为数据赋能
敏捷高效的数据处理

徐岷峰

TalkingData 资深架构师

数据处理

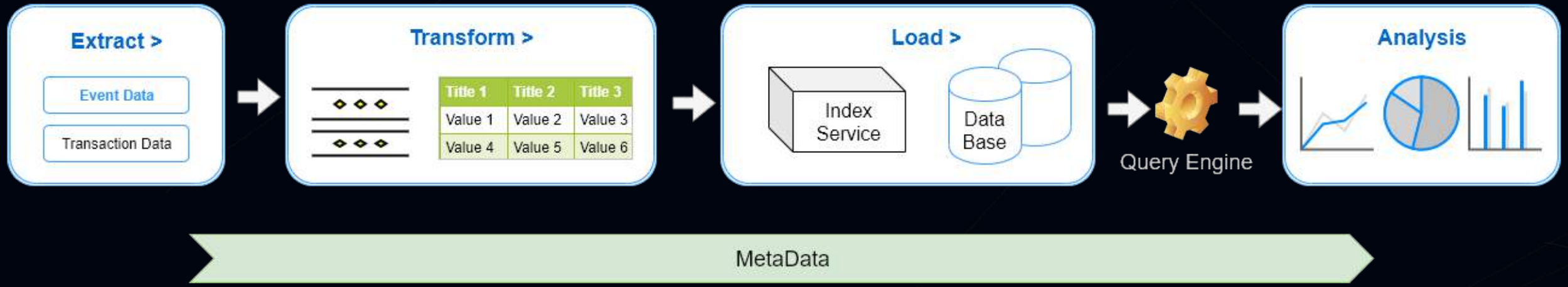
Data Wrangling

TalkingData

Data Wrangling

数据处理的流程

- ETL：从多种数据源提取数据，做清理、聚合、派生，加载到特定存储
- ELT：利用存储的计算能力做数据变换



Data Wrangling

TalkingData 遇到的问题

需求

理解不一致/不清晰

沟通成本高

开发

技术栈纷杂

功能重复开发

预计算

缺乏统一的标准与服务

即席查询

多种异构数据

元数据系统

Meta Data System

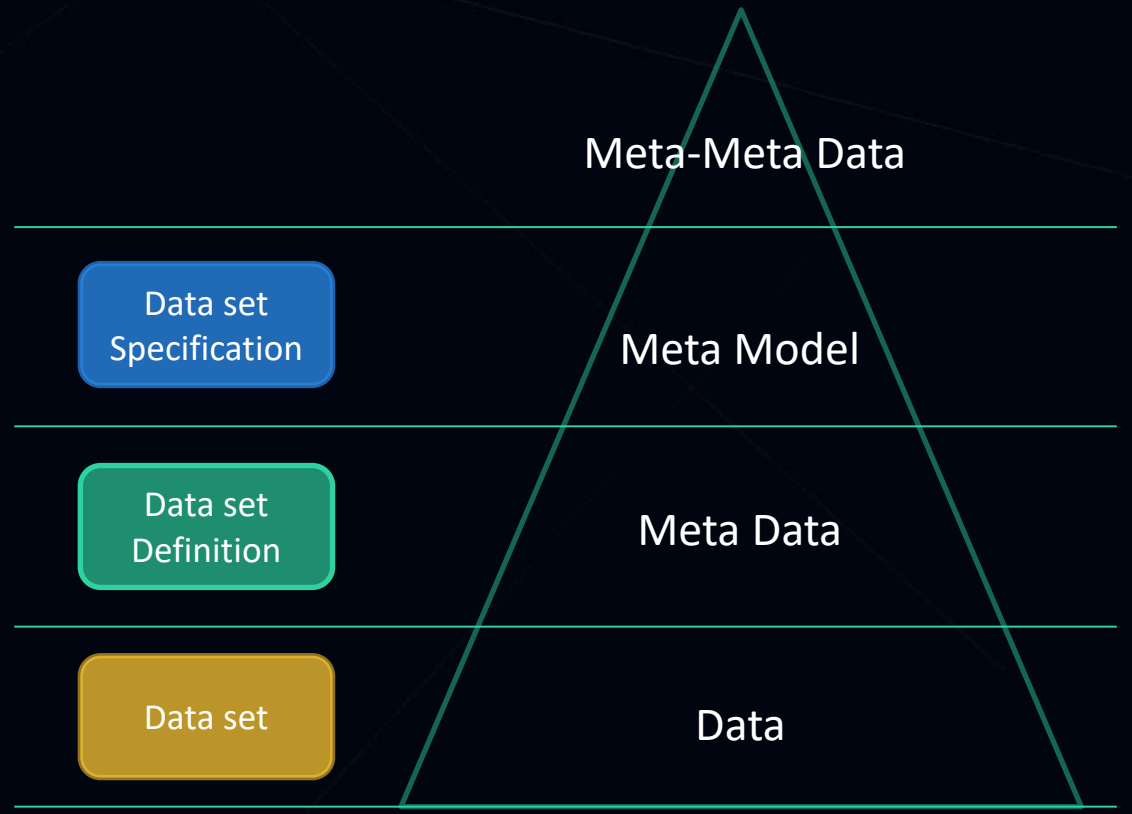
TalkingData

Meta Data System

元数据是实现敏捷的关键

元数据是定义数据的数据
是实现系统复用的关键
可以直接作为系统的外部接口
元数据驱动

元数据是对业务的高度抽象
某种程度上确定了系统的数据边界



Meta Data System

元数据规范（meta model）

数据源（Data Source）

分区定义（Partition）

数据集（Data Set）

模式（Schema）

数据处理过程（Recipe）

算子（Operator）

多维模型（Multi-Dimension）

实体关系模型（ER）

模型（model）

Meta Data System

实现

对象	存储	检索	通知	部署
JSON	MySql SQLite/Derby	索引ES	MQ	Service Process

数据变换

Data Transform

Data Transform

没有银弹

数据处理的核心
机器学习的基础
（特征工程）

不深奥的技术
却最耗费人力
尤其是维护工作

纷繁的ETL工具：
PDI（Kettle）
Talend Data Integration
Oracle Data Integration
DataX …

适合的才是最好的

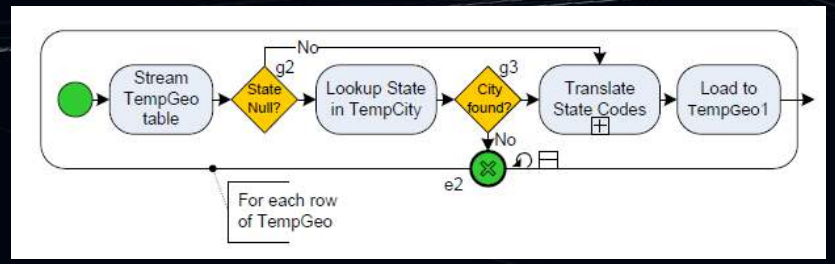
共同的思路：

- 语义抽象
  （Session/JOB/Operator，DAG）
- 元数据驱动（数据源、目的地、数据结构、依赖、过程）
- 执行引擎
- 数据质量监测
- 流程监测
- 数据安全与审计

Data Transform

TalkingData 的解决方案 – 语义抽象

- BPMN
- JSON
- DAG
- Sequence
  - condition
  - Operator
    - Function
    - Parameter
  - returnAS
- Parallelism




```
"common": ...,
"domain": ...,
"context": ...,
"input": ...,
"output": ...,
"logic": {
