



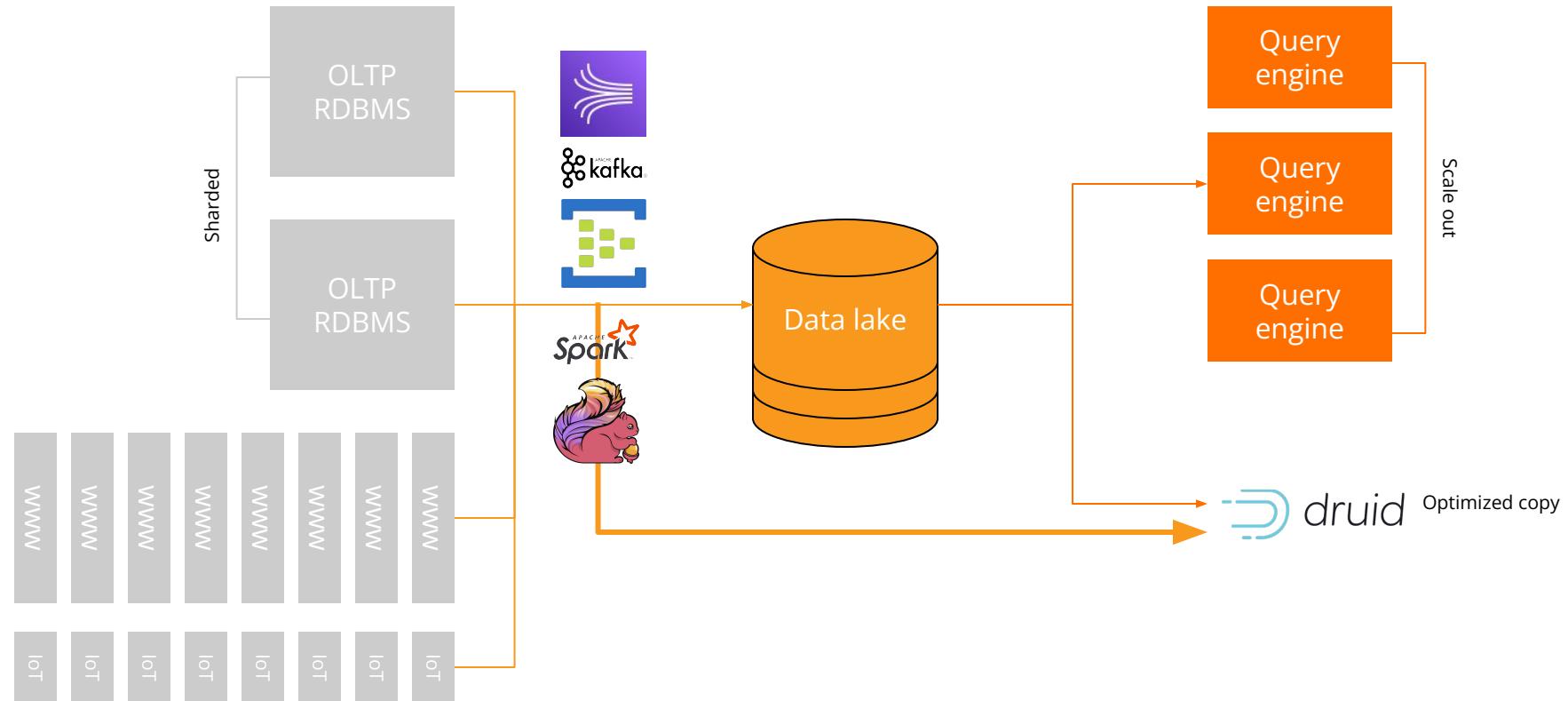
Druid Talk

실시간 분석을 위한 Apache Druid 이야기

이기훈 이사, Imply

데이터 플랫폼과 Apache Druid

Big data + OLAP serving database + streaming



Apache Druid 의 탄생 배경

Fully scalable



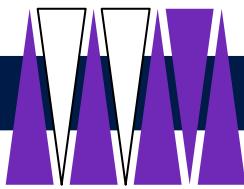
Batch and real-time data

Ad-hoc statistical queries

Low latency delivery

log search

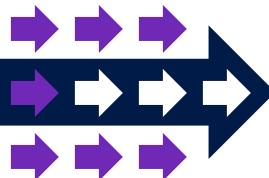
real-time ingest
flexible schema
text search



Fully scalable

timeseries

low latency ingest
time-based storage
time functions

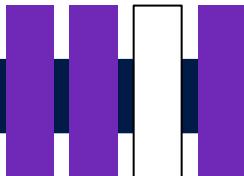


Batch and real-time data

Ad-hoc statistical queries

columnar

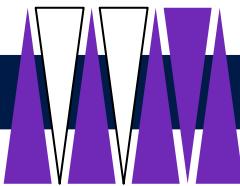
efficient storage
fast analytic queries
data distribution



Low latency delivery

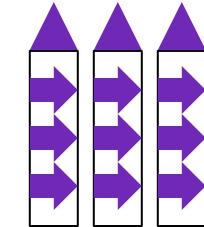
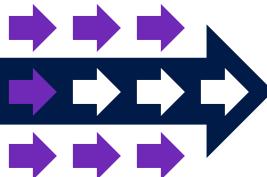
log search

real-time ingest
flexible schema
text search



timeseries

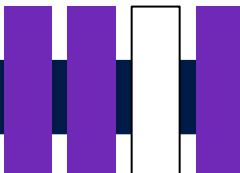
low latency ingest
time-based storage
time functions

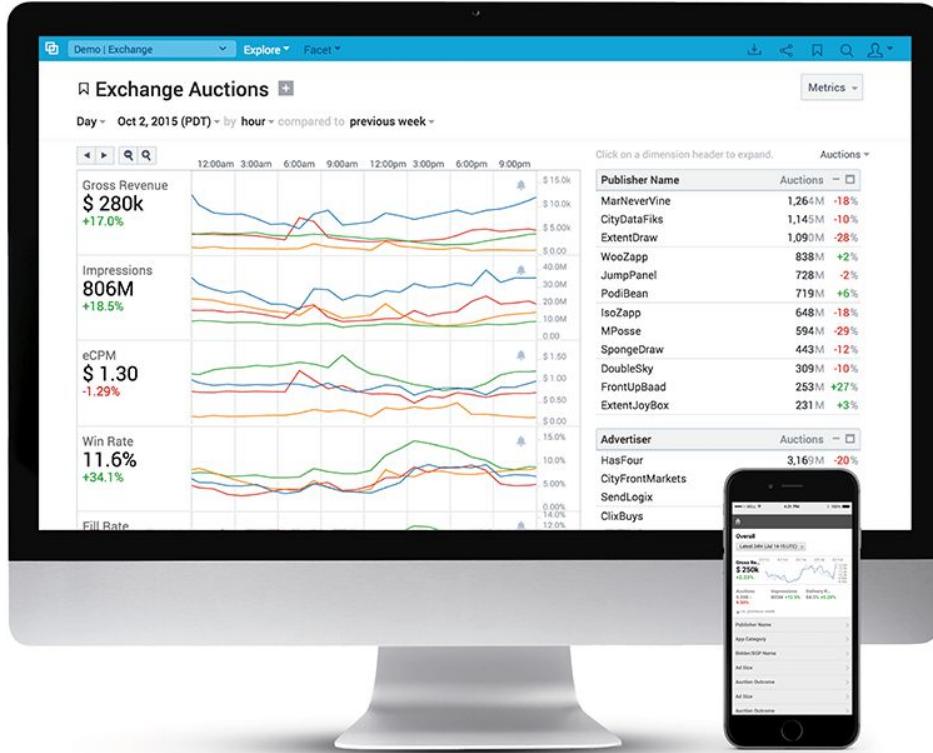


HIGH PERFORMANCE
REAL-TIME
ANALYTICS
DATABASE

columnar

efficient storage
fast analytic queries
data distribution





Advertisers loaded impressions and clicks data and used web and mobile apps to optimize user and ad engagement

Billions of events per month

30+ dimensions



NETFLIX

CONDÉ NAST



dripstat



eBay



YAHOO!

hulu

criteo.

PayPal



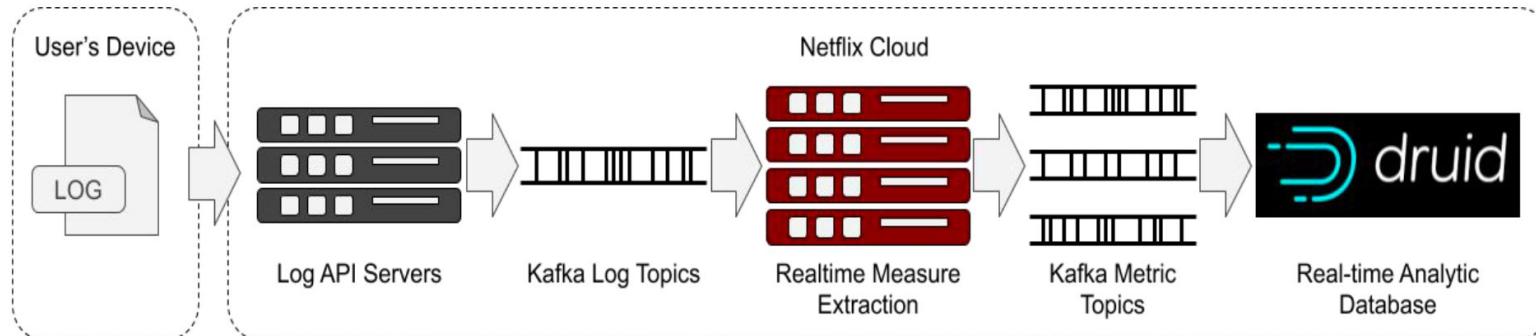
triplelift



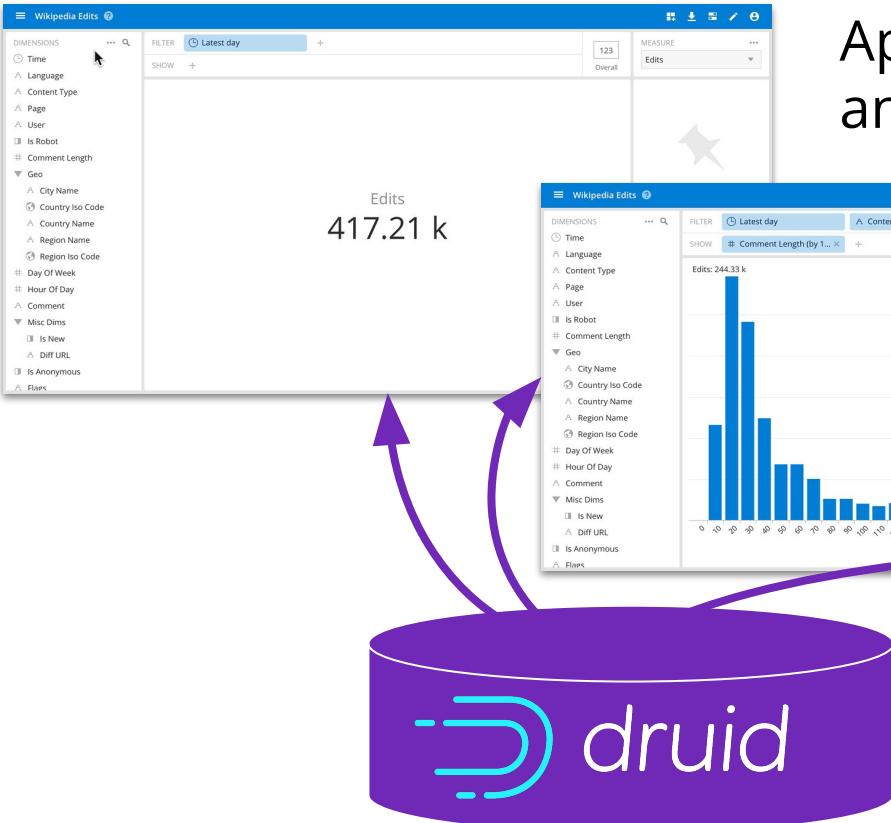
Netflix 의 Druid 활용 사례



“Netflix의 엔지니어들은 콘텐츠에서 나오는 다양한 데이터를 처리하는데 Druid를 적극적으로 활용합니다. 데이터는 시간당 2TB에 이르며, 실시간으로 집계하고 조회가 가능하도록 구현되어 있습니다. 서비스의 이상 감지, 콘텐츠 제공 현황, 사용자 활동 기록 등을 모두 이 방식으로 처리합니다.”



Apache Druid drives interactive analytics at any scale.



