

数据来源：数据库产品上市商用时间



第十三届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2022

数据智能 价值创新



线上直播 | 2022/12/14-16



B 站基于 Iceberg 湖仓一体优化实践及 智能化管理平台的助力

向阿鲲/BILIBILI/OLAP 平台资深开发工程师

湖仓一体项目在 B 站 OLAP 平台的落地实践

基于 Iceberg 湖仓一体内核优化

智能化管理平台 Magnus

未来规划

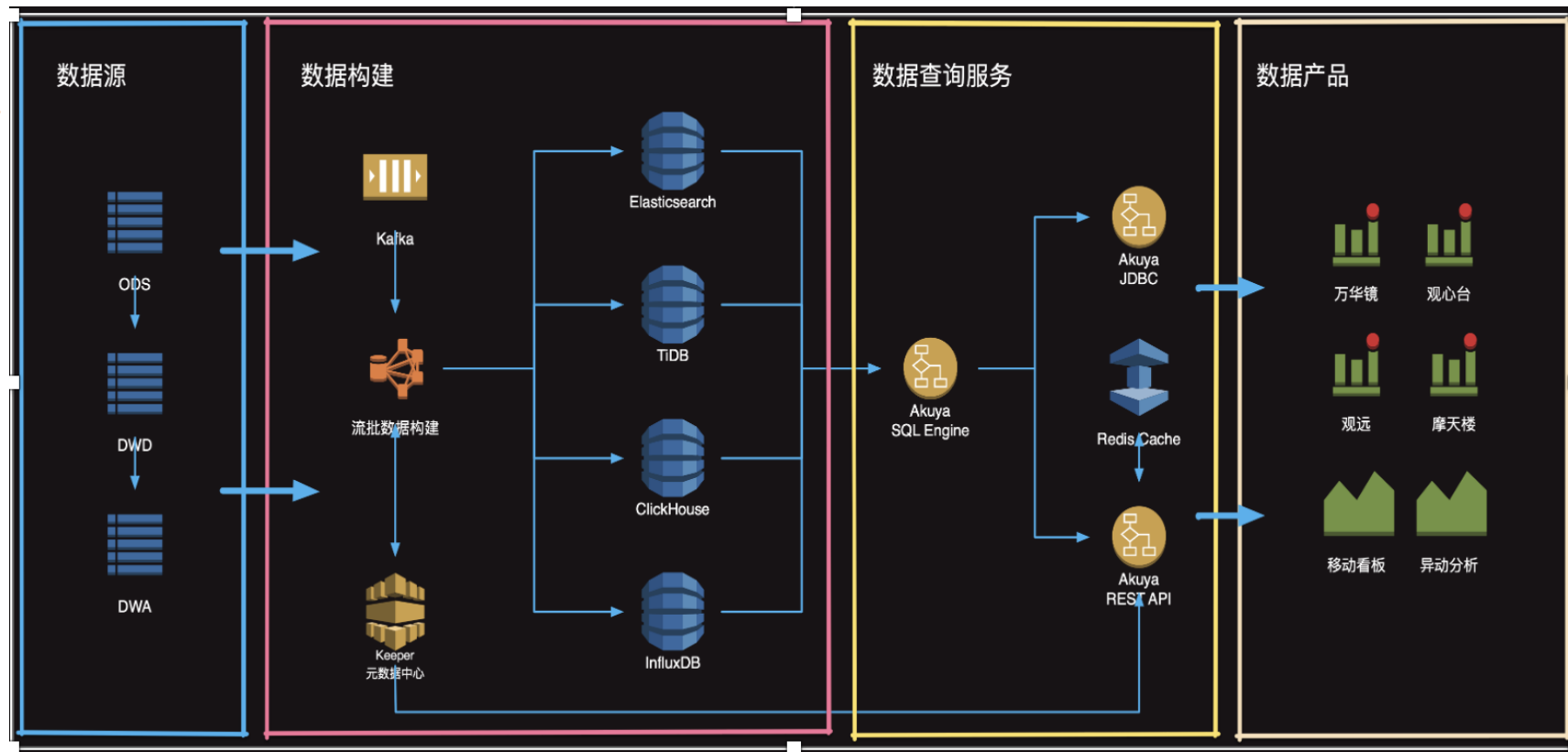


湖仓一体项目在 B 站 OLAP 平台的落地实践

