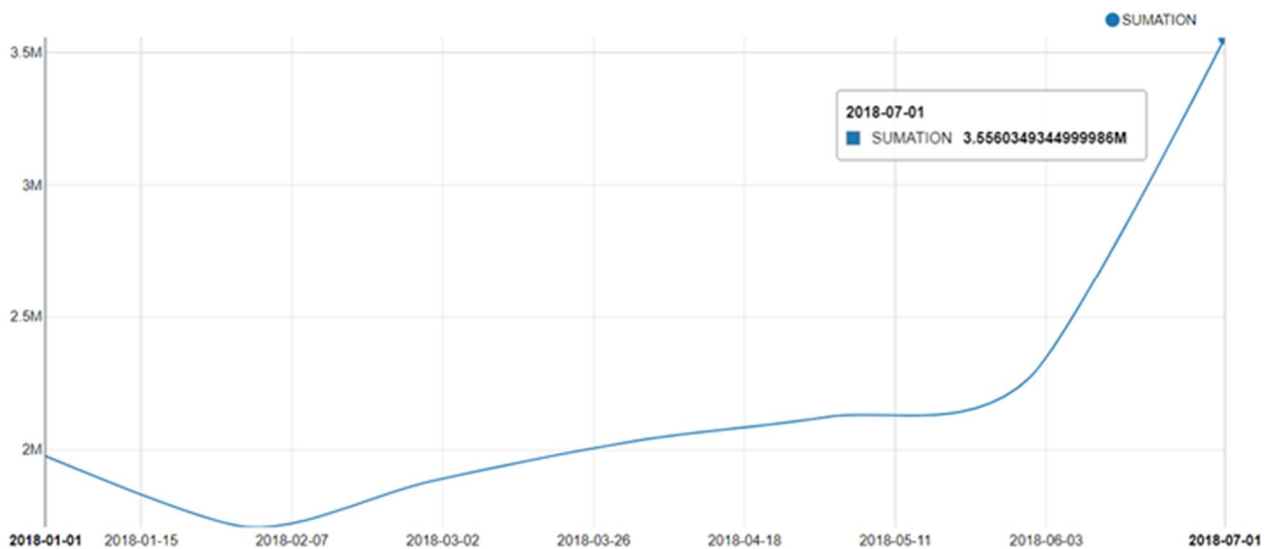


그림 21. 월별 전력 사용량 집계 시각화

### 7.3 Druid 분석 예시

본 논문에서 구성한 Druid 시스템 내에서의 간단한 질의와 이를 Superset을 통하여 시각화한 결과에 대한 설명이다.

### 7.3.1 질의 결과

먼저 Druid 시스템은 위에서 설명한 것과 같이 데이터를 수집하는 동안 특정 컬럼의 정해진 형태의 Roll-up을 지원하여 데이터를 요약한다. 요약을 할 때는 전체 데이터에서 수집할 차원을 선택한 후 지정한 query granularity에 해당하는 데이터를 요약하게 된다. 그렇게 수집 단계에서부터 Roll-up이 되고 Druid는 Kylin에서의 스타 스티마 조인과 같은 기능을 지원하지 않기 때문에 구축한 시스템에서의 질의 예시로 일반적인 SQL을 하게 된다. 6.3절에서 설명한 것과 같이 다른 수준의 query granularity로 데이터 소스를 생성하여 원본의 경우는 분 단위로 모든 데이터를 요약하지 않은 상태 그대로 질의가 가능하고 시간 단위로 query granularity를 설정한 데이터 소스의 경우는 질의를 하였을 시에 기본적으로 시간 단위의 집계된 상태로 질의 결과가 나오게 된다. 또, 일 단위로 query granularity를 설정한 데이터 소스의 경우에는 질의하였을 때 기본으로 일 단위로 집계된 상태의 질의 결과가 나오게 된다. 집계된 개수는 metric 중 count 컬럼을 확인해보면 몇 줄의 데이터가 요약됐는지 확인할 수 있다. 아래 그림들은 각각 다른 수준의 query granularity로 질의한 결과이다. 아래 그림 22는 공통으로 사용한 질의이다. 질의 예시는 고압 LP 데이터에서 결과를 얻었고 저압의 경우와는 다르게 dcu 정보가 없다.

```

SELECT
    _time
    Meter_ID
    MAX(전력 사용 관련 컬럼)
    SUM(전력 사용 관련 컬럼)
FROM AMI_Data