    "type": "array",
    "items": {
        "type": "string",
        "description": "the sequence or parallel name"
    }
},
"dependency": {
    "type": "array",
    "items": {
        "type": "string",
        "description": "The other TDAG's name in same domain"
    }
}
```

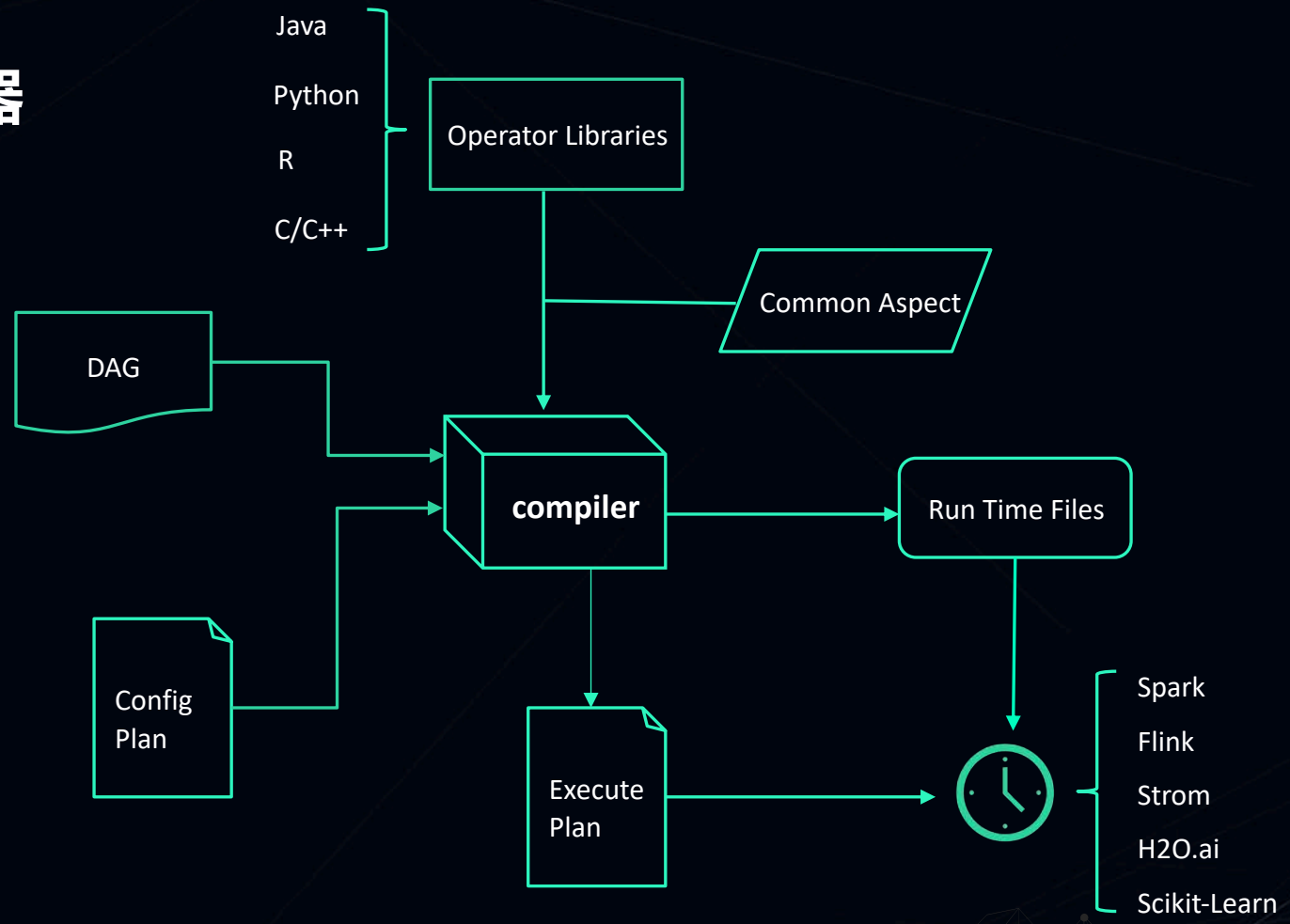
```
"func": {
    "name": "isBlank",
    "description": "check whether the string is blank",
    "parameters": [
        {
            "name": "str",
            "type": "string",
            "description": "The original string to deal with"
        }
    ],
    "returns": {
        "name": "isBlank",
        "type": "boolean"
    }
},
"pattern": "MAP"
```

```
"name": ...,
"processingType": {
    "$ref": "definitions.json#/definitions/ProcessingTypeEnum"
},
"operators": {
    "type": "array",
    "items": {
        "condition": {
            "$ref": "defintions.json#/definitions/condition"
        },
        "operator": {
            "type": "string"
        },
        "parameters": {
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "returnAs": {
            "type": "string"
        },
        "comment": {
            "type": "string"
        },
        "else": {
            "$ref": "#/definitions/invoke"
        }
    }
},
"parallelism": {
    "type": "integer",
    "default": 1
}
```