⌚ 0.1-3s query

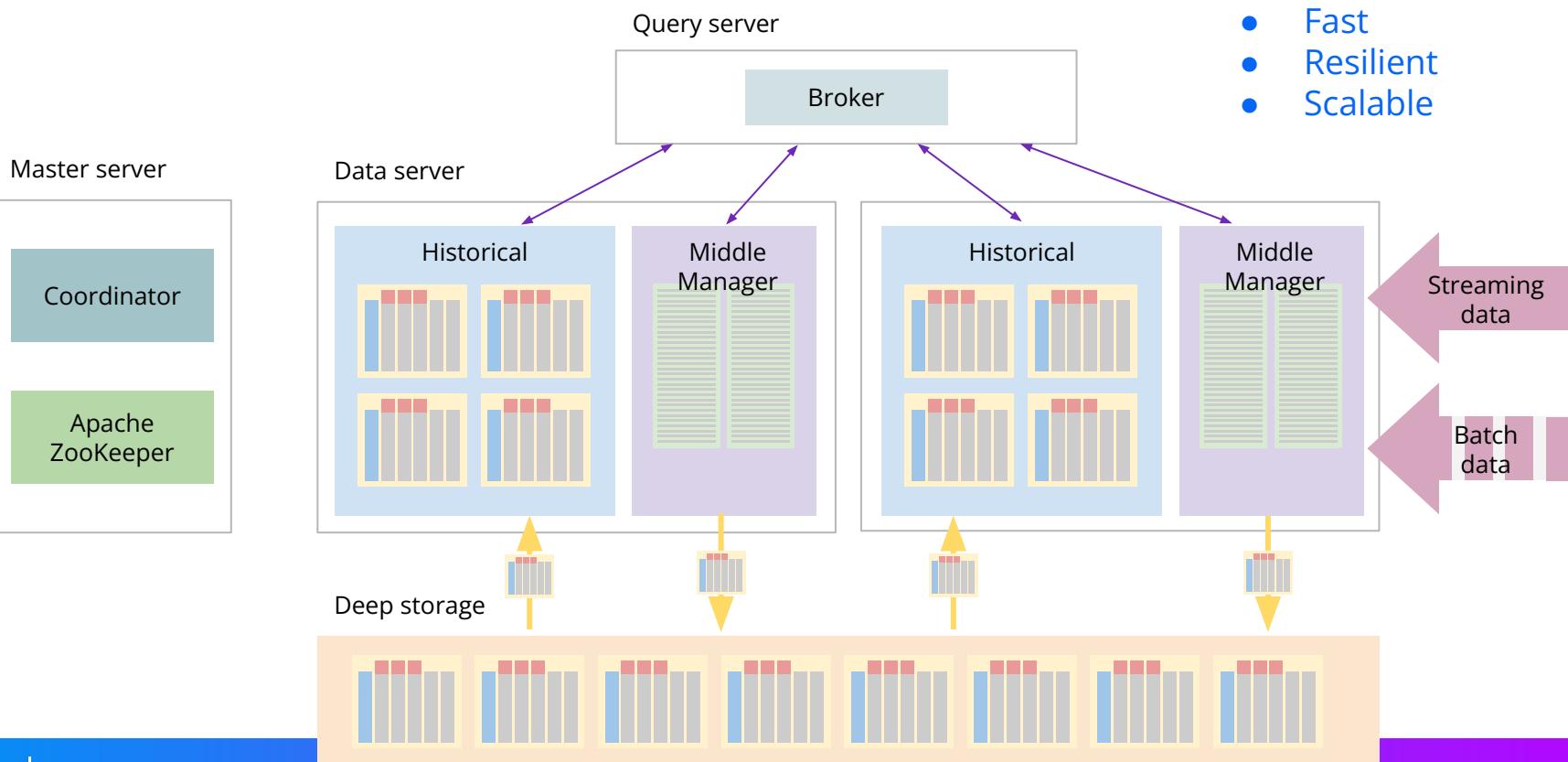
🕒 fresh data

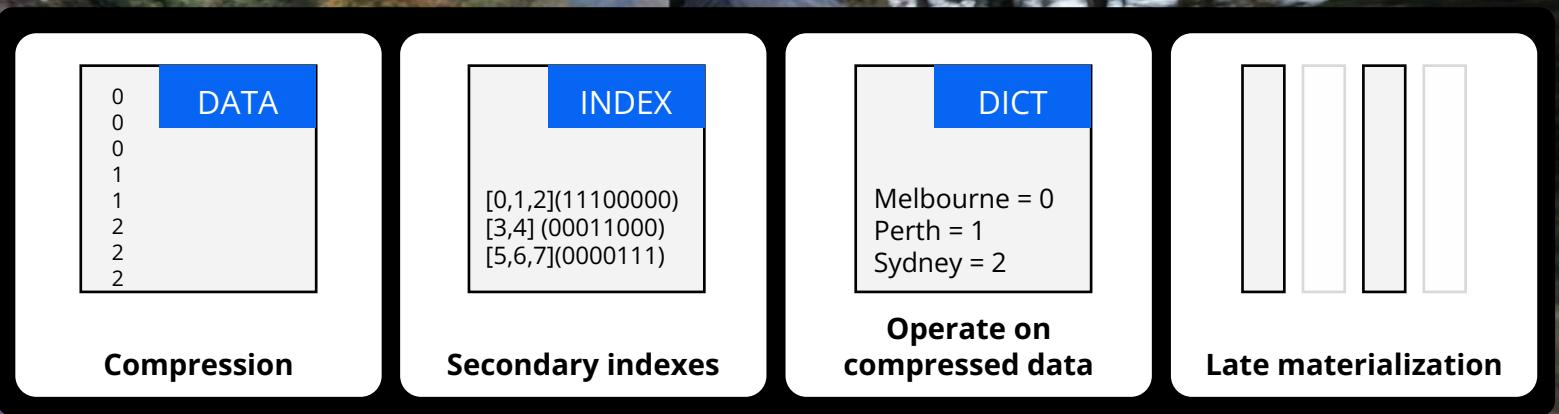
🏋️ high concurrency

🚴‍♂️ highly interactive

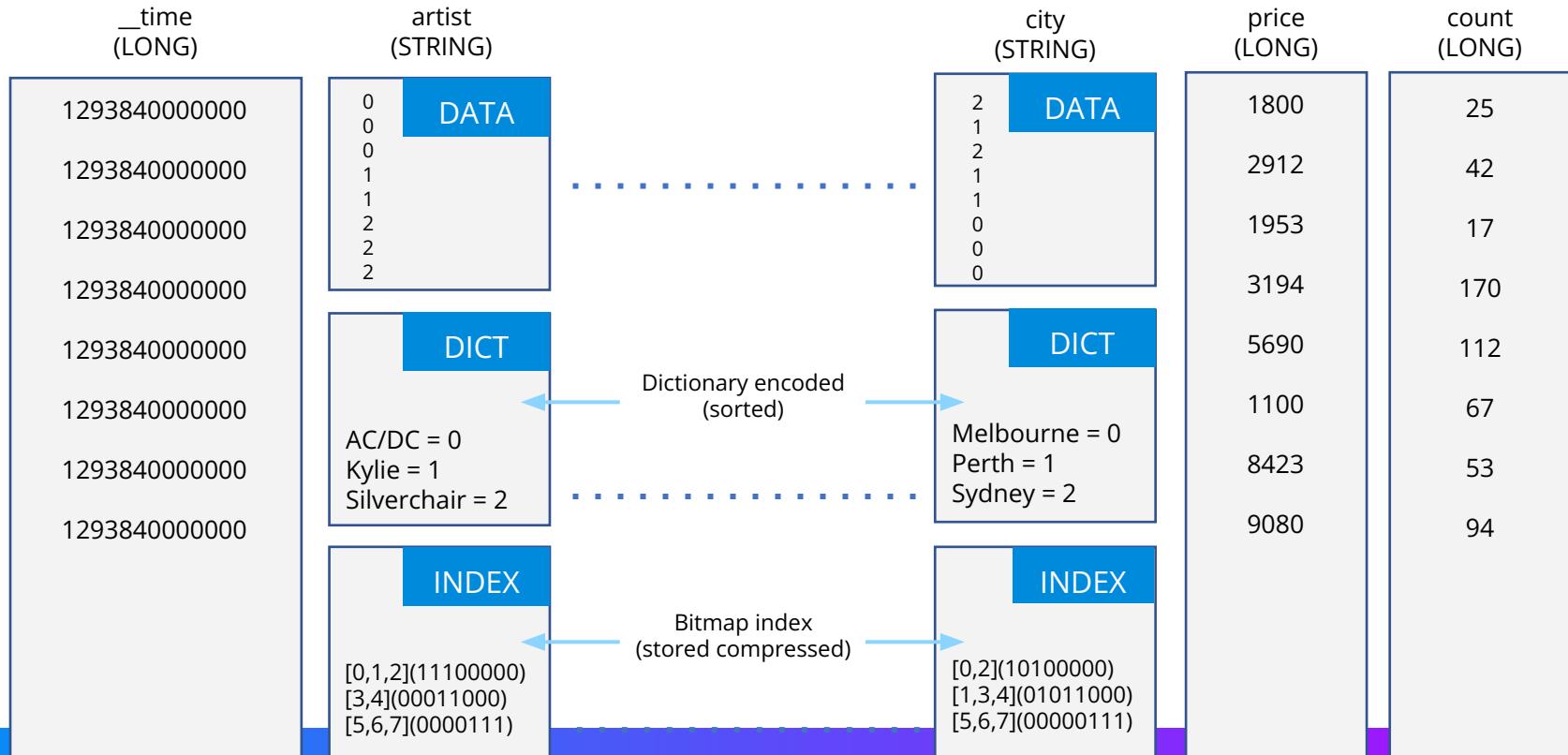
Apache Druid 아키텍처

Druid uses a microservice architecture

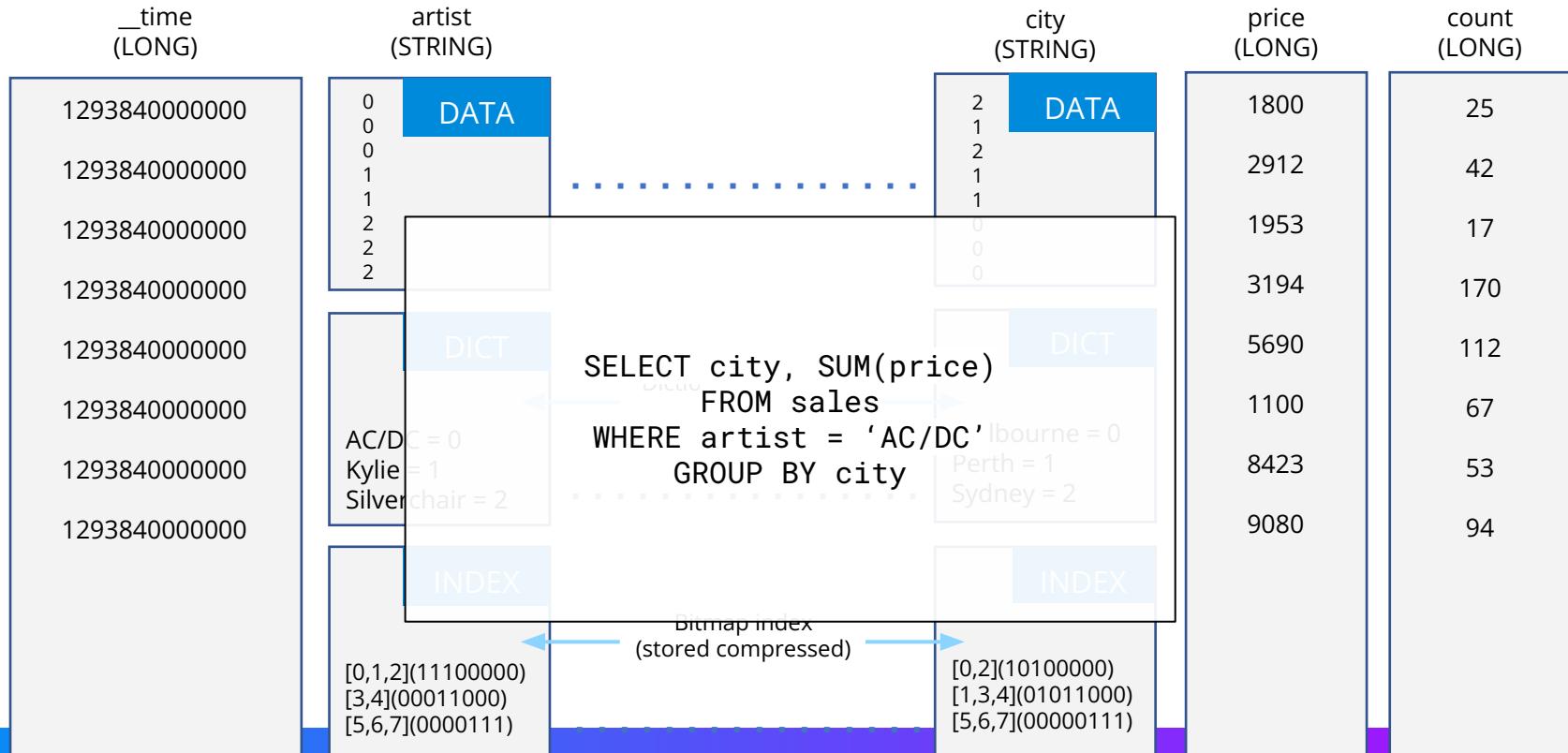




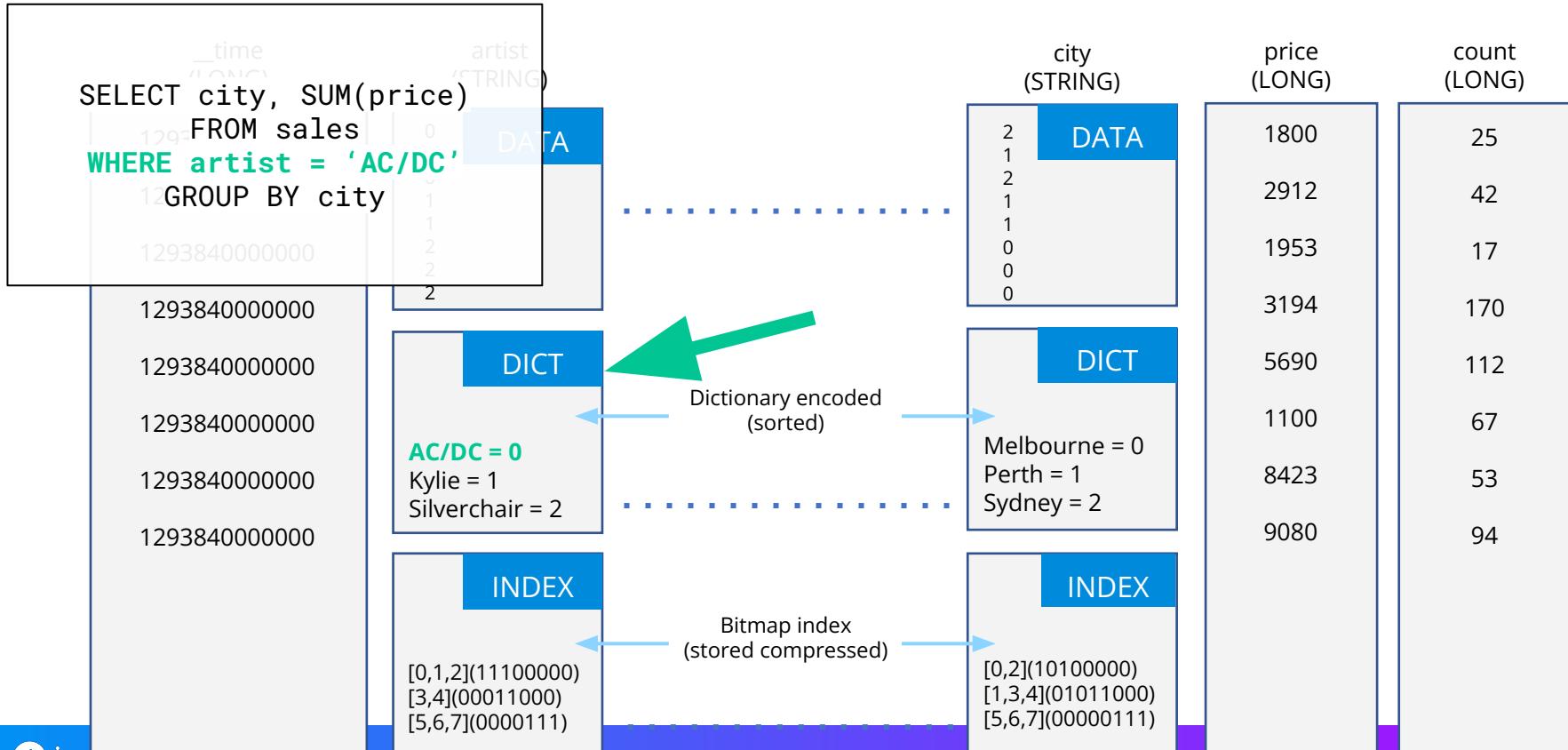
Engine and data format are tightly integrated