```

그림 22. 시간/미터기별 전력 관련 컬럼 집계 질의

_time	METER_ID	max_ACT_PWR	max_LAGG	max_LEAD	sum_ACT_PWR	sum_LAGG	sum_LEAD
2015-01-01T00:15:00.	zl1EEzaaPaP	129.46	3.28	0	258.92	6.56	0
2015-01-01T00:15:00.	iHqMqizHsc	1.73	0.13	0.68	3.46	0.26	1.36
2015-01-01T00:15:00.	gLmpVpgg\	2.02	0	1.25	4.04	0	2.5
2015-01-01T00:30:00.	w7Fssswww	0.64	0	2.96	1.28	0	5.92
2015-01-01T00:30:00.	G8yPaPwPG	0.56	0	2.76	1.12	0	5.52

그림 23. 분 단위의 원본 데이터 질의 결과

_time	METER_ID	count	max_ACT_PWR	max_LAGG	max_LEAD	sum_ACT_PWR	sum_LAGG	sum_LEAD
2018-03-03T00:00:00.	al1E4EaEnEzi	3	3.12	1.93	0	9.33	5.69	0
2018-03-03T00:00:00.	iHqMqizHsoT	3	2.5	0	1.3	6.1400003	0	3.25
2018-03-03T00:00:00.	gLmpVpggVzpg	3	19.09	6.18	0	56.71	18.4	0
2018-03-03T01:00:00.	01dUJ0Jf1dm	4	9.05	0.07	1.51	32.43	0.21	5.8900003
2018-03-03T01:00:00.	A12f0dddddACA	4	0	0	0.01	0	0	0.02

그림 24. 시간 단위의 데이터 질의 결과

_time	METER_ID	count	max_ACT_PWR	max_LAGG	max_LEAD	sum_ACT_PWR	sum_LAGG	sum_LEAD
2018-01-01T00:00:00	dN2GNfdd	190	16.21	6.72	0	2238.76	661.0998	0
2018-01-01T00:00:00	i5q5ziizoHi	192	5.78	7.81	0.02	805.89966	671.57996	0.1199999
2018-01-01T00:00:00	gLmpVpgg	190	3.47	1.32	0	417.81992	234.22008	0
2018-01-02T00:00:00	geMpeVgg	97	26.54	7.58	1.03	1175.5106	249.67998	16.18
2018-01-02T00:00:00	01dUJ0Jf1d	97	31.58	0.58	2.83	1267.3298	10.3600025	172.5

그림 25. 일 단위의 데이터 질의 결과

위의 결과에서 볼 수 있듯이 다른 query granularity로 저장된 데이터 소스의 경우 해당 집계 수준으로 데이터가 요약된 것을 확인할 수 있다. 그림 23의 원본 데이터의 경우 AMI 데이터의 특징으로 미터기를 측정하는 시간이 15분 단위와 60분 단위 2가지가 존재하는데 이 경우에는 15분 단위의 데이터이기 때문에 분 단위로 query granularity를 지정하여도 15분 단위로 질의 결과가 나오게 된다.

### 7.3.2 Superset을 통한 시각화

Superset에서는 시각화를 할 때 시간 세부성의 정도를 조정할 수 있는데 Druid에서 데이터 소스를 정의할 때 Roll-up을 지정한 것의 경우를 원본으로 만들 수는 없지만 분 단위 데이터를 시간으로 집계해주거나 시간 단위 데이터를 일로 집계해주는 등으로 시각화할 수 있다. 아래 그림 26는 15분 단위로 시간이 지남에 따른 전력 사용량의 합계를 집계하여 시각화한 것이다. 하단에 형성되는 구간은 정각에서부터 15분마다의 데이터이고 상단에 높은 값을 나타내는 부분이 매 정각을 나타낸다. 그래프가 이런 식으로 표현이 되는 것은 매 정각에서는 15분 단위의 데이터와 60분 단위의 데이터를 모두 집계하기 때문에 정각에서부터 15분마다의 데이터보다 높을 수밖에 없기 때문이다.