Data Transform

TalkingData 的解决方案 – 编译器

- 逻辑计划（DAG）
- 配置文件
- 算子库，多种语言实现
- 通用特性
  - 日志
  - 审计
  - 计量
  - 安全
- 运行时文件，Jar，.R，.so，.py



索引服务

Index Service

TalkingData

Index Service

索引服务

索引，是数据查找和定位的关键

- B+树、跳表、倒排 …

索引，也是一种数据，可以运算

- Bitmap
- 精准的排重统计
- 集合运算，Intersect, Union, Except

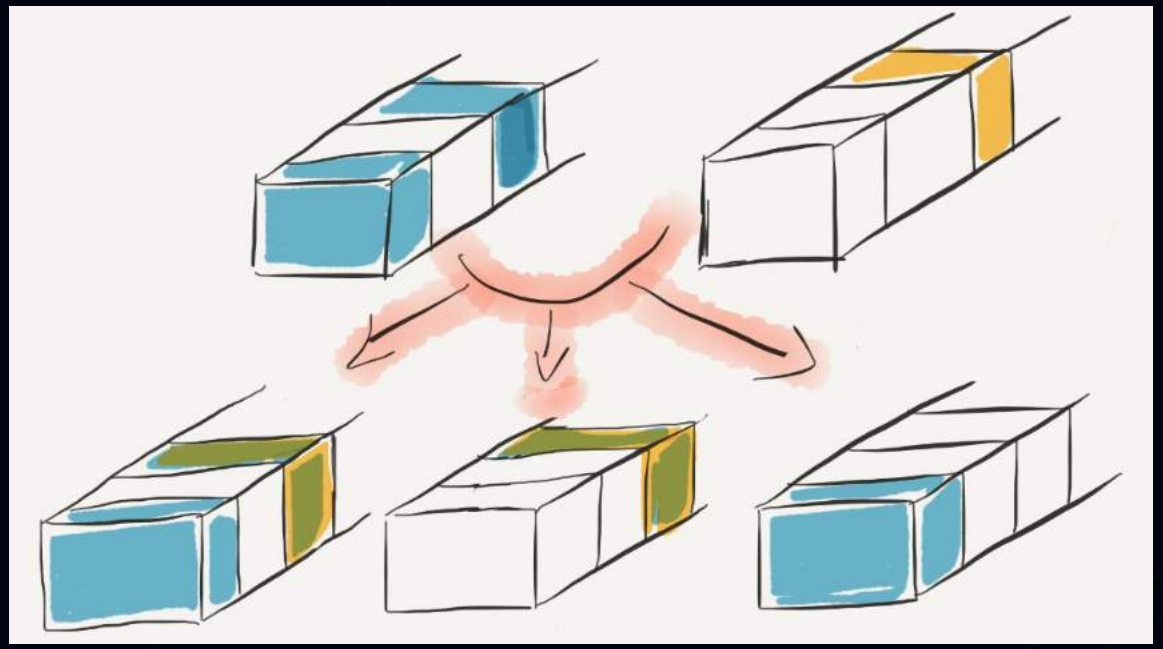
索引服务

- 生成索引
- 检索
- 运算
- 维护

Index Service

Bitmap

举个例子：
定义Offset为UserId
时间粒度为天，
索引1（蓝色）代表玩“吃鸡”游戏的用户
索引2（黄色）代表玩“王者”游戏的用户



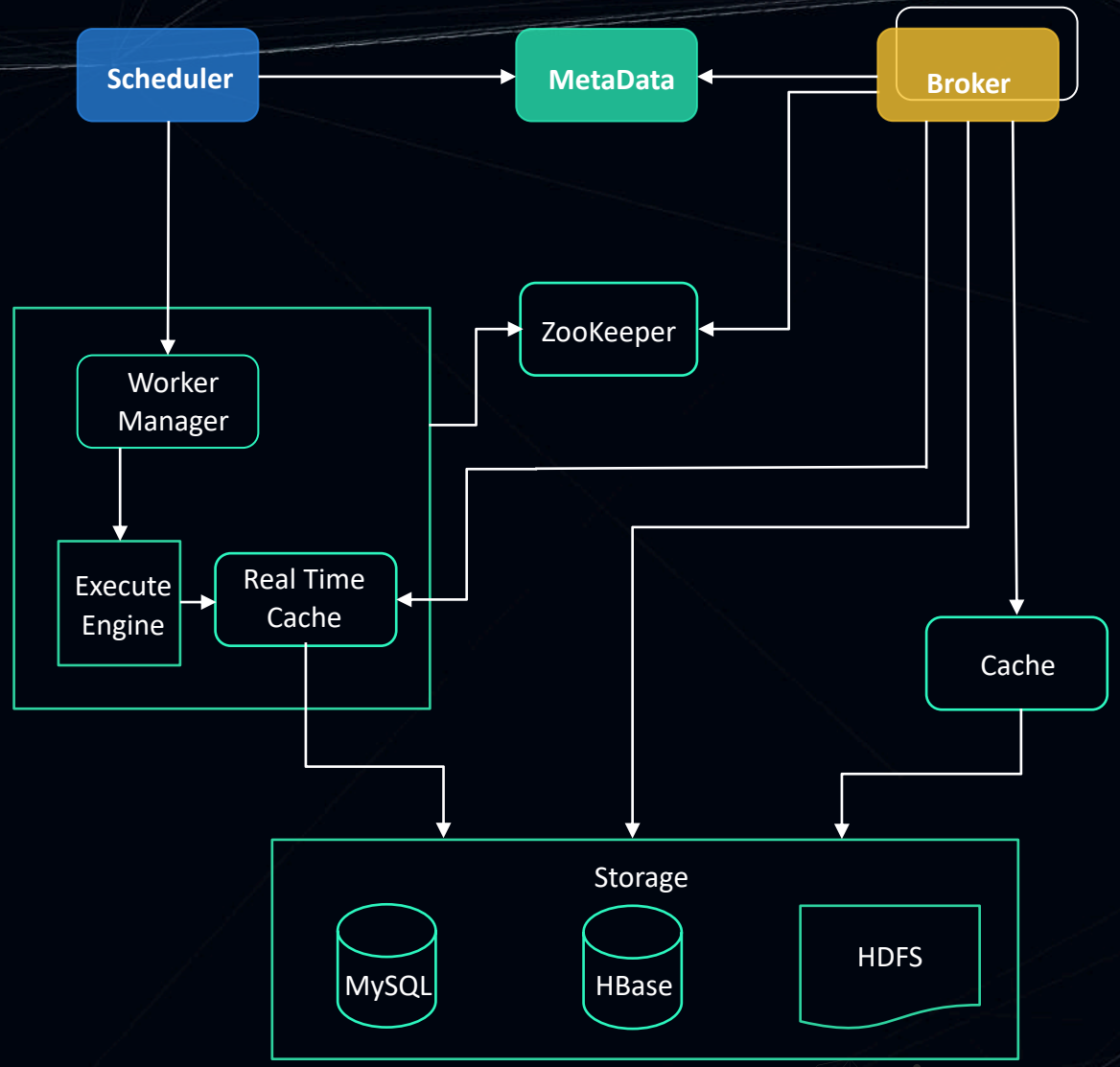
统计：

- 今天玩了“吃鸡”或“王者”的用户
- 今天既玩了“吃鸡”又玩了“王者”的用户
- 今天玩了“吃鸡”但没玩“王者”的用户

Index Service

架构

- 索引生成
  - 可适配多种执行引擎，Storm，Flink，Spark
- 索引存储
  - 支持异构存储
- 索引查询
  - 查询能力可扩展
  - 支持实时数据查询与运算
- 元数据驱动
- Zookeeper同步状态



Index Service

元数据

- Domain
- Dimension
- Storage
- Implementation
- uniqueKey
- Index

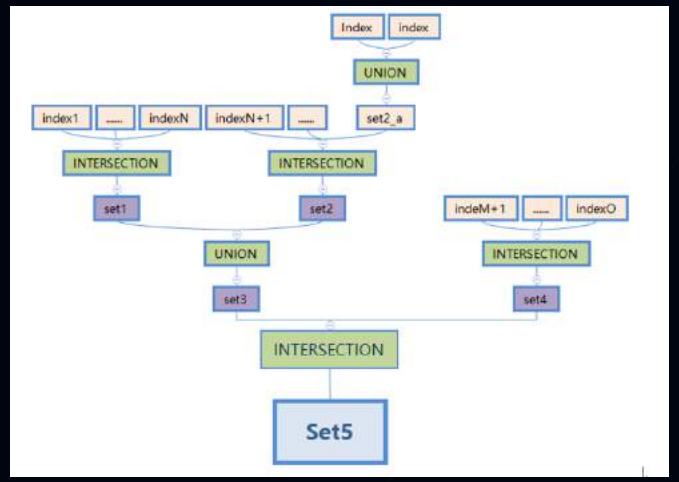
```
"common": ...,
"domain": ...,
"uniqueKey": ...,
"maxEffective": ...,
"timeDimension":{