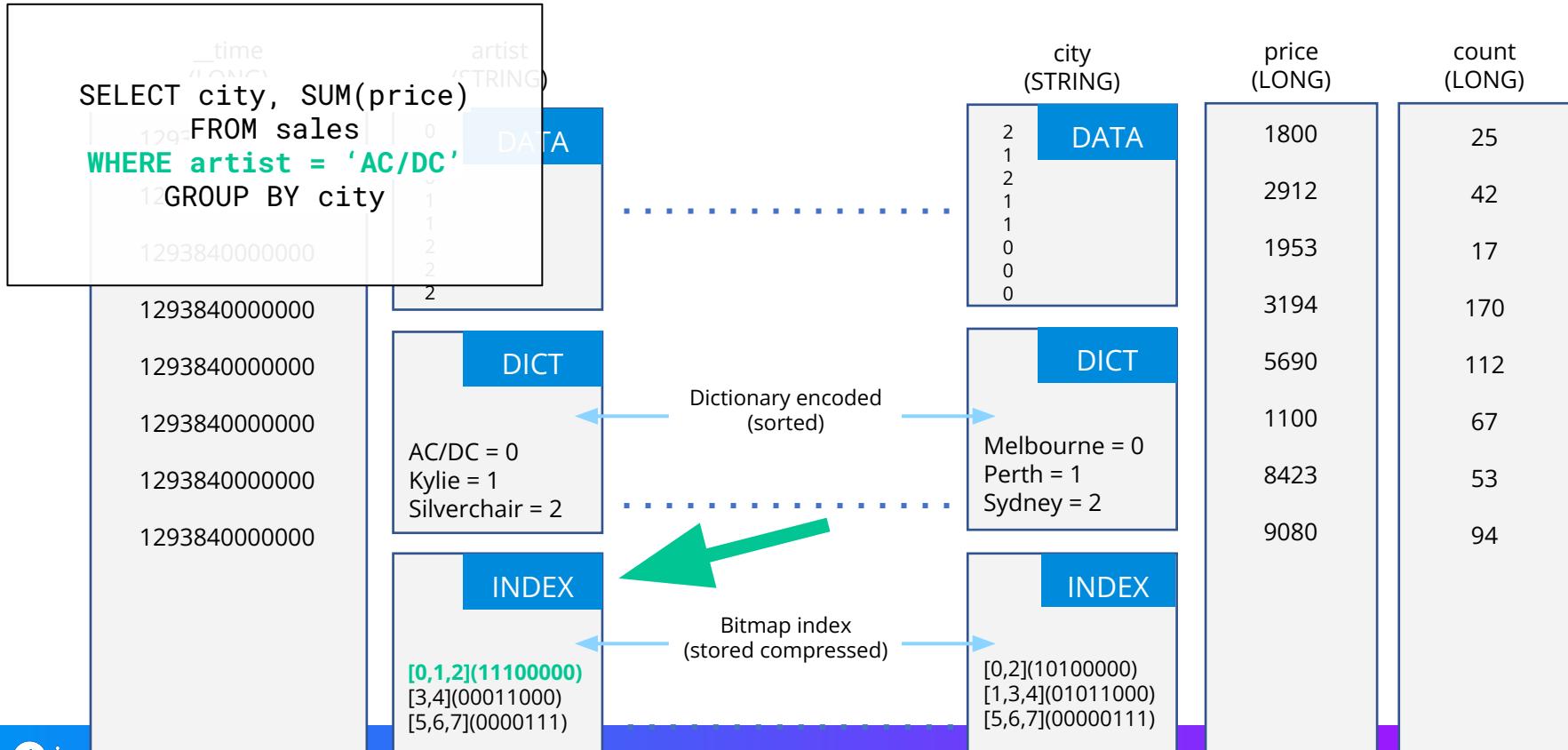
Engine and data format are tightly integrated



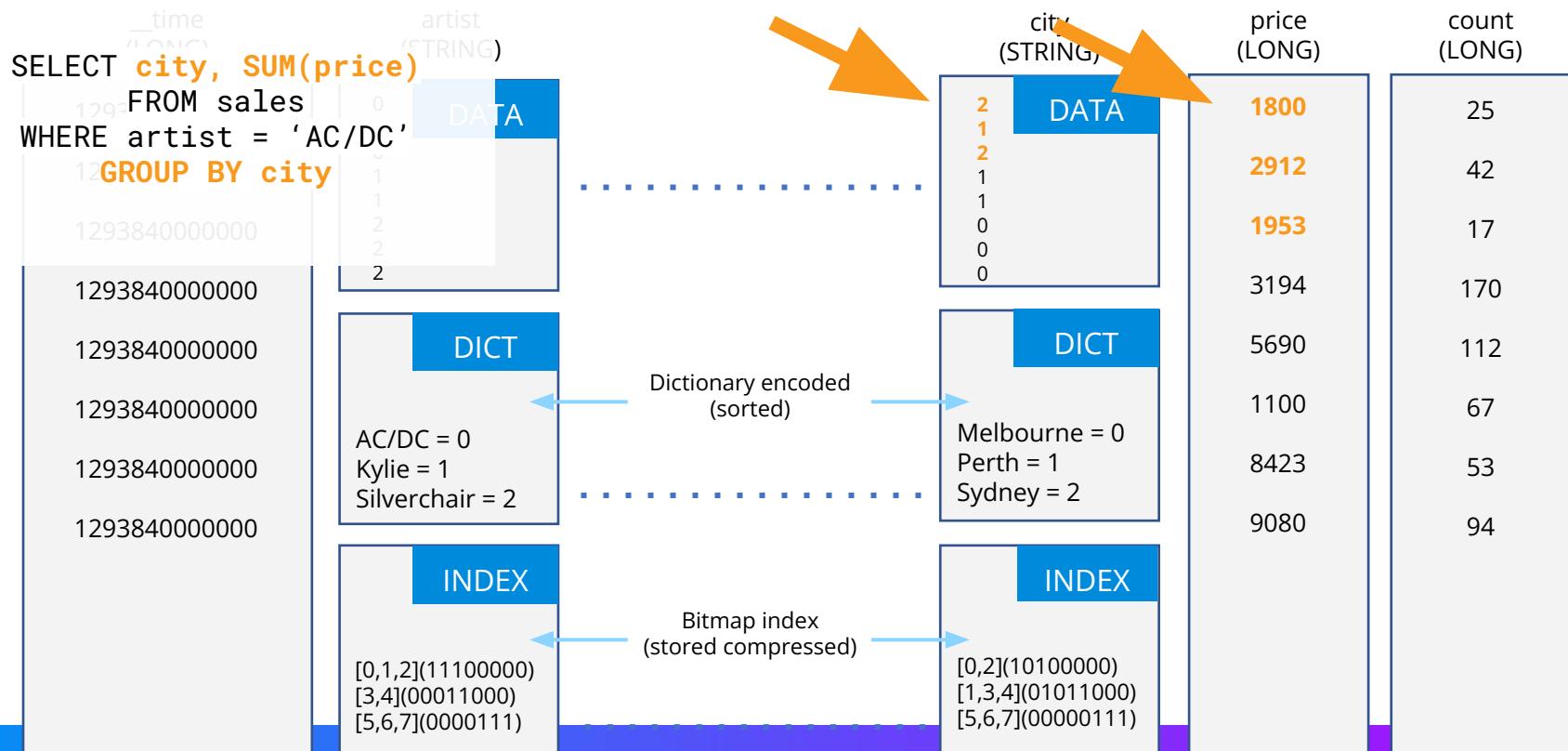
Engine and data format are tightly integrated



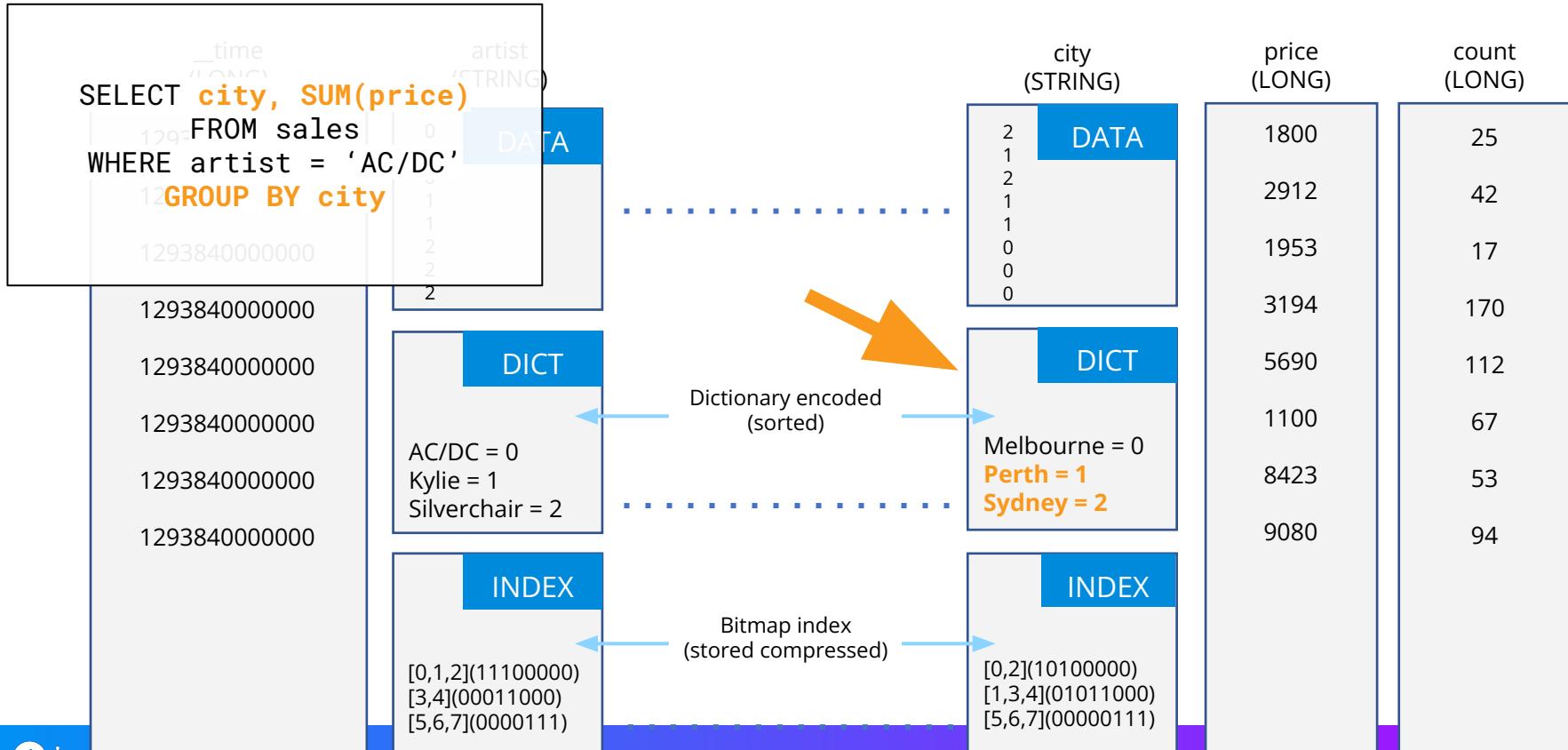
Engine and data format are tightly integrated



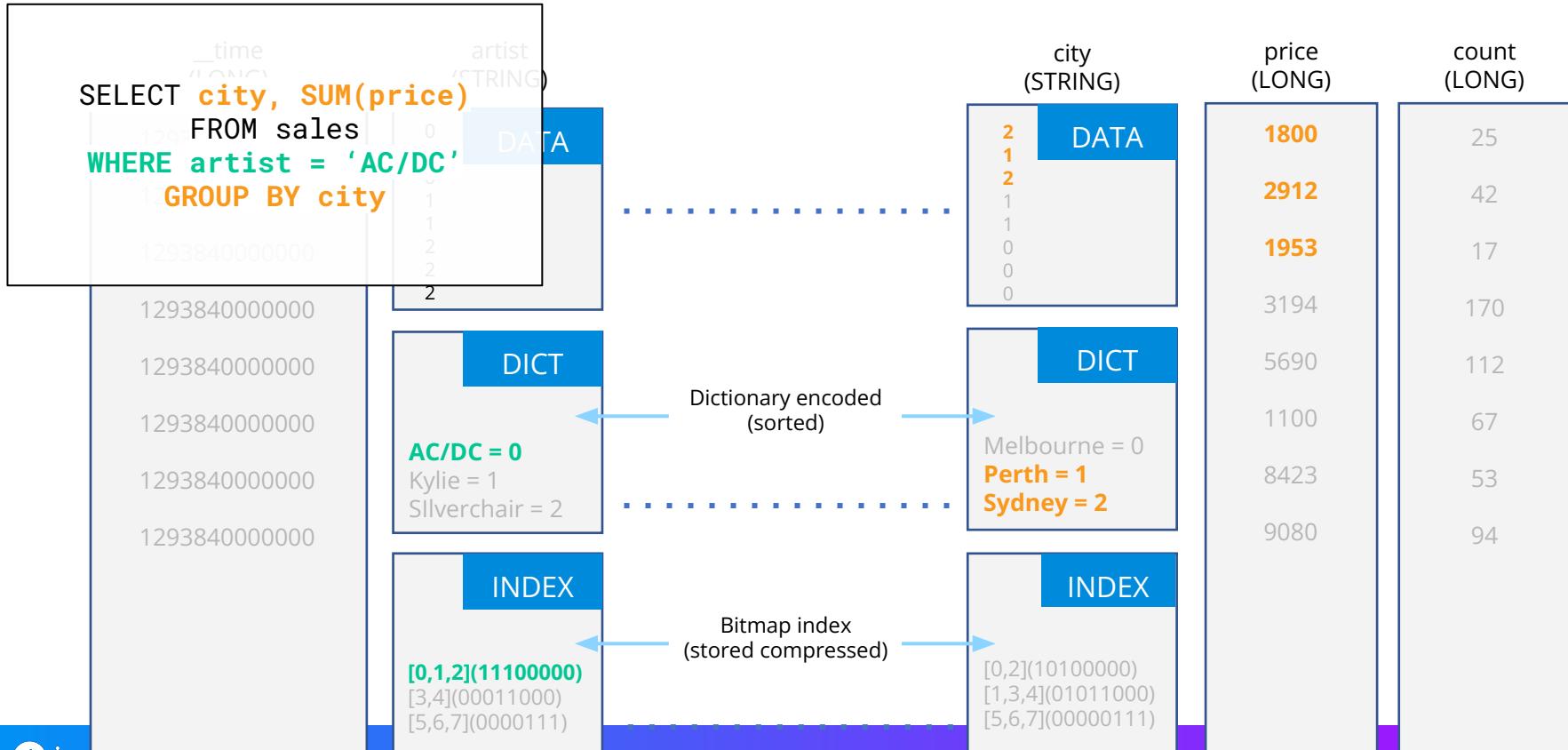
Engine and data format are tightly integrated



Engine and data format are tightly integrated



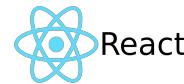
Engine and data format are tightly integrated



DECISIONS

DEVELOPMENT

DATA SCIENCE



 Superset



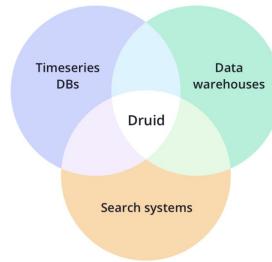
The Druid logo features a large, stylized, light blue "D" icon composed of several curved lines, followed by the word "druid" in a lowercase sans-serif font with a registered trademark symbol (®) at the end.