湖仓一体项目背景（早期数据服务架构）

Hadoop

ETL



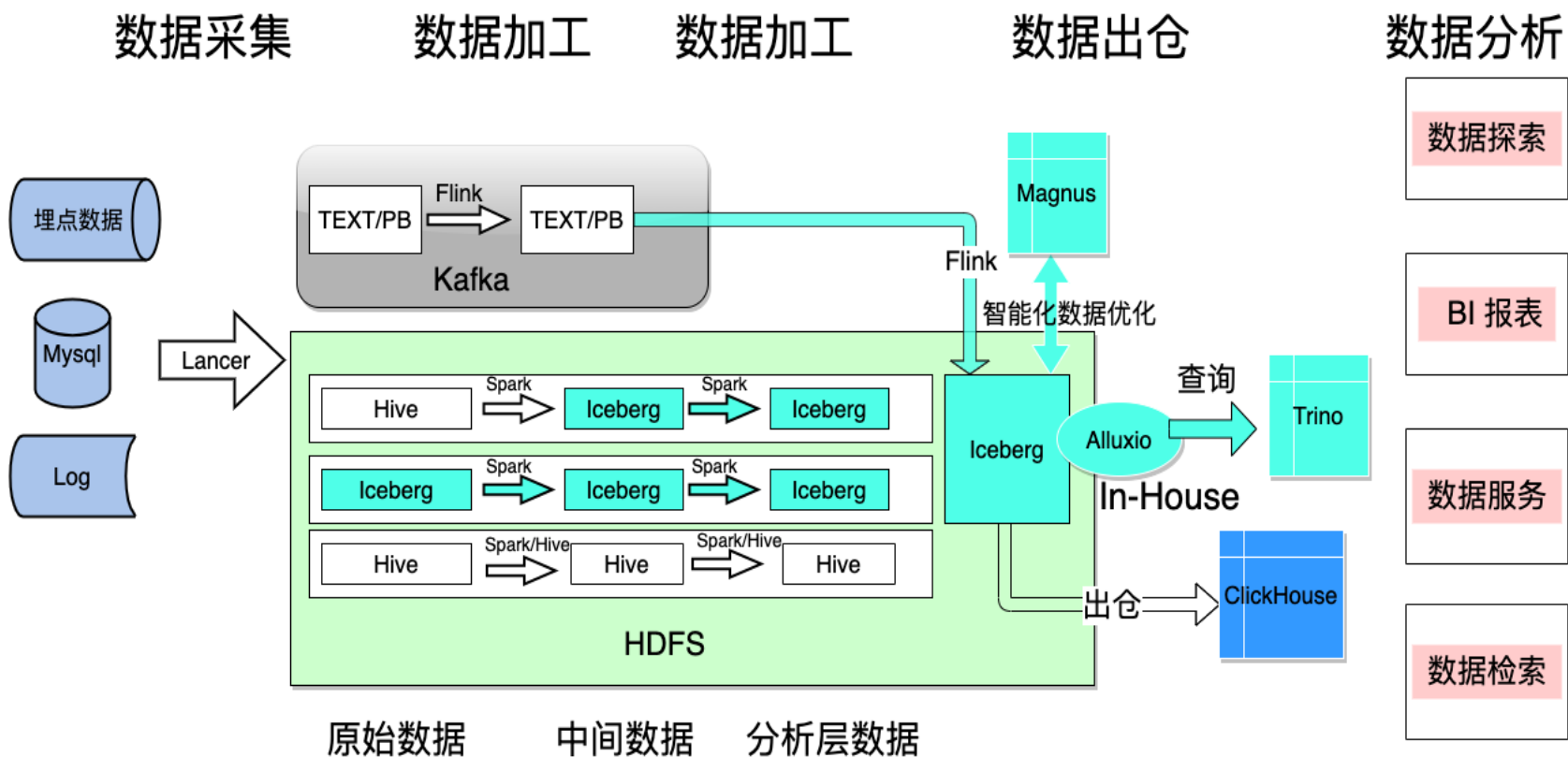
痛点:

- 数据出仓繁琐，数据一致性难保障
- 数据处理复杂，需要定制化
- 需要为不同的存储引擎优化计算

湖仓一体项目架构

Hadoop 生态上引入 Iceberg 组建

在拥有数据湖灵活性的同时、打造一套高性能的数仓体验



- 数据不用出仓，避免了数据重复存储
- 大数据可以添加索引，支持毫秒级/秒级查询能力，满足基本取数场景
- 数据处理更加高效，从数据处理到取数的过程缩短
- 支持数据增量 update，支持事务，能对接更多业务场景