    "type": "object",
    "properties": {
        "startTime": {
            "type": "string",
            "format": "date-time",
            "pattern": "yyyy-MM-dd'T'HH:mm:ss.SSSXXX",
            "examples": [
                "2018-01-12T02:34:56.000+08:00"
            ]
        },
        "granularity": {
            "$ref": "common.json#/definitions/timeGranularityEnum"
        }
    }
},
"dimensionsRel": {
    "$ref": "#/definitions/operand"
},
"implementation": {
    "$ref": "implementation.json"
}
```

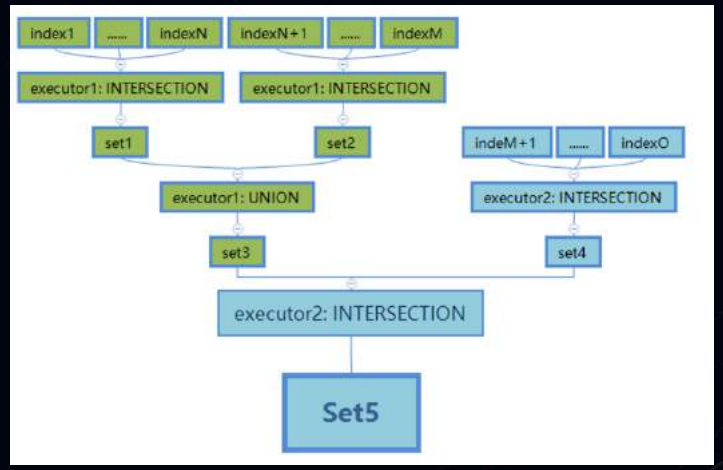
Index Service

查询

- DSL – ISQL
  - Filter，维度间的集合运算
  - 查询类型：select，groupby，topN，timeSeries，window，subQuery
  - 返回结果：index，count，idList
- Logical Plan – Physical Plan
- 固定“场景”的查询

```
"name": ...,
"queryType": ...,
"queryOptions": ...,
"domain": ...,
"scence": ...,
"intervals": ...,
"subQueries": ...,
"sets": ...,
"computations": ...,
"result": ...
```