Druid 와 Imply

Druid / Imply 소개

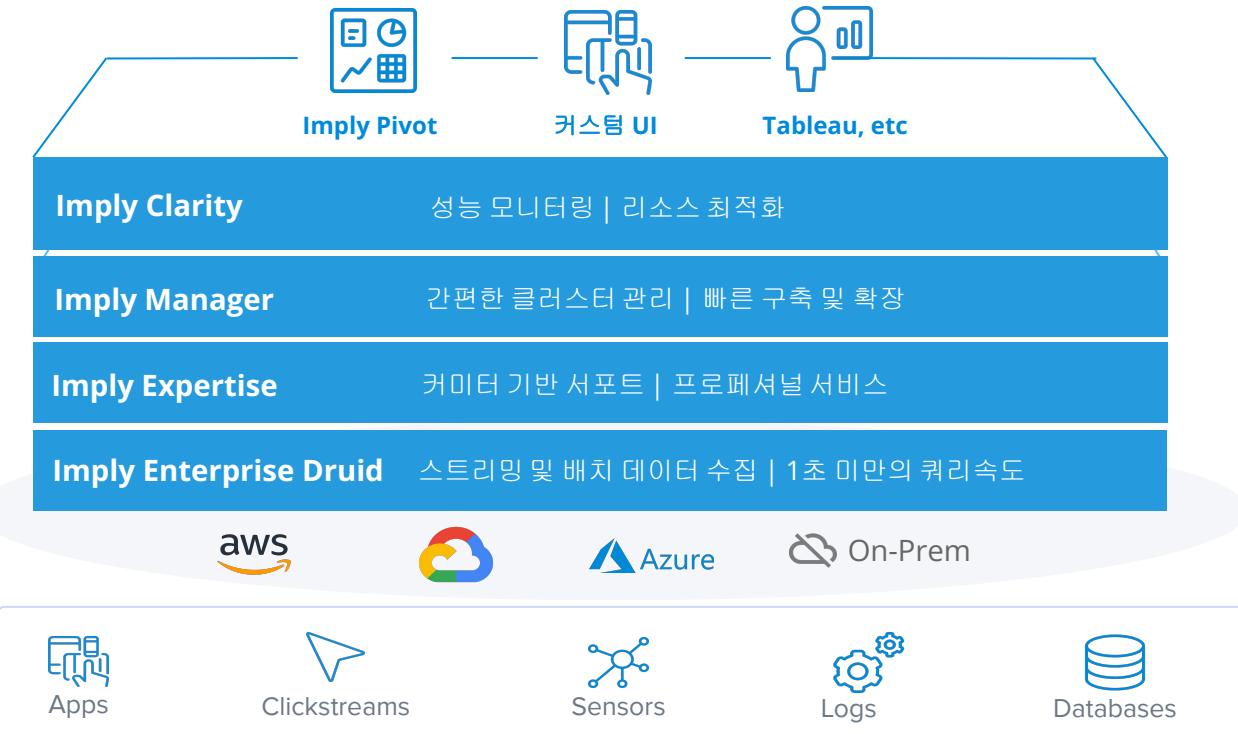


- 2010년 Apache 프로젝트로 시작
- 시계열 데이터에 대한 효과적인 저장 및 집계를 실시간으로 지원하기 위해 설계
- 스트리밍, 펀테크, 광고 등의 Digital Native 회사들이 다수가 참여



- 2015년 Druid 개발자들이 설립한 Series D 단계의 회사
- Apache Druid 기반의 데이터 분석 플랫폼 제공
 - 엔터프라이즈 기능 (성능 개선, 보안 기술)
 - 관리 편의성, 모니터링
 - 시각화
- 기존 Druid 고객 중 다수가 Imply 고객으로 전환

Imply 플랫폼



Imply 는 Druid를 위한 새로운 백본을 제공합니다

오픈소스 Druid



커뮤니티 서포트

오픈소스 엔진 - 6개월마다
업데이트

기본 커넥터



확장성

서포트

관리 및 보안

데이터
시각화

코어 엔진

연동성

최초 구축 지원 및 정기적인 헬스체크 + 베스트
프랙티스

연중무휴 세계적 수준의 서포트,
프로페셔널 서비스, 커스터머 섹세스, 엔지니어링

클라우드 매니지드 서비스, OOTB 모니터링,
셀프 힐링, 엔터프라이즈급 보안

Imply Pivot: 빠른 답변 및 기본 대화형
탐색 OOTB

Druid창시자가 제공하는 월간 릴리스, 고유 기능
및 더 빠른 수정 사항 제공

커넥터, 데이터 모델링 및 데이터 수집에 대한
공식 지원



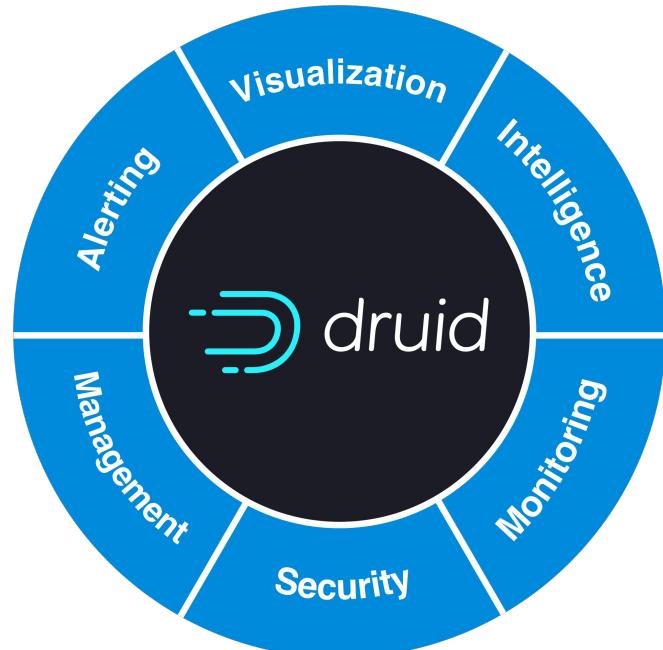
Imply는 Druid를 엔터프라이즈 솔루션으로 완성합니다

사용자가 직접 활용 가능한 선도적인 실시간

빅데이터 분석 플랫폼의 실현

Apache Druid 위에 제공하는 가치

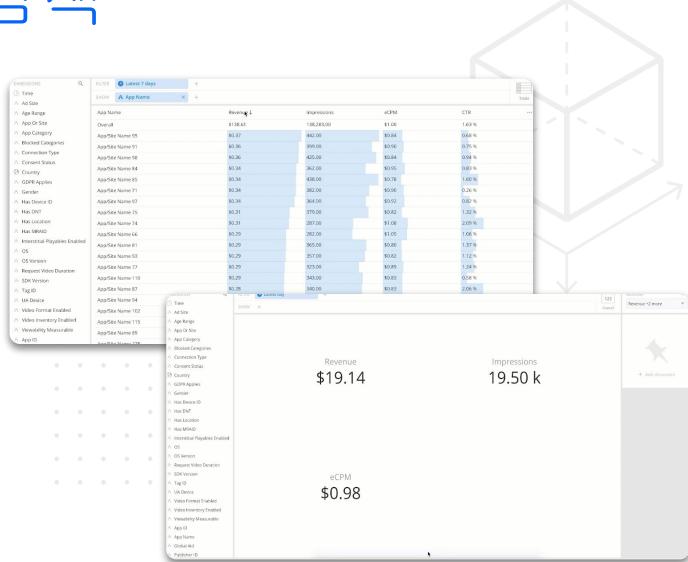
- **Imply Pivot:** 실시간 분석의 완성 (시각화)
- **Imply Manager:** 클러스터 배포 및 관리 도구
- **Imply Clarity:** 실시간 성능 모니터링 및 튜닝
- **Imply Security:** RBAC, 3rd Party 연계, 컴플라이언스
- **Global Support:** 전문가들의 기술 질의응답 및 요구사항에 대한 다양한 협업과 의견 제시
- **Professional Services:** 고객사 전담 인력 배치를 통한 맞춤형 기술 및 구축 지원 수행



Imply Pivot을 활용한 빠른 데이터 탐색

Druid 전용 시각화 기반 UI를 통한 Time-to-market 극대화

- 컴플라이언스 하에서 개인화 된 경험 제공
- 스트리밍과 과거 데이터가 통합된 화면 제공
- 드래그 앤드롭과 슬라이스 등 상호작용이 가능한 시각화 도구
- 정보수집 및 탐색을 위한 실시간 대시보드
- 보안, 사용자 관리, 알람 등의 엔터프라이즈 기능 통합

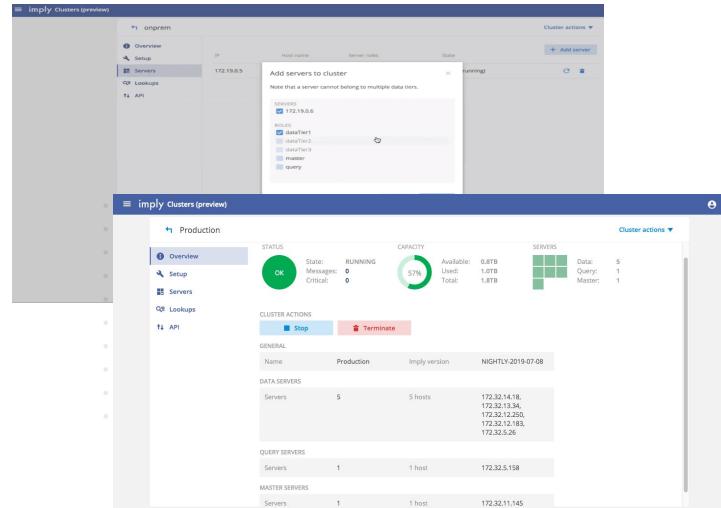


대용량 데이터에 대한 응답성 높은 데이터 탐색

Imply Manager를 통한 직관적인 관리

직관적으로 활용 가능한 Druid 클러스터 통합 관리 도구

- **클릭을 통한** Druid의 배포, 업그레이드 등 작업 수행
- **운영에 영향을 최소화 하는** 새로운 Extension 및 변경에 대한 설정 값 적용
- 중요 클러스터 정보에 대한 **대시보드 및 알람 기능** 수행
- **다운타임 없는** 클러스터 업그레이드 제공

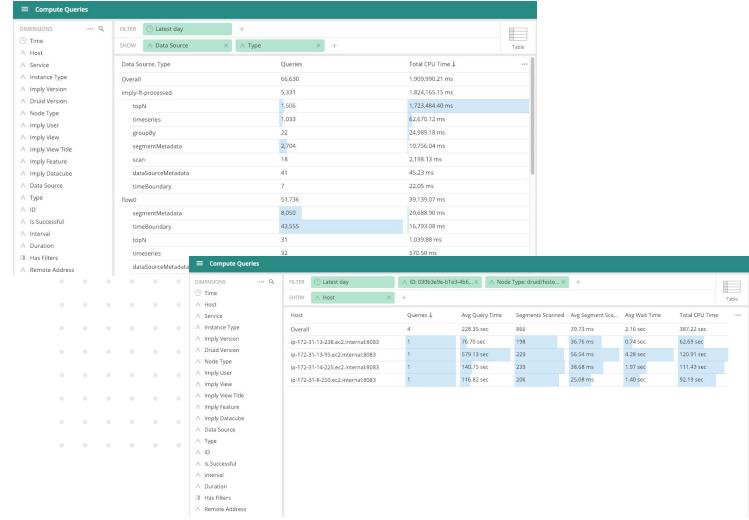


Druid 클러스터 관리에 대한 완성

Imply Clarity를 통한 Druid 클러스터 모니터링

Druid 클러스터 모니터링을 통한 성능 최적화 수행

- 성능 저하에 영향을 주는 **병목을 식별하고, 탐색하여, 원인을 제거할 수 있는 다양한 정보 제공**
- 클러스터 상태에 대해 다각도의 측면에서 탐색하고 시각적으로 확인하도록 하여 **종합적인 상황을 이해**
- 사용량 증가에 따른 **자동화 된 확장성 제공**
- 사용 패턴의 시각화를 통한 엔드 유저의 **경험치 확장**



대규모의 환경에서 고성능을 위한 Druid 클러스터의 상태 모니터링

Imply Security를 통한 엔터프라이즈급 보안 달성

Druid 보안을 위한 선제적인 기능 추가 및 제공

- 발생하는 보안 이슈들에 빠르게 대응하여 제품에 적용
- Imply Manager와 통합한 RBAC 기능 제공
- 보안 관련 기능의 우선적인 추가 (Row Level Security 등)
- 3rd Party 보안 솔루션과의 연계 기능 제공

Name	Users	Permissions	Filter tokens
User	26	7	0
Super Admin	48	All	0
Read only	3	2	0

External role name: 0bb9
Role ID: 0bb9
Role visibility: Hidden
Member visibility: Hidden
Permissions: No perm

Filter token dialog fields:
Name: channelToken
Filter formula: \$channel == "#en.wikipedia"
Combinability: Fail if multiple different tokens
Buttons: Cancel, Save

Druid 내 중요 데이터에 대한 보안 기능 제공

Global Support 와 Professional Service를 통한 최고의 Druid 경험을 제공

지속적인 Global Support를 통한 최적화 방안 제공

- 정기 Cadence Call을 통한 고객 사용 현황 관리
- 중요 작업에 진행시 Imply 엔지니어와 의견 교환을 통한 적합한 수행 방안 도출
- 버그, 장애 등 운영시 문제 상황에 대한 기술 지원
- 신규 기능 등에 대한 Knowledge Base 제공과 필요시 협의에 따른 기술교육 수행

특정 요구사항 달성을 위한 Professional Service 수행

- Imply 전문가와 함께 프로젝트 초기 단계부터 프로세스 설계
- Imply 도입 고객 사례를 바탕으로 한 효과적인 Best Practice 제공
- 지속적인 고도화 방안 및 효과적인 활용 방안 논의