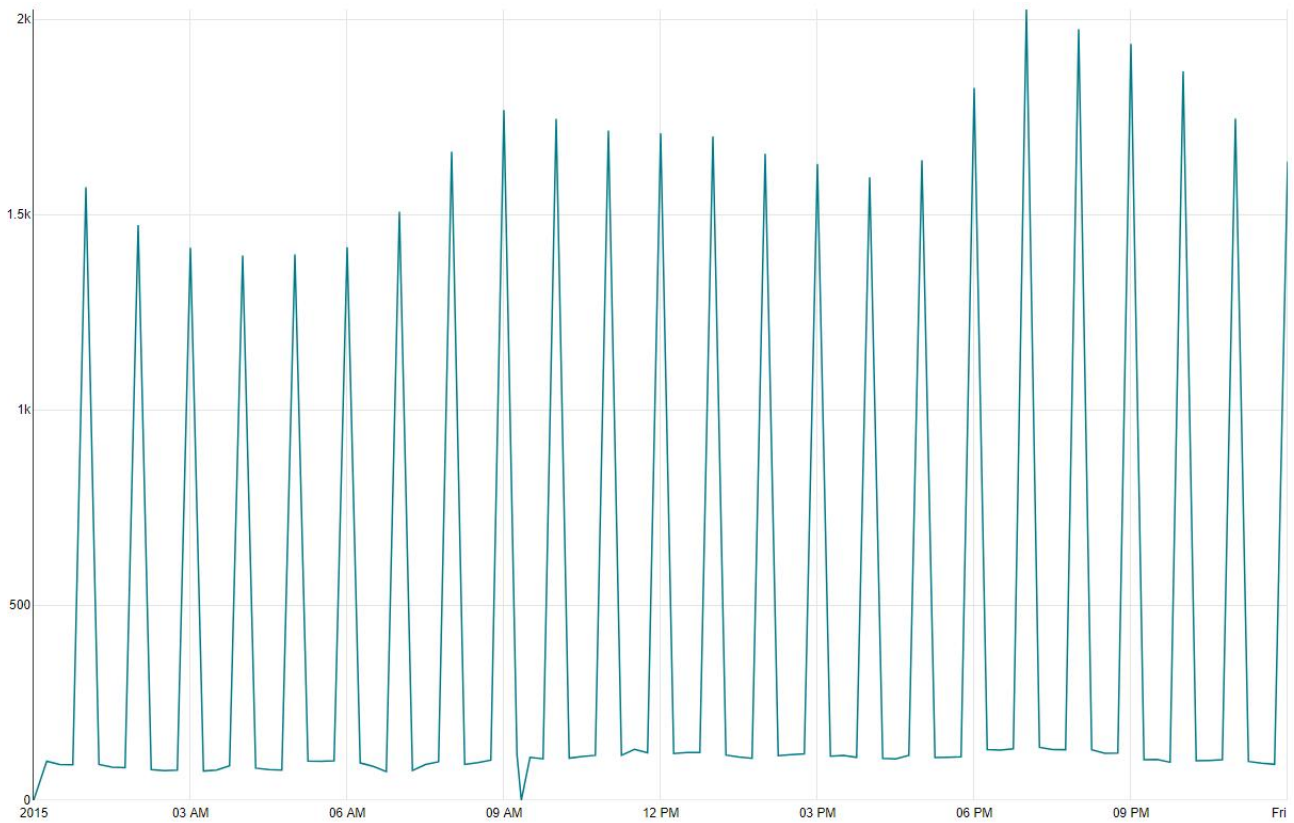


그림 26. 15분 단위 구간별 전력 사용량 합 집계 시각화

Superset에서는 모니터링하기 위한 대시보드를 구성할 수 있다. 예시로 간단한 몇가지 AMI 데이터를 이용하여 대시보드를 구성하는 요소를 설명한다. 대시보드를 구성하기 앞서서 먼저 차트를 그려 저장하고 대시보드에 불러오면 된다. 아래 그림들은 실시간 대시보드를 구성하는 차트들에 대한 설명과 구성된 대시보드이다. 그림 27은 1시간동안의 사용 전력량을 집계한 것 중 최대 사용량을 큰 글씨로 보여주는 차트이다. 그림 28은 Meter\_ID로 그룹핑하고 time granularity를 하루로 설정하여 상위 5개의 미터기와 미터기의 전력 사용량을 보여주는 차트이다. 그림 29은 월별로 총 전력사용량 대비 무효전력량 2가지의 정도를 알고자하는 차트이다. 그림 30은 분기별로 총 전력사용량을 집계한 것이다. 이 예제에서는 봄보다는 여름,

그리고 겨울에 더 많은 전력을 사용한 것을 알 수 있다. 그림 31은 이러한 여러가지 차트를 이용하여 대시보드를 구성하여 실시간으로 모니터링할 수 있는 환경을 구축하였다.