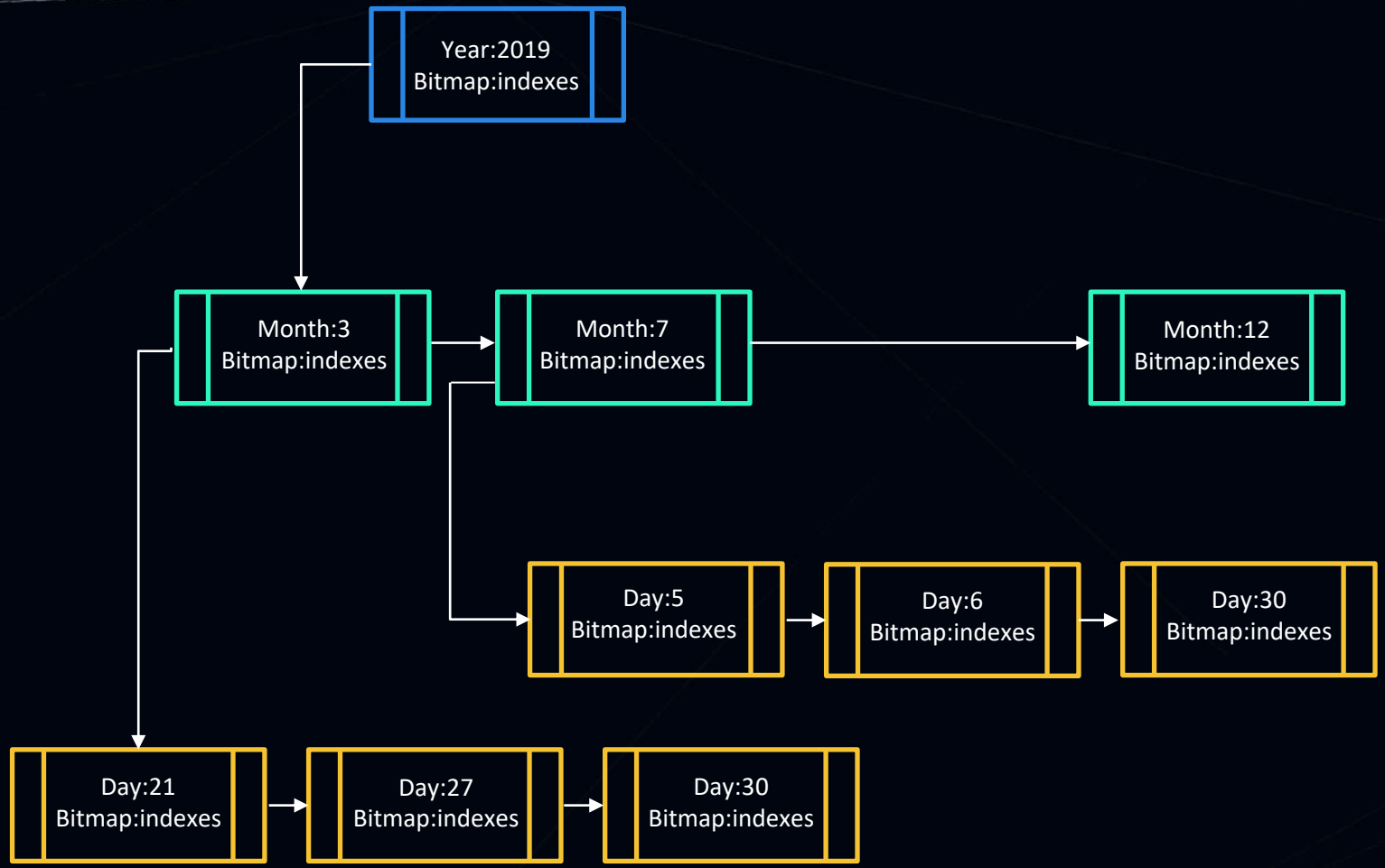


Index Service

二级索引

按时间粒度分层的多叉树

用bitmap实现的倒排索引
- 以索引维度做Key
- 以索引标识作为Offset



Index Service

维护-金字塔模型

处理方式：

合并

删除

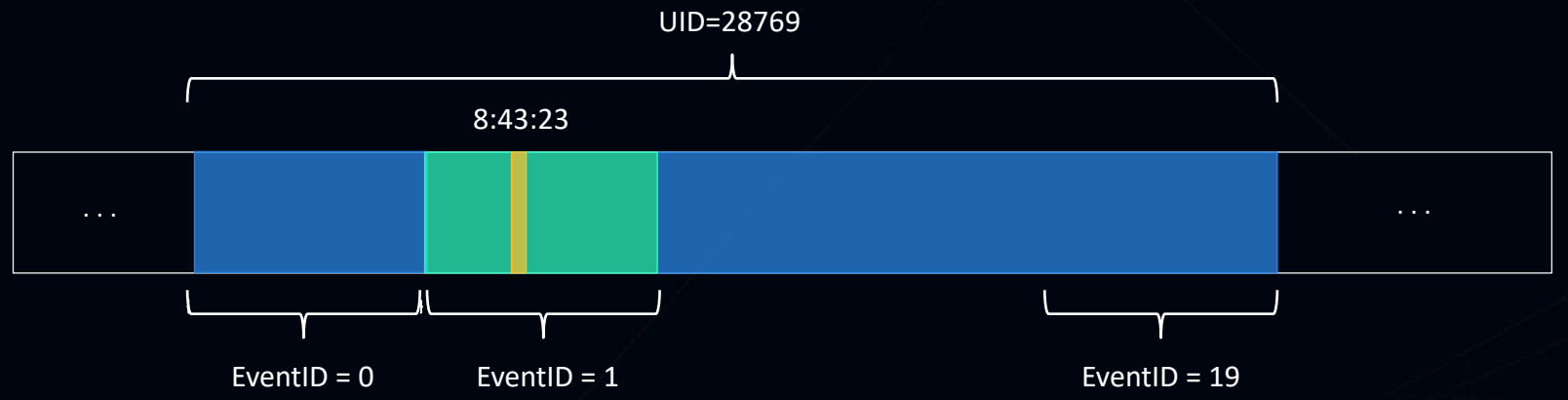
导出

Index Service

有序漏斗 - 时序索引

- 用bitmap存储事件的时序关系：
  - 以时间作为Offset，精度不宜太高，秒
  - 压缩存储：1个RoaringBitmap可以存储1242个用户在一天内的20个基准事件序列。（2^31/3600/24/20）
  - 掩码
  - 不同的二级索引。