높은 수준의 Druid 활용성 확보를 위한 전문 기술인력들의 지속적인 지원

Druid 대비 Imply 특장점



OSS 기반 운영환경 적용의 리스크 존재	통상 3~6배의 빠른 Time-to-Market 제공
최적의 튜닝을 위한 다수의 사전 작업이 필요	예측 가능한 성능 지표 및 튜닝에 필요한 지표 제공 튜닝에 대한 방향성 및 경험치 공유 (워크샵 등)
다운타임의 발생 가능성 존재	99.999%의 Uptime 제공 (* Hybrid, Polaris 사용시 SLA로 보장)
관리 운영을 위한 스크립트 작성 필요	운영에 필요한 효과적인 도구들을 (Clarity, Manager) 통한 4배 이상의 생산성 제공
기술지원(Commercial Support) 없음	전담 24x7 기술지원 제공
통상 2~3배 이상의 인프라 및 운영인력 비용 소요	아키텍처 최적화를 통한 낮은 TCO 확보
최소 6개월 이상의 신규 기능 정식 추가	통상 OSS대비 3배 이상 빠른 신규 기능 추가

Apache Druid 최신 기능

오픈소스 출시 이후, Druid의 주요 마일스톤

2013: 커뮤니티 라이센스 출시

2013: Druid 0.3 출시

2015: Apache license로 채택

2017: SQL (Apache Calcite)지원 추가

2019: APACHE 최고 수준의 프로젝트 달성

2019: 종속성 없는 (dependency-free) 배치 기능

2020: 벡터 쿼리 엔진

2020: 보안 레이어 추가

2021: 1000개 이상의 사용 고객 발굴

분석 환경을 더욱 지원하기 위한 기능 개선 사항들

2022년에 확장된 기능들

높은 응답성

PB+ 데이터에 대한 수초 이내 응답
많은 동시 쿼리에 대한 빠른 수행
스트리밍과 배치 처리



리포트

매우 큰 결과 세트
오래 수행되는 쿼리
높은 CPU를 사용하는 작업

오늘날 해결 방안

Druid의 극대화 된 튜닝
또는
다른 엔진에 듀얼 로딩

알림

수십개의 알림
대량의 오브젝트에 대한 추적
다양한 조건 수행

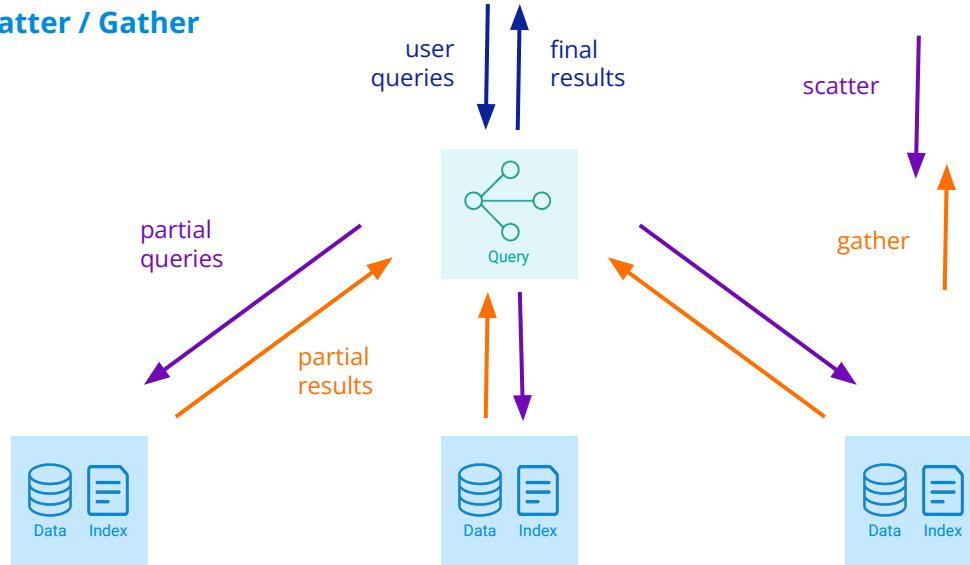
오늘날 해결 방안

Druid의 극대화 된 튜닝
또는
다른 엔진과 함께 프로세싱

Druid의 Core Query Engine

높은 동시성과 빠른 응답성을 요구하는 데에 효과적으로 설계

Scatter / Gather



- 필요없는 일은 하지 않는 형태로 동작
- 쿼리 엔진이 스토리지 포맷과 유기적으로 연계
- 효과적인 형태로 수행



Druid에 필요한 기능은 무엇인가?

리포트

매우 큰 결과 세트
오래 수행되는 쿼리
높은 CPU를 사용하는 작업
(빈번하지 않고 정형화 됨)

- 노드 간 데이터 셔플링이 발생
- 비동기적으로 긴 쿼리들이 수행
- 다른 워크로드에 영향을 주지 않고 많은 튜닝없이도 긴 시간 동안 결과값을 도출해 내는 쿼리 처리

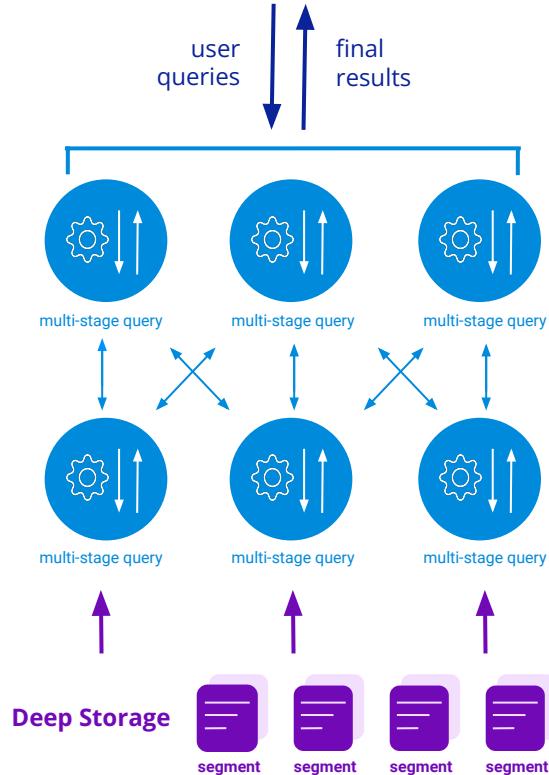
알림

수십개의 알림
대량의 오브젝트에 대한 추적
다양한 조건 수행

Multi-Stage Query Engine

새로운 엔진을 통해 다층의 쿼리로 수행

- 기존의 쿼리엔진의 코어를 더욱 활용
- 쿼리의 계층을 나눠 셤플링 수행
- 각 계층은 다수의 데이터 서버들이 병렬로 처리
- 자동화 된 튜닝
- 어느 규모에서도 확장성을 보유



Apache Druid 24.0

SQL-based ingestion

Before...

```
{  
  "type": "index_parallel",  
  "spec": {  
    "ioConfig": {  
      "type": "index_parallel",  
      "inputSource": {  
        "type": "http",  
        "uris": [  
          "https://druid.apache.org/data/wikipedia.js  
          on.gz"  
        ]  
      },  
      "inputFormat": {  
        "type": "json"  
      }  
    },  
    "dataSchema": {  
      "granularitySpec": {  
        "segmentGranularity": "day",  
        "queryGranularity": "minute",  
        "rollup": true  
      },  
      "dataSource": "target_table",  
      "timestampSpec": {  
        "column": "timestamp",  
        "format": "iso"  
      },  
      "transformSpec": {  
        "transforms": [  
          {  
            "type": "expression",  
            "name": "user",  
            "expression": "UPPER(\"user\")"  
          }  
        ]  
      }  
    }  
  }  
}  
  "filter": {  
    "type": "like",  
    "dimension": "channel",  
    "pattern": "%wikipedia"  
  },  
  "dimensionsSpec": {  
    "dimensions": [  
      "page",  
      "user"  
    ]  
  },  
  "metricsSpec": [  
    {  
      "name": "sum_added",  
      "type": "longSum",  
      "fieldName": "added"  
    },  
    {  
      "name": "sum_deleted",  
      "type": "longSum",  
      "fieldName": "deleted"  
    }  
  ],  
  "tuningConfig": {  
    "type": "index_parallel",  
    "partitionsSpec": {  
      "type": "single_dim",  
      "partitionDimension": "page"  
    },  
    "forceGuaranteedRollup": true  
  }  
}
```

After...

```
INSERT INTO target_table  
SELECT  
  FLOOR(TIME_PARSE("timestamp") TO MINUTE) AS __time,  
  "page",  
  UPPER("user") as "user",  
  SUM("added") as "sum_added",  
  SUM("deleted") as "sum_deleted"  
FROM  
  TABLE( EXTERN( ... ) )  
WHERE "channel" LIKE '%wikipedia'  
GROUP BY 1,2,3  
PARTITIONED BY DAY  
CLUSTERED BY page
```

Ingestion SQL statements

- Append to a Druid datasource:

```
INSERT INTO target_table  
    SELECT ...
```

- Replace data in a given time interval:

```
REPLACE INTO target_table OVERWRITE WHERE <condition on __time>  
    SELECT ...
```

- Test data sourcing query:

```
SELECT ...
```

SQL-based ingestion - table function EXTERN

```
SELECT
  *
FROM TABLE(
  EXTERN(
    '{'
      "type": "s3",
      "uris": [<source URIs>]
    },
    '{"type": "json"}',
    [
      {"name": "timestamp", "type": "string"},  

      {"name": "page",       "type": "long"},  

      ...
    ]
  )
)
```

The code snippet shows an SQL query using the `TABLE` function with an `EXTERN` table source. The configuration is defined in a JSON object. The structure is grouped by curly braces:

- A brace on the right groups the entire `EXTERN` block.
- A brace on the right groups the `uris` field.
- A brace on the right groups the `type: json` field.
- A brace on the right groups the entire schema definition under the `EXTERN` block.

[Druid Input Source](#)

[Druid Input Format](#)

[Input Row Schema](#)

SQL-based ingestion - INSERT / REPLACE

```
INSERT/REPLACE INTO target_table [OVERWRITE ...]  
SELECT  
    TIME_PARSE("timestamp") AS __time, ← Specify the __time dimension  
    "page",  
    UPPER("user") as "user",  
    "added",  
    "deleted"  
FROM  
    TABLE( EXTERN( ... ) )  
WHERE <input filtering conditions> ← Filter input (optional)  
PARTITIONED BY DAY ← Segment granularity  
CLUSTERED BY page ← Optional clustering dimensions
```

SQL expressions specify dimensions and metrics as input fields or SQL transformations

SQL-based ingestion - aggregation

```
REPLACE/INSERT INTO target_table
SELECT
    FLOOR(TIME_PARSE("timestamp") TO DAY) AS __time, ← Use FLOOR or CEIL to specify
    "page", Dimensions
    MV_TO_ARRAY("mv_column") AS mv_column, ← Use MV_TO_ARRAY for
    SUM("added") AS sum_added, multi-value columns
    APPROX_COUNT_DISTINCT_DS_HLL("user") AS unique_users ← Aggregate metrics
FROM
    TABLE( EXTERN( ... ) )
WHERE <input filtering conditions> ← Filter input (optional)
GROUP BY 1,2,3 ← Group by __time and dimensions
PARTITIONED BY MONTH ← Segment granularity
CLUSTERED BY page ← Optional Clustering Dimensions
```

SQL-based ingestion - lookup JOIN during ingestion

```
INSERT INTO wiki_stages  
  
WITH langs AS (  
    SELECT *  
    FROM TABLE( EXTERN( <lookup source>    ))  
,  
  
wiki AS (  
    SELECT *  
    FROM TABLE( EXTERN( <event source> ))  
)  
SELECT  
    FLOOR(TIME_PARSE(wiki."timestamp") TO HOUR) AS __time,  
    wiki.page,  
    wiki."country",  
    langs."language" AS "country_language",  
    SUM(wiki.added) sum_added,  
    SUM(wiki.deleted) sum_deleted  
FROM wiki  
LEFT JOIN langs ON wiki.country = langs.country  
GROUP BY 1,2,3,4  
PARTITIONED BY DAY  
CLUSTERED BY "page"
```

Lookups are broadcast to all workers

Source events are processed across all workers as long as source file count \geq workers

Equivalent to transformSpec lookup function

Join events with lookups

Apache Druid 24.0 - Nested JSON

Nested JSON - Why?

We often see that data is not flat, like this simplified **Order Line Item** record.

```
{  
  "Order": {  
    "Date": "2022-09-30",  
    "FirstName": "Jane",  
    "LastName": "Doe",  
    "Payment": { "Type": "PayPal", "AuthorizationID": "0123456789" }  
  },  
  "Item":  
  {  
    "Product": { "Name": "Flat Screen TV", "Brand": "Vizio", "Model": "VM432" },  
    "Quantity": 3,  
    "Price": 595  
  }  
}
```

... yes, it can be flattened, but that requires some work.

Nested JSON - Flattening for Ingestion

```
{  
  "Order": {  
    "Date": "2022-09-30",  
    "FirstName": "Jane",  
    "LastName": "Doe",  
    "Payment": {  
      "Type": "PayPal",  
      "AuthorizationID"  
    }  
  },  
  "Item": {  
    "Product": {  
      "Name": "Flat Screen TV",  
      "Brand": "Vizio",  
      "Model": "VM432"  
    },  
    "Quantity": 3,  
    "Price": 595  
  }  
}
```

Prior to Druid 24.0,
a static mapping was needed to
flatten with rules like:

“Order_Date” = “Order”.“Date”

**“Payment_AuthorizationID” =
“Order”.“Payment”.“AuthorizationID”**

...