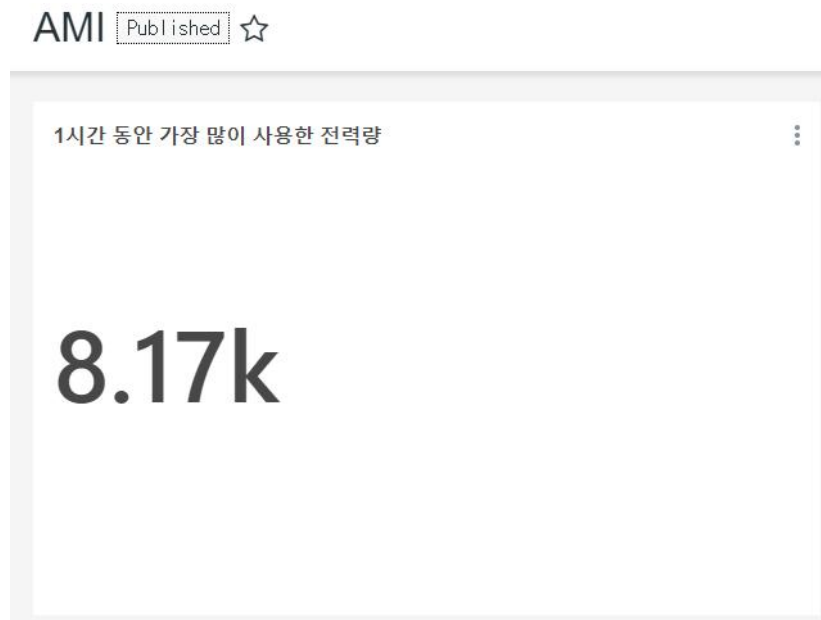


그림 27. 시간 단위 최대 전력 사용량 집계 시각화

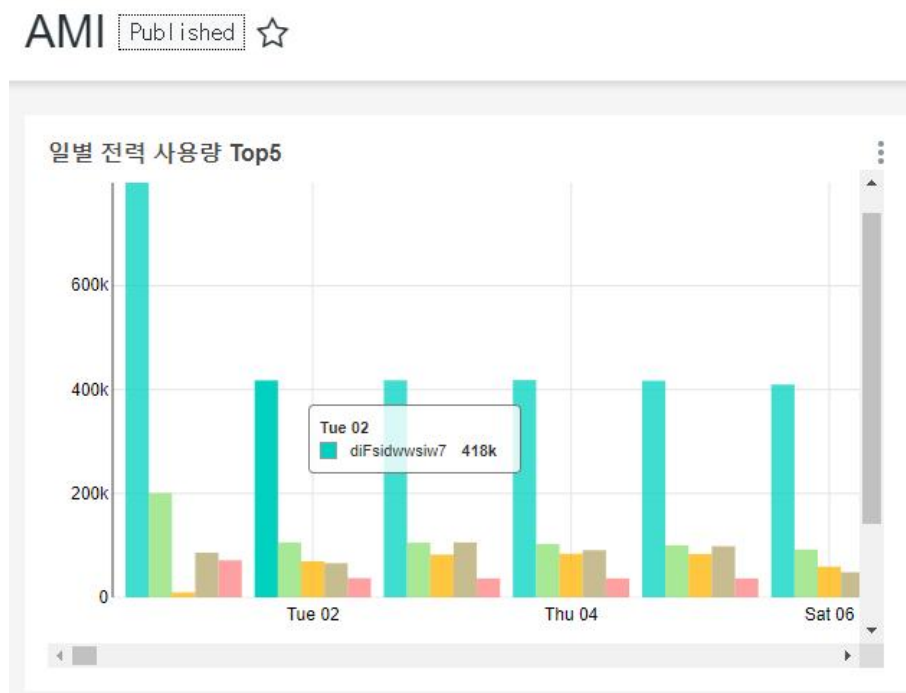


그림 28. 하루 단위 전력 사용량 Top 5 미터기 시각화

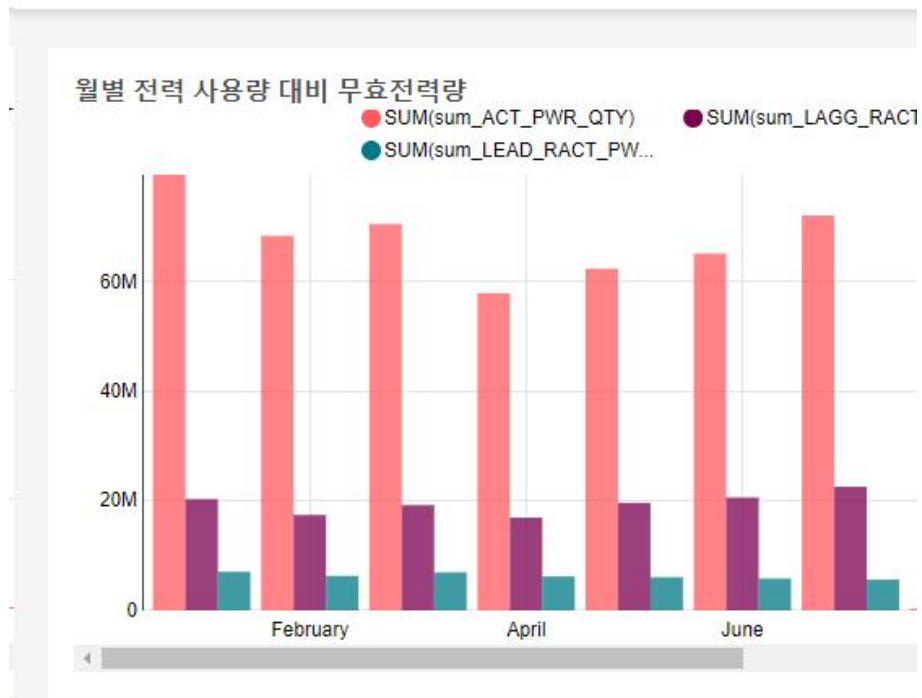


그림 29. 월 단위 전력 사용량 대비 무효전력량 시각화

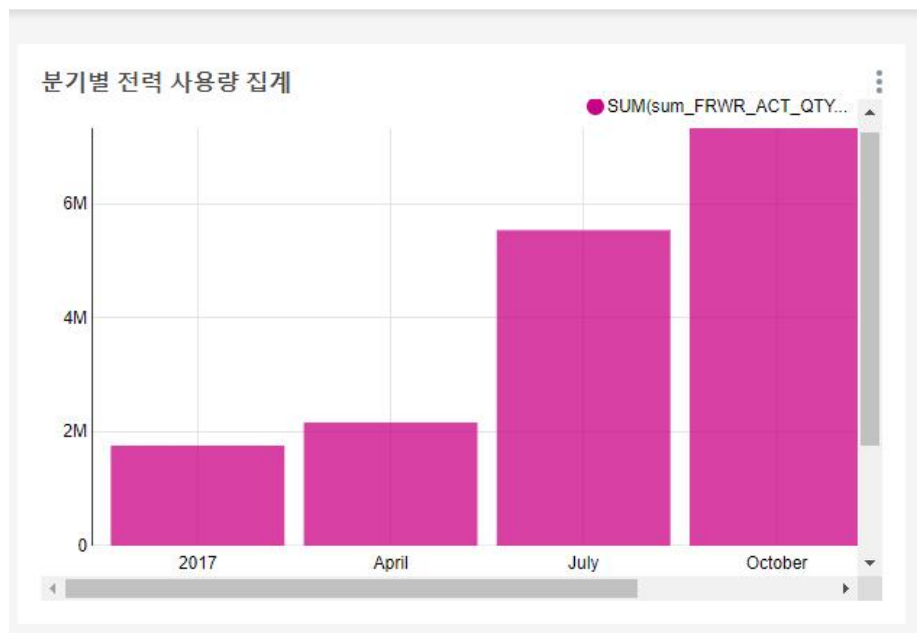


그림 30. 분기별 총 전력 사용량 집계 시각화



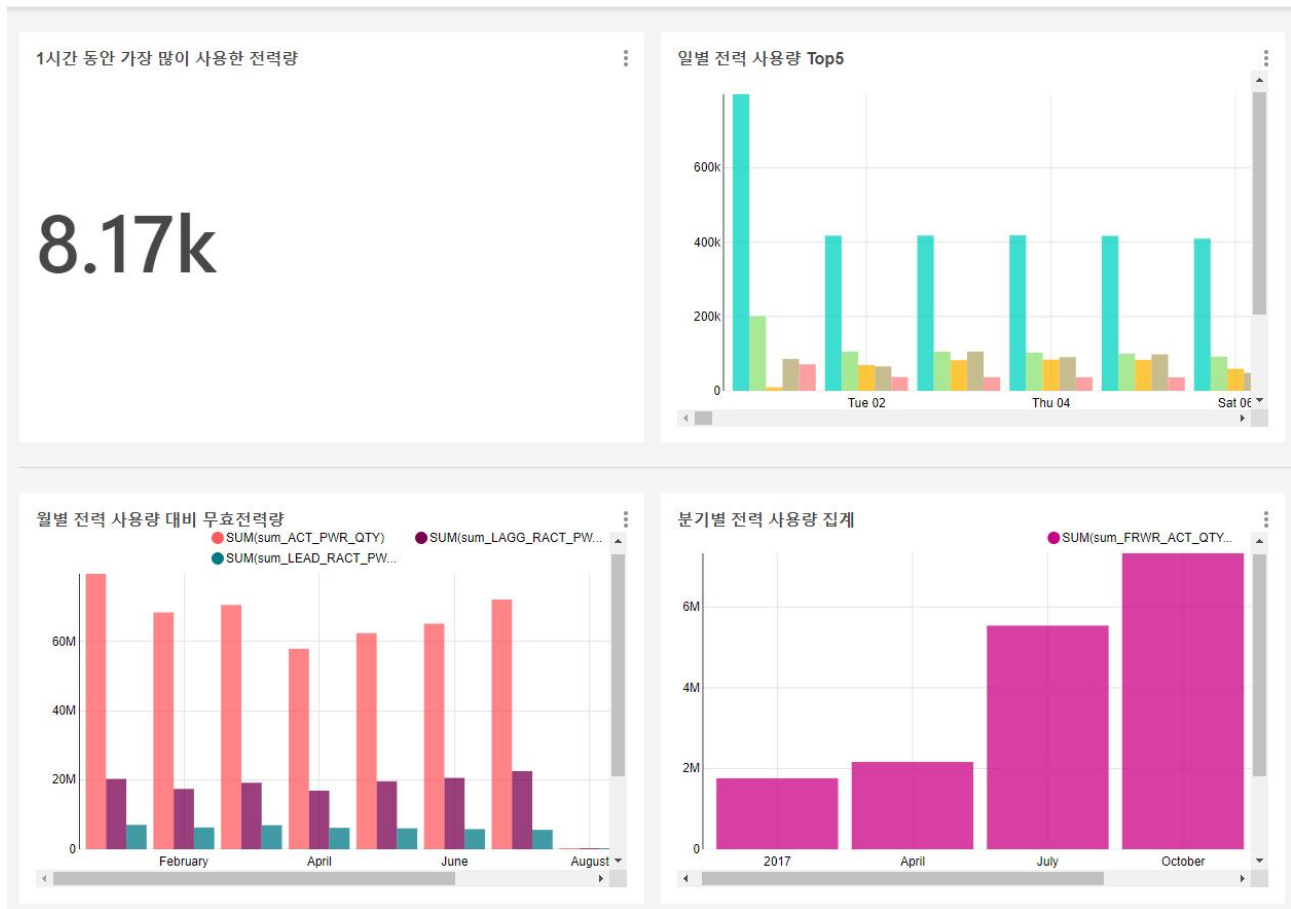


그림 31. 위의 4가지 차트를 포함하는 실시간 대시보드

## 8 결 론

본 논문에서는 대용량 데이터 및 스트림 데이터에 대한 다차원 분석을 위해 2개의 OLAP 플랫폼을 활용한 분석 시스템을 구축하였다. 스마트 그리드와 같은 곳에서 생성하는 스트리밍 센서 데이터의 양은 무한하고 집계할 때 필요한 높은 계산 비용과 저장 공간은 한계가 있다. 이 문제를 해결하기 위해 하둡 에코시스템 기반의 분산 처리 방법을 이용하여 필요에 따라 확장성을 가지고 문제를 해결하는 연구들이 있었고 이를 활용하여 배치 데이터 처리 방식과 스트리밍 데이터 처리 방식 두가지를 해결하고자 하였다. 배치 처리 방식의 경우 데이터를 전처리한 후 스타 스키마 형태로 큐브를 생성하고, 스트리밍 처리 방식의 경우 Apache Kafka를 통해서 해당 토픽의 데이터를 수집하는 형태로 스트림 데이터의 분산 OLAP을 하였다. 사용자는 Apache Superset을 통해서 원하는 차트의 형태로 시각화하고 모니터링하기 위해 대시보드를 생성하여 사용자가 원하는 형태의 분석을 실시간으로 확인할 수 있다.