- 配合Index Service和关系数据库实现有序漏斗
  - 每日事件发生的用户索引
  - 分群索引
  - 事件明细过滤



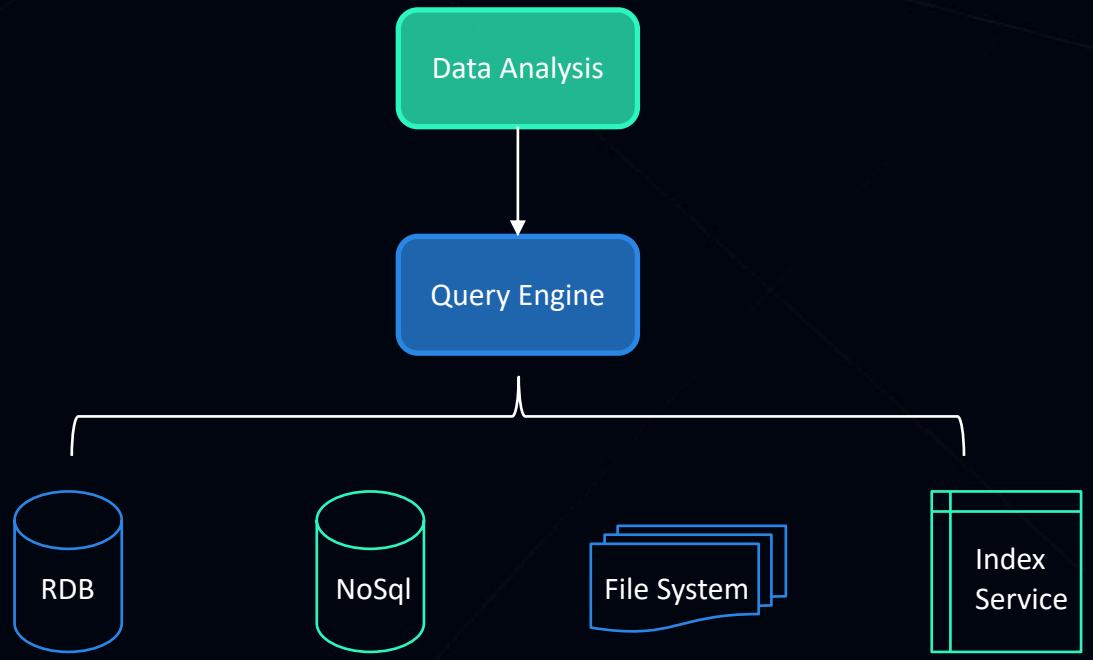
查询引擎

Query Engine

Query Engine

解决的问题

- ETL->ELT
- Custom Query
- 异构数据
  - 不同的DSL
  - 查询优化
- 解耦物理存储和业务逻辑
  - 虚拟表，视图

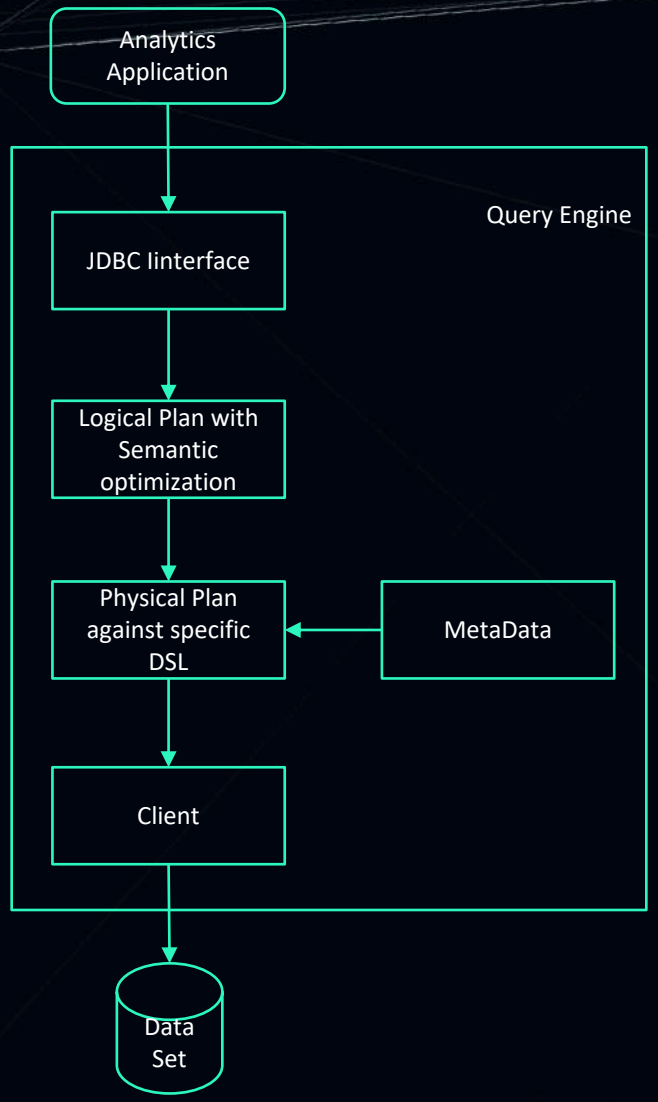


Query Engine

设计思路

- 统一的查询接口：SQL
- 语义优化的逻辑计划
- 按数据源转化到对应的DSL（物理计划）
- 通过客户端发送请求，获取数据
- 元数据：DataSet，特定的转换规则，Client