→ “Order_Date”: “2022-09-30”,
→ “Order_FirstName”: “Jane”,
→ “Order_LastName”: “Doe”,
→ “Payment_Type”: “PayPal”,
→ “Payment_AuthorizationID”: “0123456789”,
→ “Product_Name”: “Flat Screen TV”,
→ “Product_Brand”: “Vizio”,
→ “Product_Model”: “VM432”
→ “Item_Quantity”: 3,
→ “Item_Price”: 595

Nested JSON - SQL Ingestion

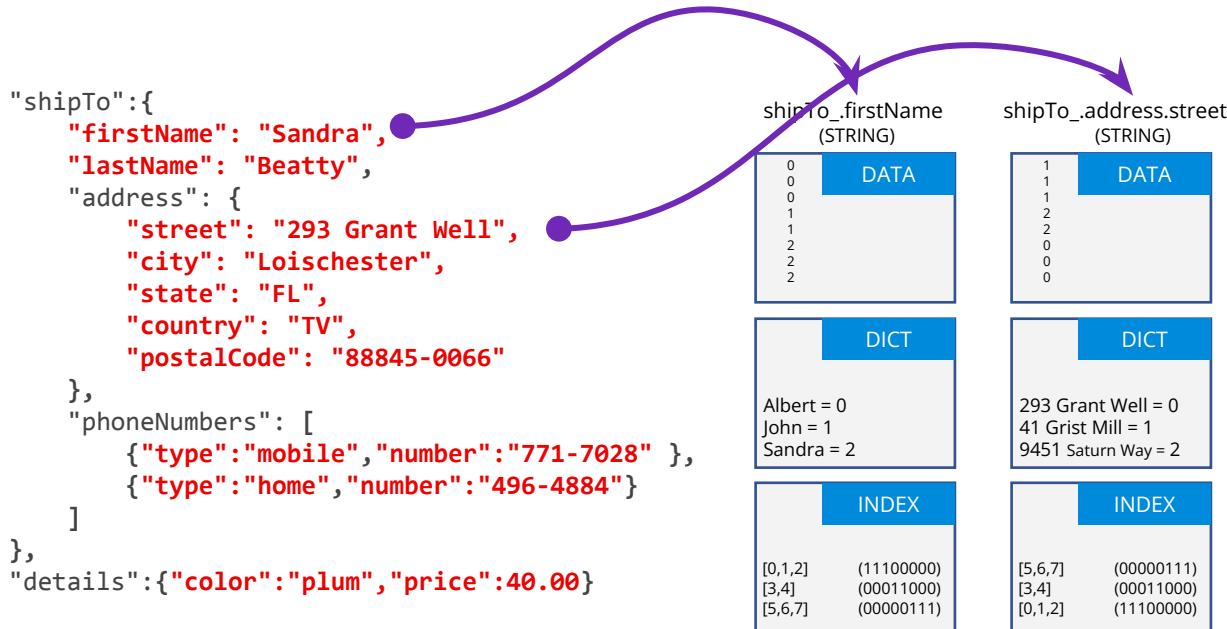
JSON Record

```
{  
    "time": "2022-6-14T10:32:08Z",  
    "product": "Keyboard",  
    "department": "Computers",  
    "shipTo": {  
        "firstName": "Sandra",  
        "lastName": "Beatty",  
        "address": {  
            "street": "293 Grant Well",  
            "city": "Loischester",  
            "state": "FL",  
            "country": "TV",  
            "postalCode": "88845-0066"  
        },  
        "phoneNumbers": [  
            {"type": "mobile", "number": "771-7028"},  
            {"type": "home", "number": "496-4884"}  
        ]  
    },  
    "details": {"color": "plum", "price": 40.00}  
}
```

Ingestion SQL

```
REPLACE INTO shipping_orders OVERWRITE ALL  
SELECT  
    TIME_PARSE("time") as __time,  
    product,  
    department,  
    shipTo,  
    details  
FROM  
    TABLE( EXTERN(  
        '{"type": "http", "uris": ["..."]}',  
        '{"type": "json"}',  
        '['  
            {"name": "time", "type": "string"},  
            {"name": "product", "type": "string"},  
            {"name": "department", "type": "string"},  
            {"name": "shipTo", "type": "COMPLEX<json>"},  
            {"name": "details", "type": "COMPLEX<json>"})  
        ]  
    )  
PARTITIONED BY DAY
```

Under the Hood - Storing and Indexing Nested JSON



Querying JSON - Discover JSON Structure

```
SELECT ARRAY_CONCAT_AGG( DISTINCT JSON_PATHS(shipTo))  
FROM shipping_orders
```

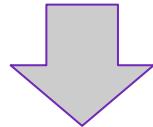


```
$.address.city  
$.address.country  
$.address.postalCode  
$.address.state  
$.address.street  
$.firstName  
$.lastName  
$.phoneNumbers[0].number  
$.phoneNumbers[0].type  
$.phoneNumbers[1].number  
$.phoneNumbers[1].type
```

Retrieve dynamic nested structure using **JSON_PATHS** to show existing and new fields in the data.

Querying JSON - Extract Fields

```
SELECT  
    product,  
    department,  
    JSON_VALUE(shipTo, '$.address.country') as country,  
    JSON_VALUE(shipTo, '$.phoneNumbers[0].number') as primaryPhone,  
    JSON_VALUE(details, '$.price') as price  
FROM shipping_orders
```



product	department	country	primaryPhone	price
Bike	Sports	BL	891-374-6188 x74568	542.0
Bike	Sports	ME	593.475.0449 x86733	955.0
Sausages	Grocery	AD	(904) 890-0696 x581	8.0
Mouse	Computers	CW	(689) 766-4272 x60778	90.0
Keyboard	Computers	TV	1-788-771-7028 x8627	40.0

Use **JSON_VALUE** in SELECT expressions to access any of the JSON fields.

Querying JSON - Filters and Aggregation

```
SELECT
    product,
    JSON_VALUE(shipTo, '$.address.country') as country,
    SUM(JSON_VALUE(details, '$.price' RETURNING BIGINT)) as total_revenue
FROM shipping_orders
WHERE JSON_VALUE(shipTo, '$.address.country') in ('BL',
GROUP BY 1,2
ORDER BY 3 DESC
```

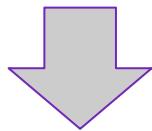


product	Country	total_revenue
Bike	BL	542
Mouse	CW	90

Use **JSON_VALUE** to GROUP BY any JSON field or to access metrics for aggregation.

Querying JSON - Filters and Aggregation

```
SELECT
    product,
    JSON_VALUE(shipTo, '$.address.country') as country,
    SUM(JSON_VALUE(details, '$.price' RETURNING BIGINT)) as total_revenue
FROM shipping_orders
WHERE JSON_VALUE(shipTo, '$.address.country') in ('BL', 'CW')
GROUP BY 1,2
ORDER BY 3 DESC
```

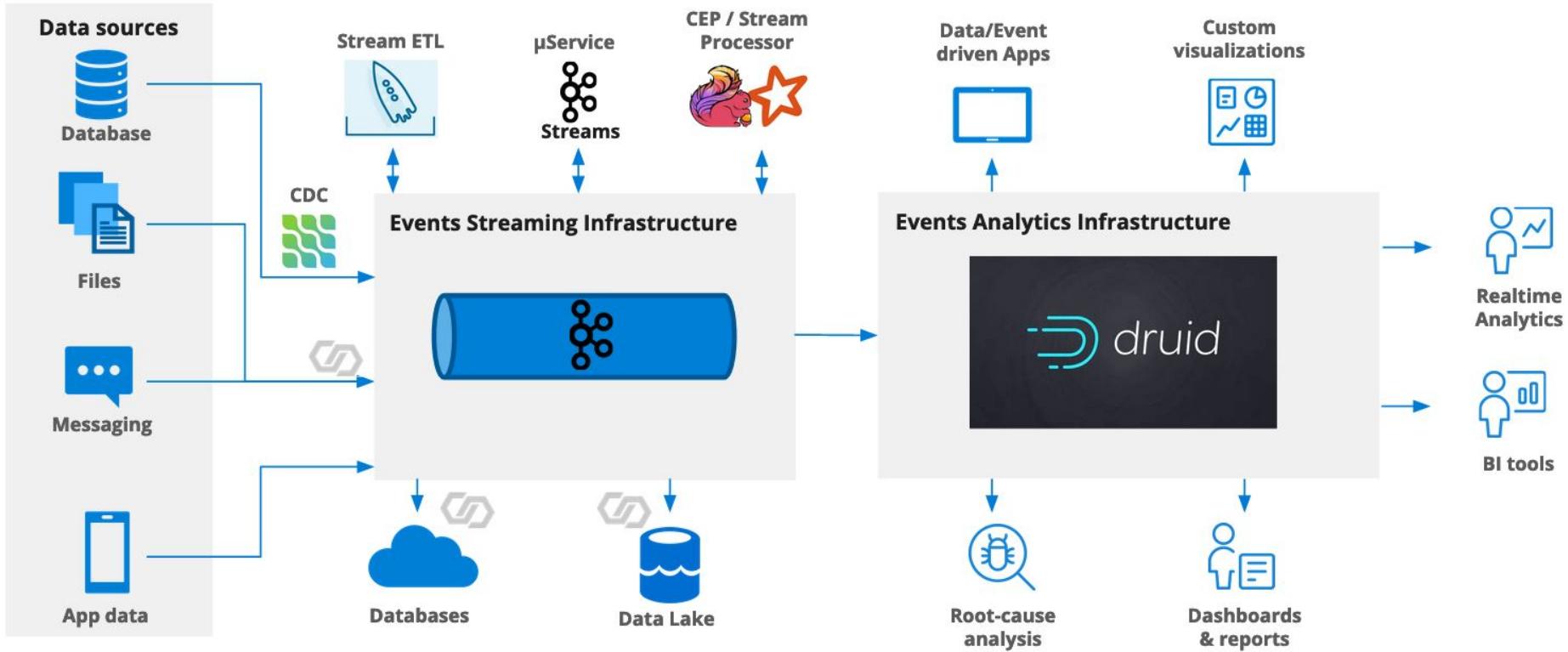


product	Country	total_revenue
Bike	BL	542
Mouse	CW	90

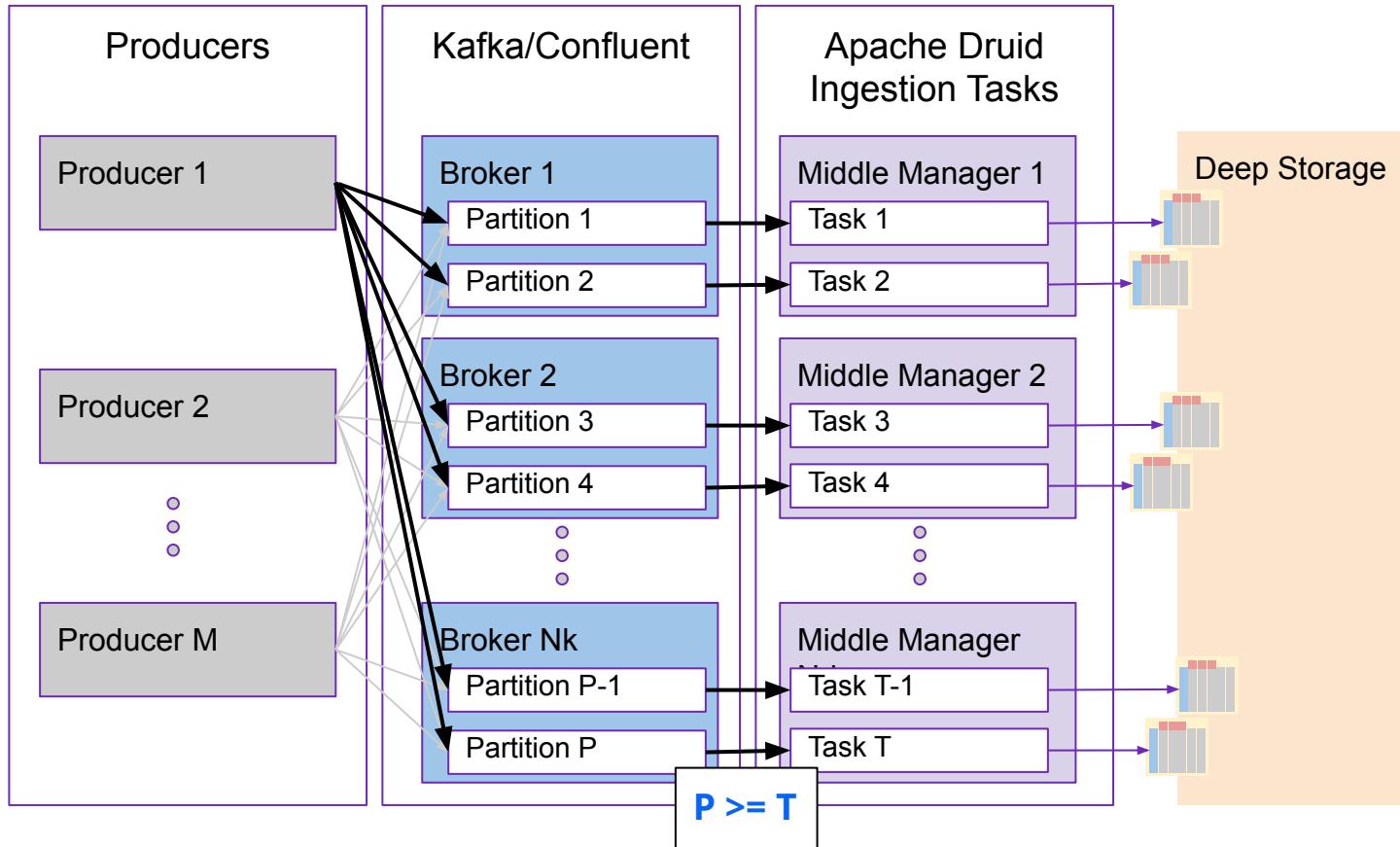
Use **JSON_VALUE** in WHERE clause to use indexed search.