오늘날 많은 응용 분야에서 다차원 분석 기술을 이용하여 의사결정에 활용하고 있다. 또한 많은 기업들이 빅데이터 처리를 위해 하둡 에코시스템을 도입하고 있다. 앞으로 이러한 다차원 분석 기술을 활용한 도구들이 실제 응용 분야에서 용이하게 사용될 것으로 기대된다. 본 논문에서 사용된 것과 같은 오픈소스 도구들이 많은데, 과거와는 다르게 기업에서도 이러한 오픈소스 도구들을 사용하려 하고 있고 점점 더 많은 곳에서 이러한 시스템을 구축하여 사용될 것이라 생각한다. 그리고 앞으로 IoT가 산업 분야 전반에 활용될 것으로 전망되는

지금, 스트림 데이터 분석의 역할은 중요할 것으로 여겨지고 있다. 따라서 사용자에게 본 논문에서 구축한 시스템을 이용하여 효율적으로 전략적인 의사결정을 함에 있어 도움이 될 것으로 기대한다.

## 9 참고 문헌

- [1] Hadoop, <https://hadoop.apache.org/>
- [2] Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010, May). The hadoop distributed file system. In *MSST* (Vol. 10, pp. 1-10).
- [3] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1), 107-113.
- [4] HBase, <https://hbase.apache.org/>