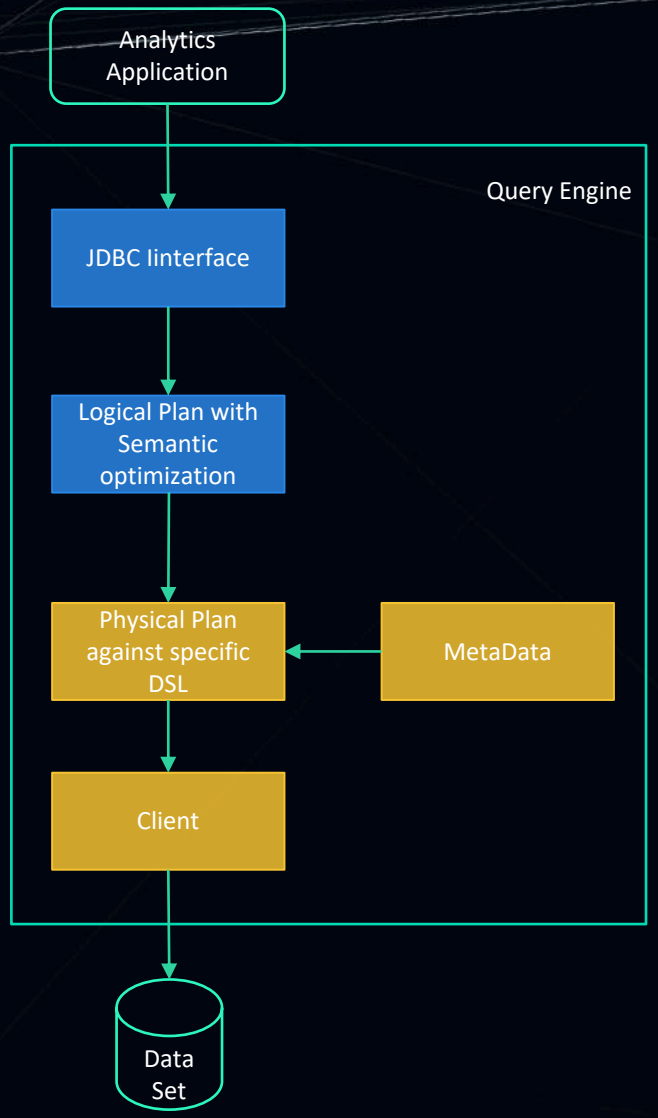
- 部署：Library（Jar）



Query Engine

实现

- Core: Apache Calcite
- 新增或改写Adapter: Druid，CarbonData，Kudu
- 元数据
- 客户端



Query Engine

实例

```
SELECT COUNT(`event`.`offset`) AS `pv`, `event`.`starttime_day` AS `starttimeDay`
FROM `CARBON`.`ae_event` AS `event`
INNER JOIN `KUDU`.`ae_profile` AS `profile` ON `event`.`offset` = `profile`.`offset` AND `event`.`productid` = `profile`.`productid`
WHERE `event`.`eventtype` IN (SELECT `DICTIONARY_ITEM`.`id`
FROM `MYSQL`.`DICTIONARY_ITEM` AS `DICTIONARY_ITEM`
WHERE `DICTIONARY_ITEM`.`dic_item_key` = 'eventtype' AND `DICTIONARY_ITEM`.`dic_item_value` = '访问页面')
GROUP BY `event`.`starttime_day`
```

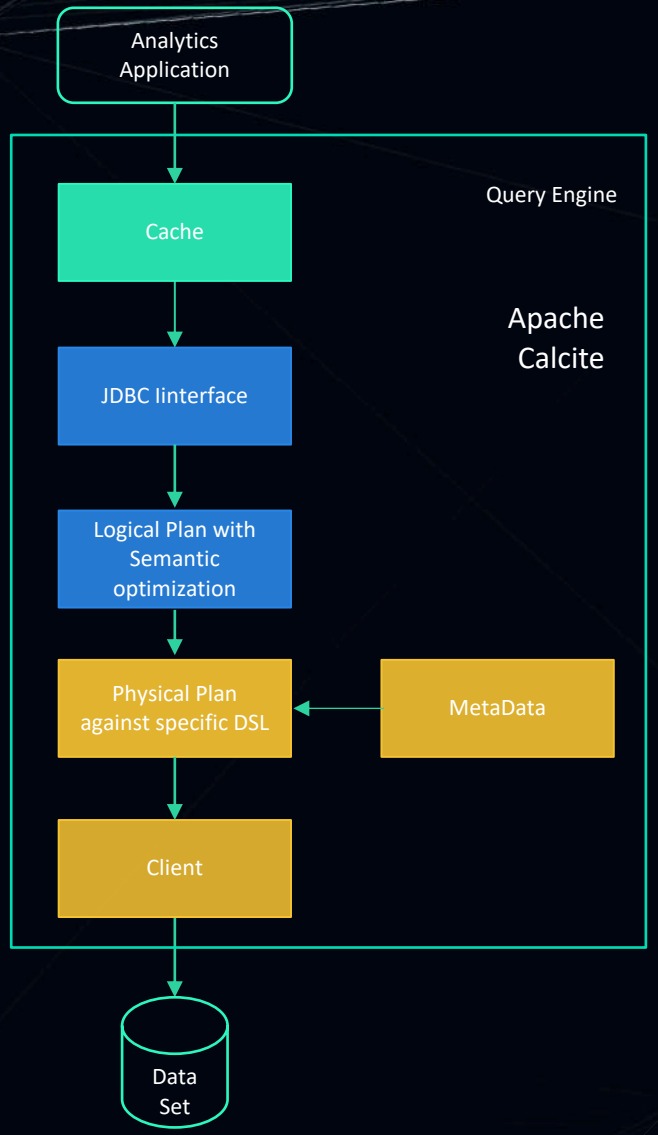
```
LogicalProject(pv=[$1], starttimeDay=[$0])
  LogicalAggregate(group=[{0}], pv=[COUNT($1)])
    LogicalProject(starttimeDay=[$9], offset=[$13])
      LogicalFilter(condition=[IN($12, {
                        LogicalProject(id=[$0])
                          LogicalFilter(condition=[AND(=($2, 'eventtype'), =($3, '访问页面'))])
                            JdbcTableScan(table=[[MYSQL, DICTIONARY_ITEM]])
                        })])
        LogicalJoin(condition=[AND(=($13, $58), =($8, $55))], joinType=[inner])
          JobServerTableScan(table=[[CARBON, ae_event]])
          JobServerTableScan(table=[[KUDU, ae_profile]])
```

```
EnumerableCalc(expr#0..1=[{inputs}], pv=[$t1], starttimeDay=[$t0])
  EnumerableAggregate(group=[{1}], pv=[COUNT($3)])
    EnumerableJoin(condition=[=($2, $6)], joinType=[inner])
      JobServerToEnumerableConverter
        JdbcJoin(condition=[AND(=($3, $5), =($0, $4))], joinType=[inner])
          JdbcProject(productid=[$8], starttime_day=[$9], eventtype=[$12], offset=[$13])
            LowCostFilter(condition=[AND(=($8, 5), =($42, _UTF-8'product_p_0'))])
              1(table=[[CARBON, ae_event]])
          JdbcProject(productid=[$12], offset=[$15])
            LowCostFilter(condition=[AND(=($12, 5), =($28, _UTF-8'product_p_0'))])
              1(table=[[KUDU, ae_profile]])
      EnumerableAggregate(group=[{0}])
        EnumerableCalc(expr#0..8=[{inputs}], expr#9=['eventtype':VARCHAR(60)], expr#10=[=($t2, $t9)], expr#11=['访问页面':VARCHAR(512)], expr#12=[=($t3, $t11)], expr#13=[AND($t10, $t12)], proj
          JdbcToEnumerableConverter
            JdbcTableScan(table=[[MYSQL, DICTIONARY_ITEM]])
```