K2D Stack

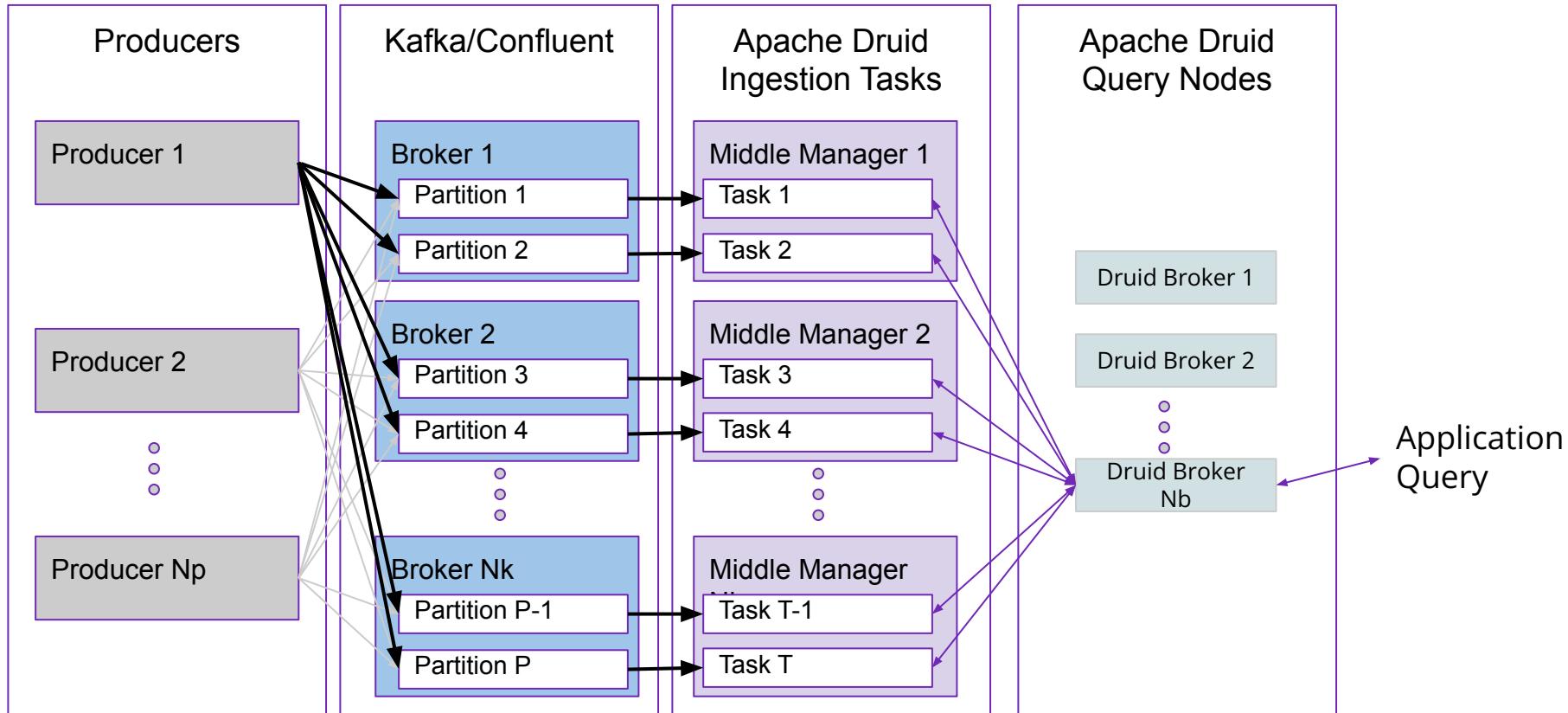
K2D Stack: Kafka to Druid



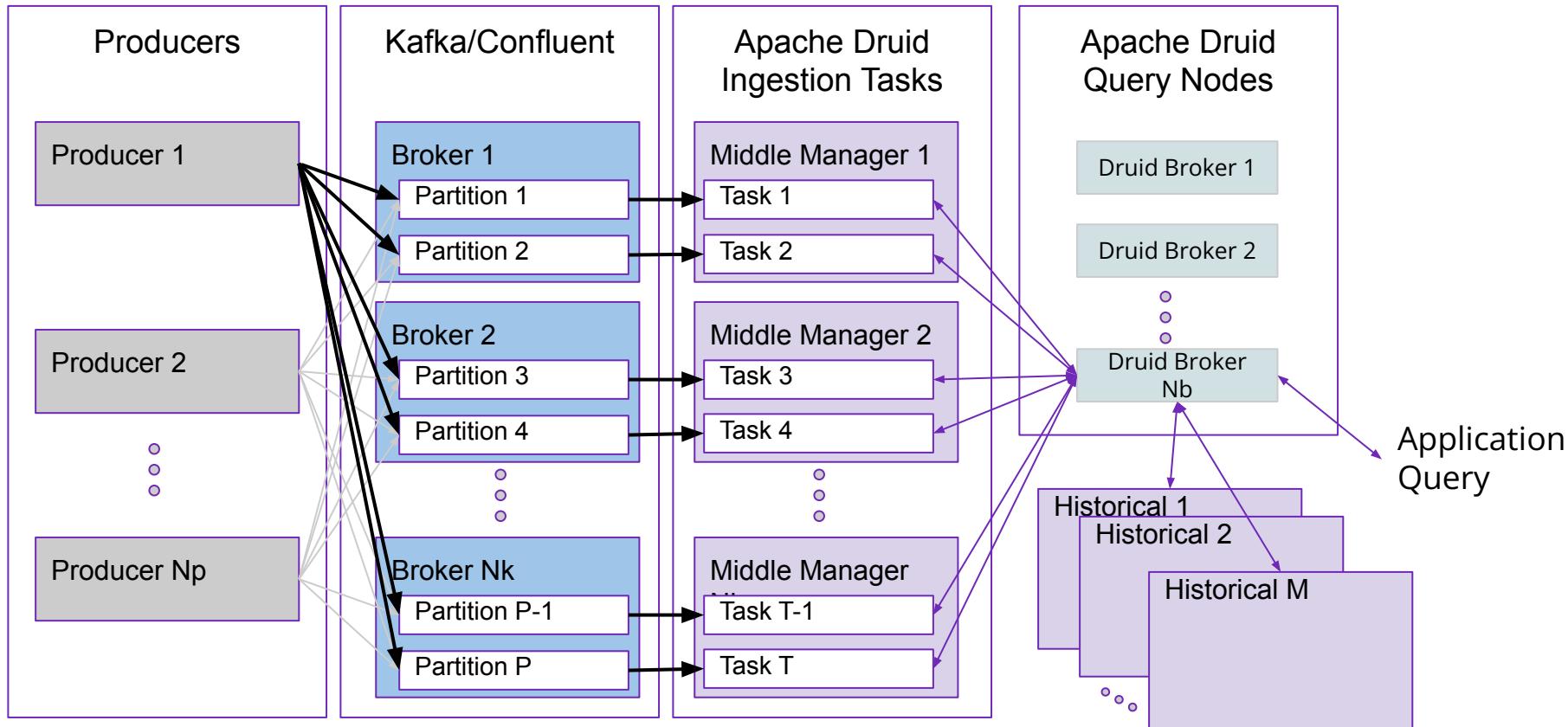
Scalable Data Ingestion



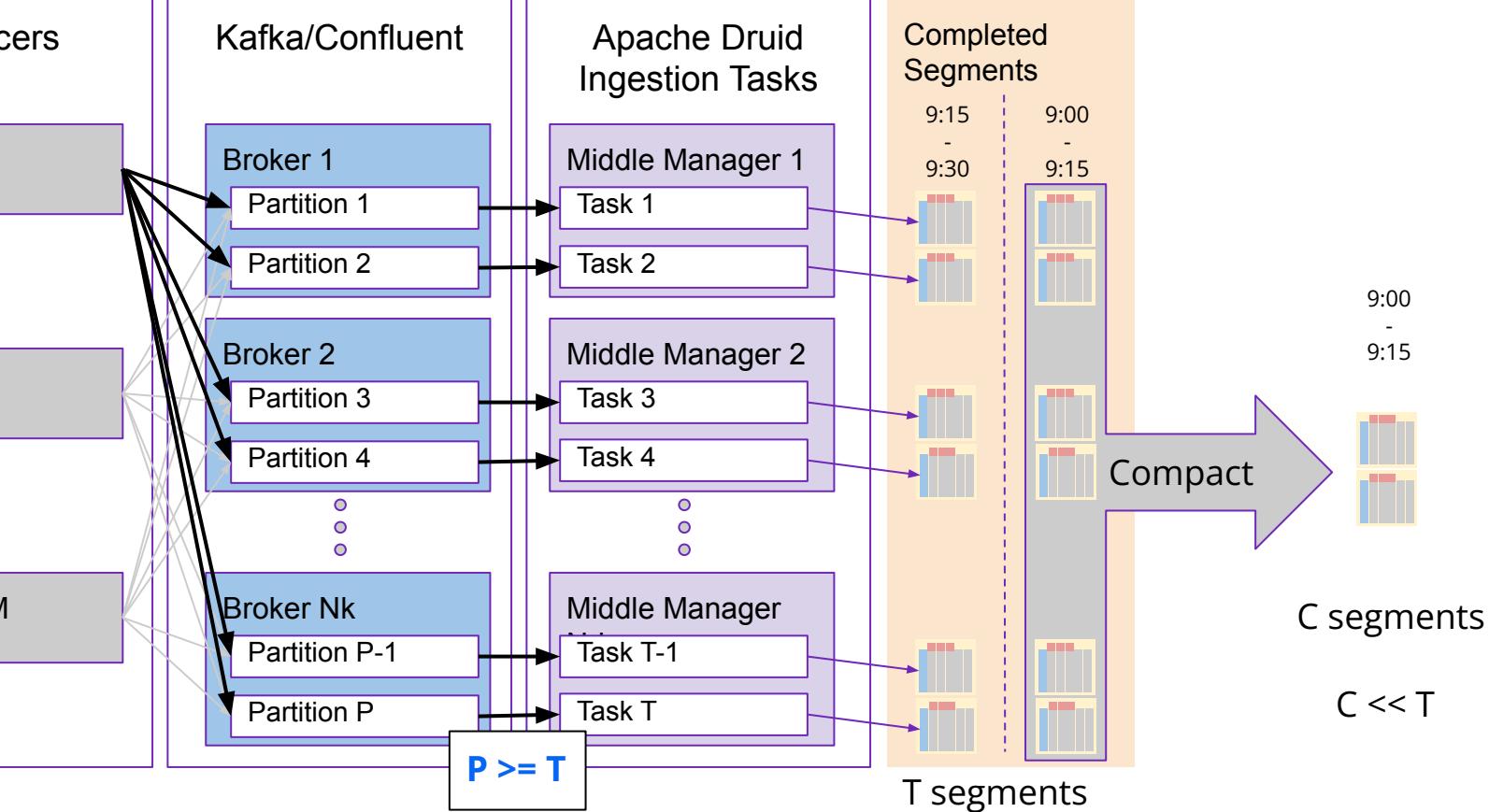
Scalable Real-time Queries



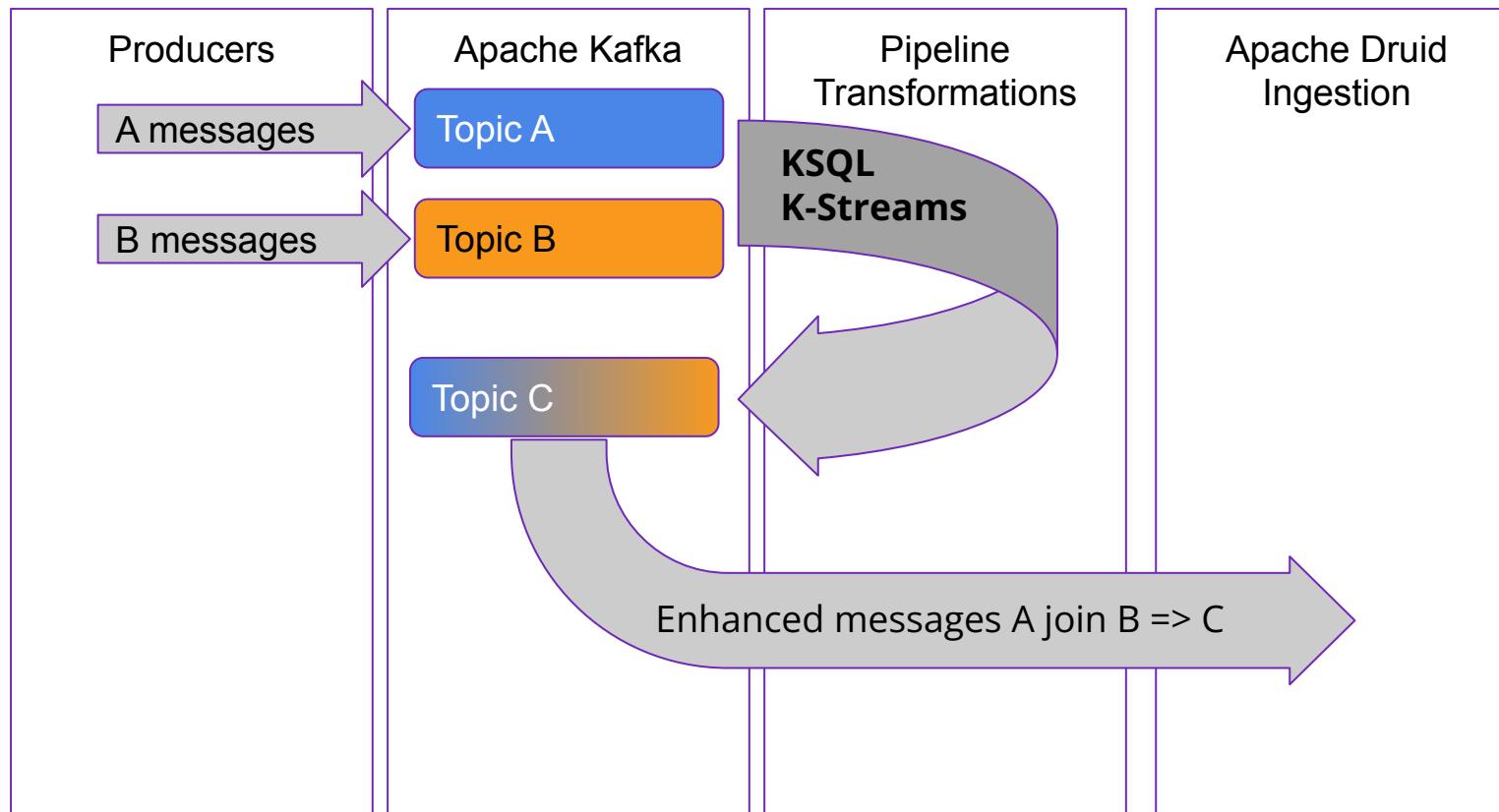
Scalable Real-time + Historical Queries



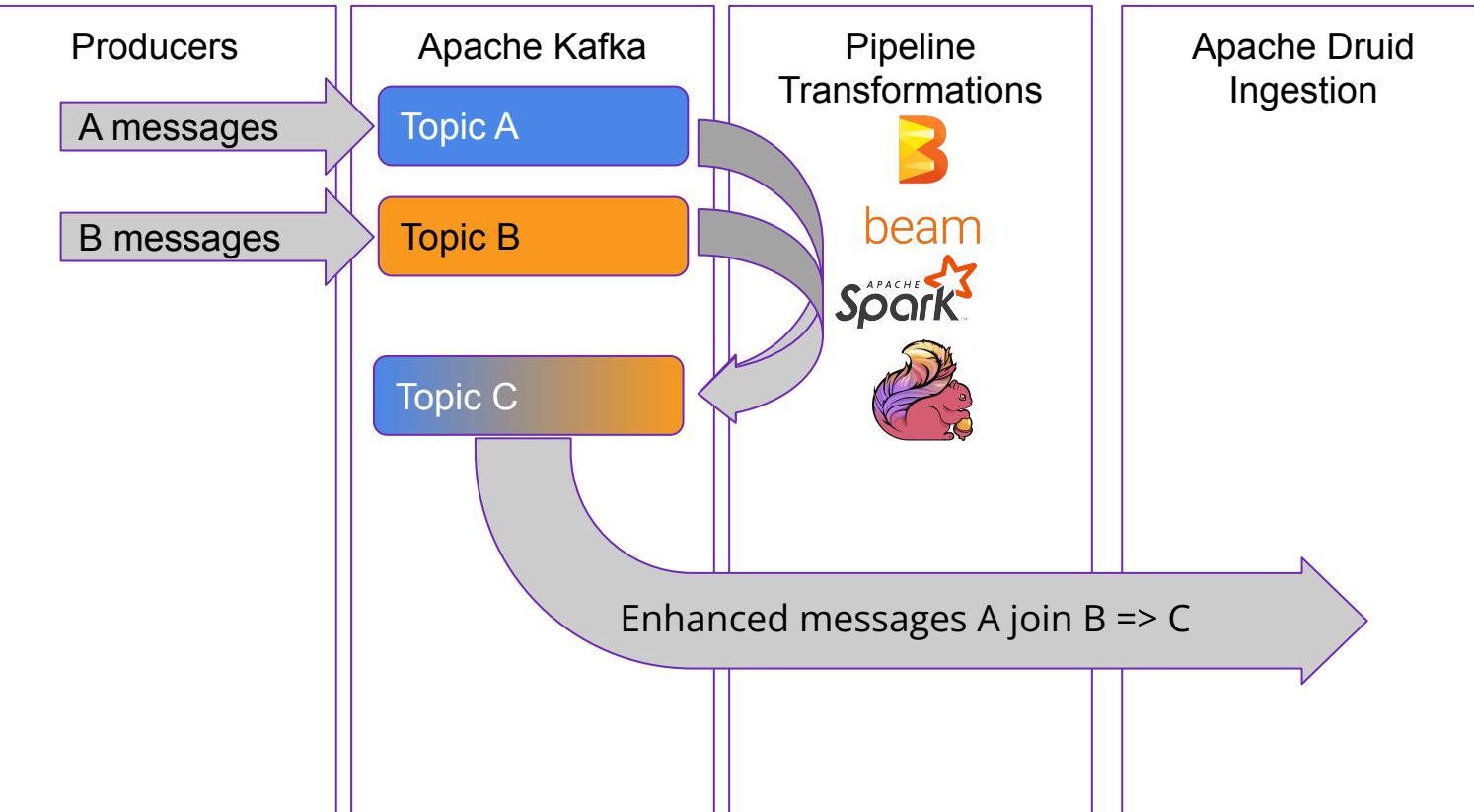
Best Practice: Scalable Data Ingestion + Autocompaction