- [5] Díaz, M., Martín, C., & Rubio, B. (2016). State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *Journal of Network and Computer applications*, 67, 99-117.
- [6] Kafka, <https://kafka.apache.org/>
- [7] Kylin, <https://kylin.apache.org/>
- [8] Yang, F., Tschetter, E., Léauté, X., Ray, N., Merlino, G., & Ganguli, D. (2014, June). Druid: A real-time analytical data store. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 157-168). ACM.
- [9] Superset, <https://superset.apache.org/>
- [10] Berson, A., & Smith, S. J. (1997). *Data warehousing, data mining, and OLAP*. McGraw-Hill, Inc.
- [11] Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *ACM Sigmod record*, 26, 65-74.
- [12] Thomsen, E. (2002). *OLAP solutions: building multidimensional information systems*. John Wiley & Sons.
- [13] Ranawade, S. V., Navale, S., Dhamal, A., Deshpande, K., & Ghuge, C. (2016). Online analytical processing on hadoop using apache kylin. 2016-12-02]. <http://www.ijais.org/archives/volume12/number2/ranawade-2017-ijais-451682.pdf>.
- [14] Chen, W., Wang, H., Zhang, X., & Lin, Q. (2017, September). An optimized distributed OLAP system for big data. In *2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA)* (pp. 36-40). IEEE.
- [15] Chambi, S., Lemire, D., Godin, R., Boukhalfa, K., Allen, C. R., & Yang, F. (2016, July). Optimizing druid with roaring bitmaps. In *Proceedings of the 20th International Database Engineering & Applications Symposium* (pp. 77-86). ACM.
- [16] Correia, J., Costa, C., & Santos, M. Y. (2019, June). Challenging SQL-on-Hadoop Performance with Apache Druid. In *International Conference on Business Information Systems* (pp. 149-161). Springer, Cham.