Query Engine

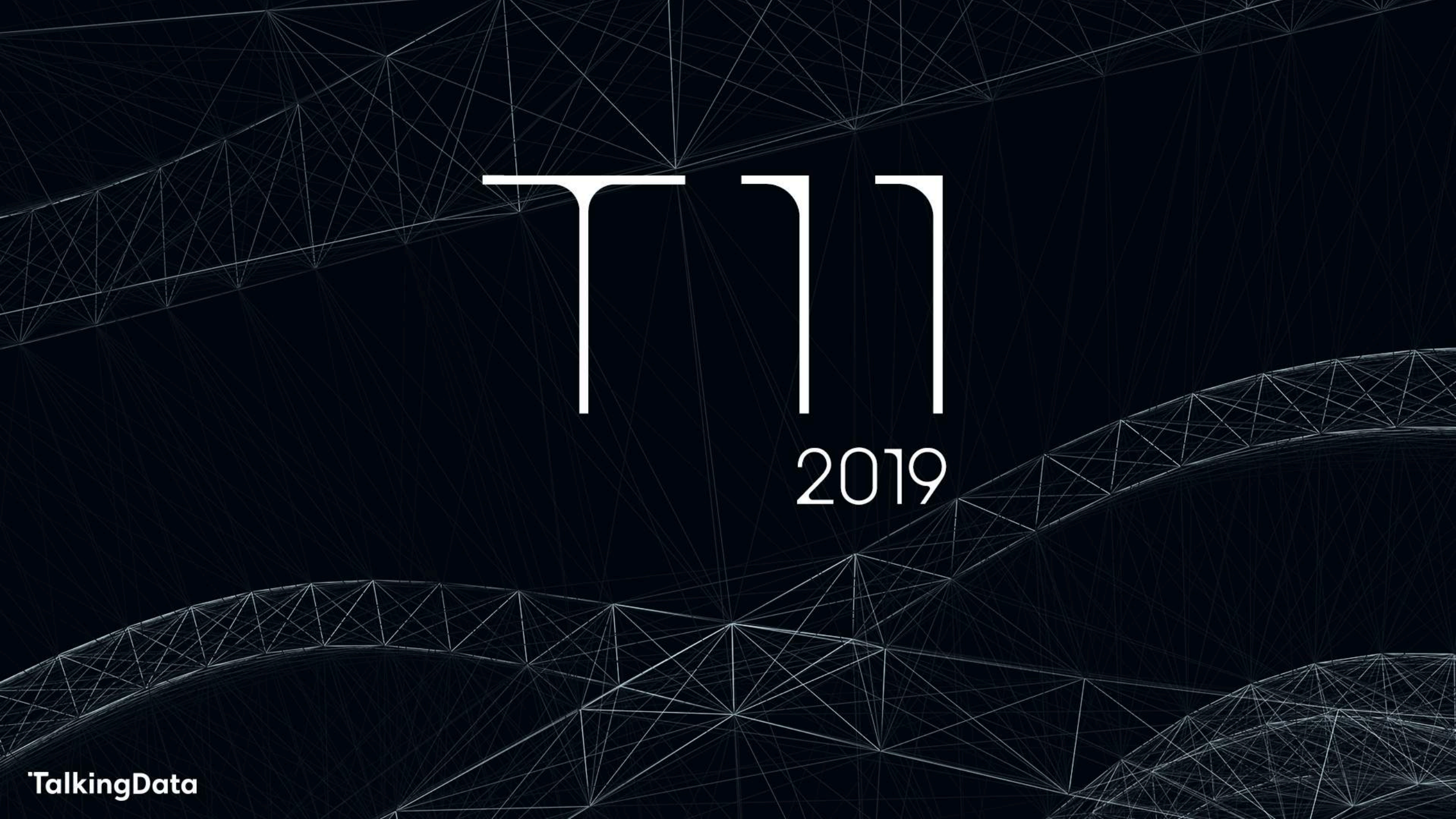
增强

- Pitfall : 性能降低 10%-30%
- Cache <SQL,DSL>
- Bypass optimization and Translation



Summary

需求	开发	预计算	即席查询
理解不一致/不清晰 沟通成本高	技术栈纷杂 功能重复开发	缺乏统一的标准与服务	多种异构数据
MetaData	Transform Tool	Index Service	Query Engine


2019
TalkingData