Data Enhancement in the Pipeline - Kafka SQL



Data Enhancement in the Pipeline - Other Tech



Confluent Schema Registry 지원

```
"inputFormat": {
    "type": "avro stream",
    "binaryAsString": false,
    "avroBytesDecoder": {
        "type": "schema registry",
        "url": "<SCHEMA REGISTRY API ENDPOINT URL>",
        "config": {
            "basic.auth.credentials.source": "USER INFO",
            "basic.auth.user.info": "<SCHEMA REGISTRY API KEY>:<SCHEMA REGISTRY SECRET>"
        }
    }
}
```

Apache Druid - Confluent Cloud - localhost:8888/unified-console.html#load-data

Connect and parse raw data Transform data and configure schema Tune parameters Verify and submit

Start Connect Parse data Parse time Transform Filter Configure schema Partition Tune Publish Edit spec

Orchestrator ingests raw data and converts it into a custom schema format that is compatible with Apache Druid. To get started, please specify what data you want to ingest.

Learn more

Bootstrap servers: plc-druid-infra-east1.terraform.druid.cloud:9009

Type: Consumer properties

Where should the data be sampled from? Start on index: 0 Apply Next Parse data

remote_user	request	referrer	agent	bytes	ip	time	userid	jtime	status	
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	1	27	1	407
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	21	31	21	302
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	31	29	31	405
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	131	17	131	302
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	171	24	171	407
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	201	5	201	200
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	231	1	231	200
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	311	26	311	302
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	361	26	361	406
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	391	22	391	404
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	451	35	451	302
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	501	1	511	406
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	501	5	501	405

Apache Druid - Confluent Cloud - localhost:8888/unified-console.html#load-data

Connect and parse raw data Transform data and configure schema Tune parameters Verify and submit

Start Connect Parse data Parse time Transform Filter Configure schema Partition Tune Publish Edit spec

Orchestrator requires flat data (non-nested, non-hierarchical). Each row should represent a single event.

If you have nested data, you can flatten it here. If the provided flattening capabilities do not work for your data, you must flatten your data before ingesting it into Druid.

Ensure that your data appears correctly in a new column or operator.

Learn more

Input format: avro_stream

Binary as string: True

Add column flattening: Apply

Next Parse time

remote_user	request	referrer	agent	bytes	ip	time	userid	jtime	status	
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	1	27	1	407
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	21	31	21	302
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	31	29	31	405
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	131	17	131	302
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	171	24	171	407
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	201	5	201	200
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	231	1	231	200
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	311	26	311	302
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	361	26	361	406
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	391	22	391	404
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	451	35	451	302
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	501	1	511	406
122.73.162.102	GET /avro/	-	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36	406	122.73.162.102	2023-07-10T00:00:00Z	501	5	501	405

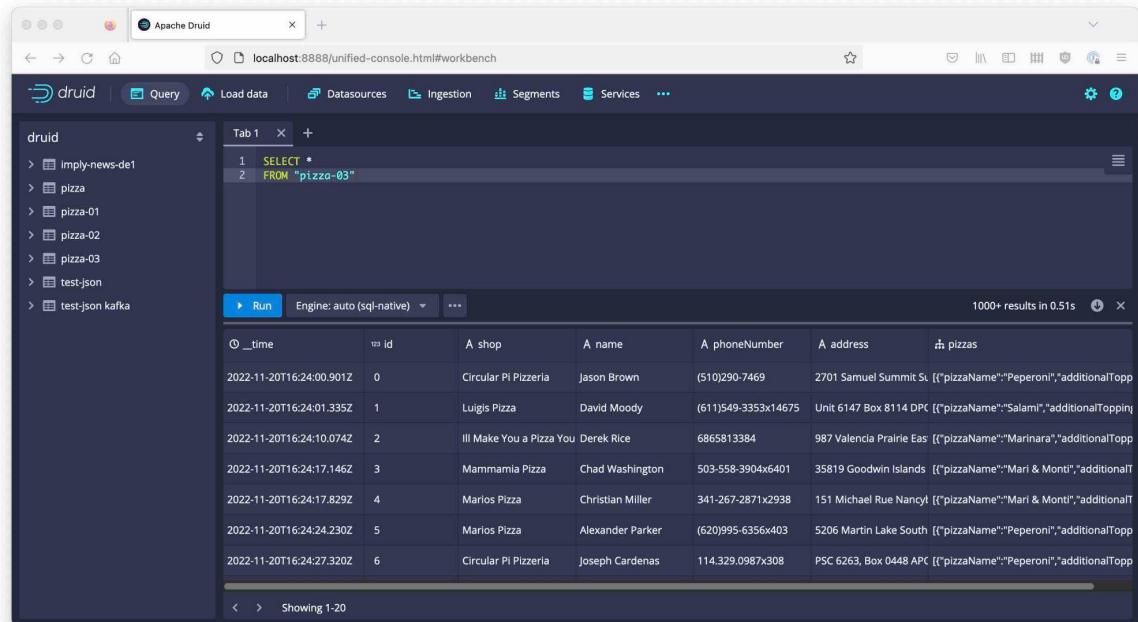
Avro Schema Registry 지원

Json, Protobuf Schema Registry 지원 (Imply)

Kafka Key, Timestamp, Header Field 가져오기

```
"inputFormat": {  
    "type": "kafka",  
    "headerLabelPrefix": "kafka.header.",  
    "timestampColumnName": "kafka.timestamp",  
    "keyColumnName": "kafka.key",  
    "headerFormat": {  
        "type": "string"  
    },  
    "keyFormat": {  
        "type": "csv",  
        "columns": [  
            "k"  
        ]  
    },  
    "valueFormat": {  
        "type": "json"  
    }  
},  
  
"timestampSpec": {  
    "column": "kafka.timestamp",  
    "format": "millis"  
}
```

중략



The screenshot shows the Apache Druid unified console interface. On the left, there is a sidebar with a tree view of data sources: 'druid' (imply-news-det, pizza, pizza-01, pizza-02, pizza-03, test-json), 'druid kafka' (test-json kafka), and a 'workbench' tab. The main area has a 'Query' tab selected. A SQL query is entered in the editor:

```
1 SELECT *  
2 FROM "pizza-03"
```

Below the editor, there are buttons for 'Run' and 'Engine: auto (sql-native)'. The results table has columns: '_time', 'id', 'shop', 'name', 'phoneNumber', 'address', and 'pizzas'. The results show 20 rows of pizza data from the 'pizza-03' topic, including columns like pizzaName, additionalToppings, and address details.

_time	id	shop	name	phoneNumber	address	pizzas
2022-11-20T16:24:00.901Z	0	Circular Pi Pizzeria	Jason Brown	(510)290-7469	2701 Samuel Summit St	[{"pizzaName": "Peperoni", "additionalToppings": null}]
2022-11-20T16:24:01.335Z	1	Luis's Pizza	David Moody	(611)549-3353x14675	Unit 6147 Box 8114 DPC	[{"pizzaName": "Salami", "additionalToppings": null}]
2022-11-20T16:24:10.074Z	2	Ill Make You a Pizza You	Derek Rice	6865813384	987 Valencia Prairie Eas	[{"pizzaName": "Marinara", "additionalToppings": null}]
2022-11-20T16:24:17.146Z	3	Mamma Mia Pizza	Chad Washington	503-558-3904x6401	35819 Goodwin Islands	[{"pizzaName": "Mari & Monti", "additionalToppings": null}]
2022-11-20T16:24:17.829Z	4	Marios Pizza	Christian Miller	341-267-2871x2938	151 Michael Rue Nancy	[{"pizzaName": "Mari & Monti", "additionalToppings": null}]
2022-11-20T16:24:24.230Z	5	Marios Pizza	Alexander Parker	(620)995-6356x403	5206 Martin Lake South	[{"pizzaName": "Peperoni", "additionalToppings": null}]
2022-11-20T16:24:27.320Z	6	Circular Pi Pizzeria	Joseph Cardenas	114.329.0987x308	PSC 6263, Box 0448 APC	[{"pizzaName": "Peperoni", "additionalToppings": null}]

Kafka Lookup

kafka lookup 세팅

```
{  
    "type": "kafka",  
    "kafkaTopic": "testTopic",  
    "kafkaProperties": {  
        "bootstrap.servers": "kafka.service:9092"  
    }  
}
```

Offset	Key	Payload
1	NZ	Nu Zeelund
2	AU	Australia
3	NZ	New Zealand
4	AU	null
5	NZ	Aotearoa
6	CZ	Czechia



Kafka 메세지

Key	Value
NZ	Aotearoa
CZ	Czechia

기대 결과값

Update 요건 등에 활용

```
SELECT * FROM ORDER  
WHERE "ORDER_ID" IN ('ORDER-NUMBER')  
AND TIME_PARSE(LOOKUP("ORDER_ID", 'KAFKA-LOOKUP')) = __TIME
```

마치며

Druid Resources

- Druid Tech Talk
- 신청:

[https://calendly.com/druid
community/tech-talk](https://calendly.com/druid/community/tech-talk)

Or sunny.yoon@imply.io



druid



Imply에서는 Apache Druid를 소개하고, 기술 심층 분석 세션을 제공합니다.

이 세션은 오픈소스 Druid 사용자 혹은 Druid를 검토하시는 팀이 Druid의 기술 기능과 디자인, 그리고 비즈니스에 어떻게 적용되는지 뿐 아니라 분석의 최신 동향에 대해 배우기에 적합합니다. Druid가 어떻게 Kafka & Kinesis를 보완하고 Netflix 및 Salesforce와 같은 회사에서 사용하는지 알아보십시오.

참석 대상자

- CTO
- 데이터 아키텍트
- 프로젝트 매니저
- 데이터 엔지니어
- 소프트웨어 개발자
- BI/ETL 분석가

 imply

1시간 TECH TALK 아젠다 예시

- 소개 (5분)
- 최신 분석 애플리케이션 환경 (10분)
 - 분석 애플리케이션이라 무엇인가
 - 기업에서 분석 애플리케이션을 구축하는 이유와 분석 애플리케이션의 현대화의 중요성에서 변곡점에 있는 이유
 - 업계의 최신 분석 애플리케이션 구축 예시
 - 분석 애플리케이션은 어떻게 구축되는가?
 - 분석 애플리케이션 구축의 주요 과제
- 최신 분석 애플리케이션의 아키텍처 (15분)
 - Archetype/high level
 - 데이터 소스 및 ELT/ETL
 - 데이터베이스
 - 쿼리 엔진
 - 프론트 엔드 애플리케이션 스택 및 데이터 계층에 대한 통찰
- Druid 아키텍처 별류 (10분)
 - Druid 기초
 - Druid 가 아키티입 / 일반 아키텍처의 문제를 해결하는 이유
- 데모 (5분)
- Q&A & Next Steps (10 분)

무료 Druid Basic Training

<https://learn.implicitly.io/>

← ➔ imply Sign In

Brought to you by contributors and committers to the Apache Druid project, welcome to Imply's public catalog of practical courses on all things Druid!

If you're new to Druid, take "Apache Druid Basics" to get hands-on with this open source real-time analytics database from Apache. Hear about how Druid came to be, and about all of the database's processes. Once completed, you'll receive a certification that you can share publicly, including LinkedIn!

This service is provided by Imply; this domain and service are not affiliated with, endorsed by, or otherwise associated with the Apache Software Foundation (ASF) or any of its projects. Apache, Apache Druid, Druid, and the Druid logo are either registered trademarks or trademarks of ASF in the USA and other countries.



Apache Druid® Basics

Druid is becoming the go-to cloud-native answer to scalable time-series data storage and analytics. So, if you have time-series data, you'll want to know how to use Druid.

FREE 4 hr 30 min



Apache Druid® Ingestion and Data Modeling

Data modeling is the key to leveraging your Apache Druid® database. Learn how to ingest data into Druid data models that are fast and scalable.

FREE



Apache Druid® Multi-Stage Query Framework Basics

New to Druid 24.0, learn what MSQ is and how to use it!

FREE

Try Druid in 20 Minutes

Learn how to standup a Druid cluster, ingest some data and then query the data, all in the time it takes to drink a cup of coffee!

FREE

Druid Meetup

<https://www.meetup.com/ko-KR/Druid-Seoul/members/>



Start a new group - 30% off!

PRO
Try for free

메시지

알림



Druid@Seoul | 오픈소스 Druid 사용자 모임

★★★★★ (74) ?

서울, 한국(대한민국)

회원 525명 · 공개 그룹 ?

주최자: Jungryong Lee 외 7명

공유하기:

카카오톡 오픈채팅 - Druid 사용자 모임



Thank you!

문의 : 윤선정 이사 | sunny.yoon@imply.io | Imply Korea