- [17] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., . . . Pirahesh, H. (1997). Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery*, 1, 29-53.
- [18] Li, X., Han, J., & Gonzalez, H. (2004). High-dimensional OLAP: A minimal cubing approach. *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, (pp. 528-539).
- [19] Beyer, K., & Ramakrishnan, R. (1999). Bottom-up computation of sparse and iceberg cube. *ACM SIGMOD Record*, 28, pp. 359-370.
- [20] Goil, S., & Choudhary, A. (1997). High performance OLAP and data mining on parallel computers. *Data Mining and Knowledge Discovery*, 1, 391-417.
- [21] Wang, Z., Chu, Y., Tan, K.-L., Agrawal, D., Abbadi, A. E., & Xu, X. (2013). Scalable data cube analysis over big data. *arXiv preprint arXiv:1311.5663*.
- [22] Nandi, A., Yu, C., Bohannon, P., & Ramakrishnan, R. (2012). Data cube materialization and mining over mapreduce. *IEEE transactions on knowledge and data engineering*, 24, 1747-1759.
- [23] Milo, T., & Altshuler, E. (2016, June). An efficient MapReduce cube algorithm for varied DataDistributions. In *Proceedings of the 2016 International Conference on Management of Data* (pp. 1151-1165). ACM.
- [24] Han, J., Chen, Y., Dong, G., Pei, J., Wah, B. W., Wang, J., & Cai, Y. D. (2005). Stream cube: An architecture for multi-dimensional analysis of data streams. *Distributed and Parallel Databases*, 18, 173-197.
- [25] Farhangi, H. (2009). The path of the smart grid. *IEEE power and energy magazine*, 8(1), 18-28.
- [26] Baran, M., & McDermott, T. E. (2009, March). Distribution system state estimation using AMI data. In *2009 IEEE/PES Power Systems Conference and Exposition* (pp. 1-3). IEEE.
- [27] 유남조, 이은애, 정범진, 김동식. (2019). 공동주택 AMI 데이터 분석 및 K-Means 알고리즘을 이용한 전력 소비 패턴 분석. *대한전자공학회 학술대회*, (), 506-508.
- [28] 김류희, 류도현, 송민석, 최동구, 고영명, 김영진, 김광재. (2019). 캠퍼스 전력 사용 데이터 기반의 신서비스 컨셉 개발. *대한산업공학회 추계학술대회 논문집*, (), 3297-3297.
- [29] 박찬희, 조민수, 박진욱, 노홍찬, 이지은, 박상현. (2019). 시계열 데이터의 장단기 모델링을 통한 전력 수요 예측 기법. *한국정보과학회 학술발표논문집*, (), 409-411.

# **Efficient On-line Multidimensional Analysis System Based on Distributed Stream Cube for Electric Power Data**

Jongwon Jang

*Department of Computer Science  
Graduate School, Kangwon National University*

## **Abstract**

Today, with the proliferation of the Internet of Things(IoT), one of the major technologies of the Fourth Industrial Revolution and the rise of the Smart Grid, the amount of data generated by various devices and sensors is very huge. In the past, the focus was on processing batch data, but now the area of streaming data analysis is emerging in many fields such as IoT and monitoring systems. The analysis of stream data adds value to the stream data by capturing and resolving where immediateness is required. In order to efficiently store and analyze such a large amount of data, OLAP and data cubes have been studied, and there have been studies such as *Stream Cube* for stream processing and distributed cube. In this paper, I constructed a system using two open source OLAP tools based on Hadoop Ecosystem. The system processes batch data using Apache Kylin and processes stream data using Apache Druid. After preprocessing the electric power data, I created a batch data cube and analyzed it with some simple examples, and visualized it. And I use Apache Kafka to send stream data, and collect it in a pull way to store the original data or summarize it. Data can be collected and analyzed in real time via queries. Later, I used Apache Superset to visualize and construct a dashboard for real-time monitoring. As a result of constructing the system,

the data summarized as needed saves the storage space than the original data, and only queries to the level initially set at the time of query were possible.

□ keywords

Big data, Electric Power Data, Data Cube, OLAP, Data Warehouse