业务场景落地

➤ 取数服务

运营后台、数据产品（万华镜、观星台、Boss 看板等）、OLAP 多维分析等

➤ ABTest 实验平台

➤ BI 报表

➤ 标签人群圈选

➤ 日志检索

查询统计详情

最近一周查询数量

222k

上周Trino 平均P90响应时间

1.49

Trino查询成功率

0.9996

单日查询量：22.2w

P90 耗时：1.49s

P95 耗时：2.5s

P99 耗时：5.6s

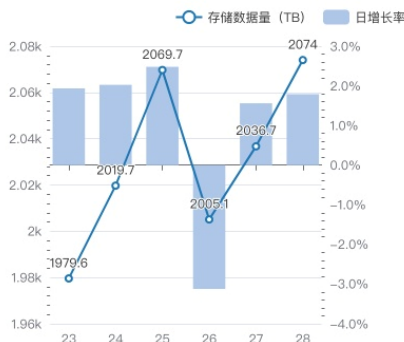
Iceberg 数据总量：2PB+

峰值 QPS：300q/s（单集群）

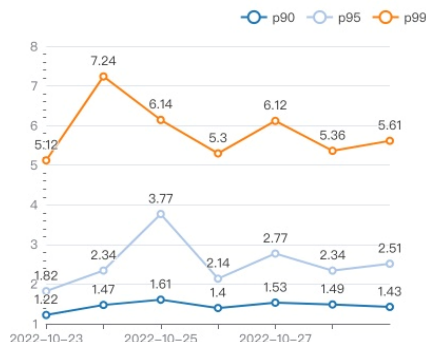
数据产品（万华镜）查询平

均耗时：200ms

Iceberg表存储数据量



上周Trino查询耗时p90/95/99(s)



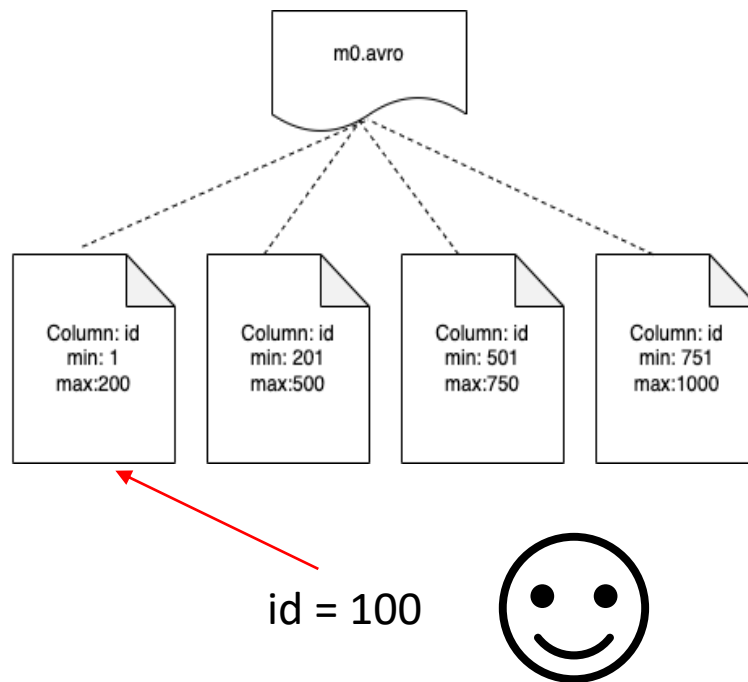
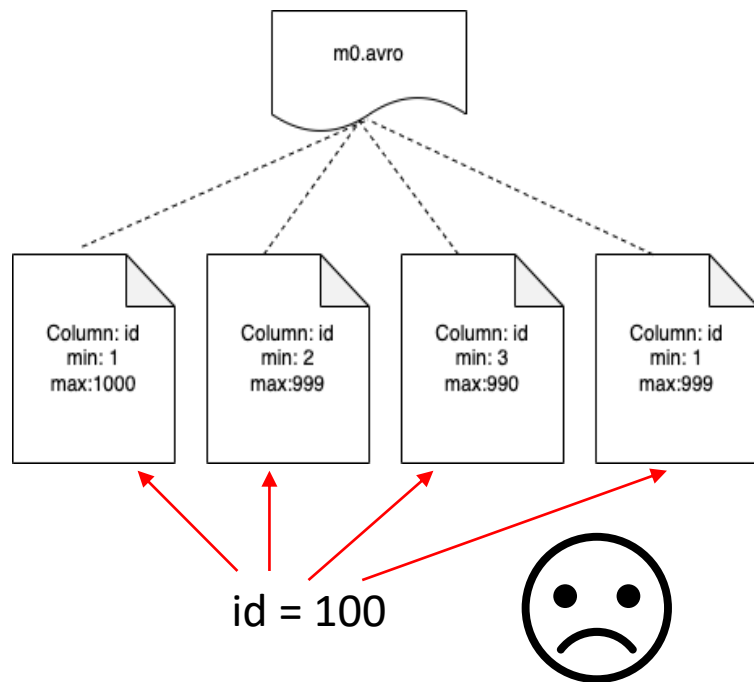


基于 Iceberg 湖仓一体的内核优化

数据组织排序

Iceberg 提供了文件级别元数据

怎么高效利用呢？



排序后 文件根据
字段 id 有很
好的聚集性

数据组织排序

➤ 线性排序

基于一个或多个字段进行分区内线性全排序

```
ALTER TABLE catalog.schema.table  
WRITE DISTRIBUTED BY col1, col2  
WITH RANGE LOCALLY ORDERED BY col1, col2
```

典型应用场景:

根据 up_id/avid 进行点查/范围查询

```
SELECT *  
FROM catalog.schema.table  
WHERE col1 = 'xxx';
```



```
SELECT *  
FROM catalog.schema.table  
WHERE col2 = 'xxx';
```

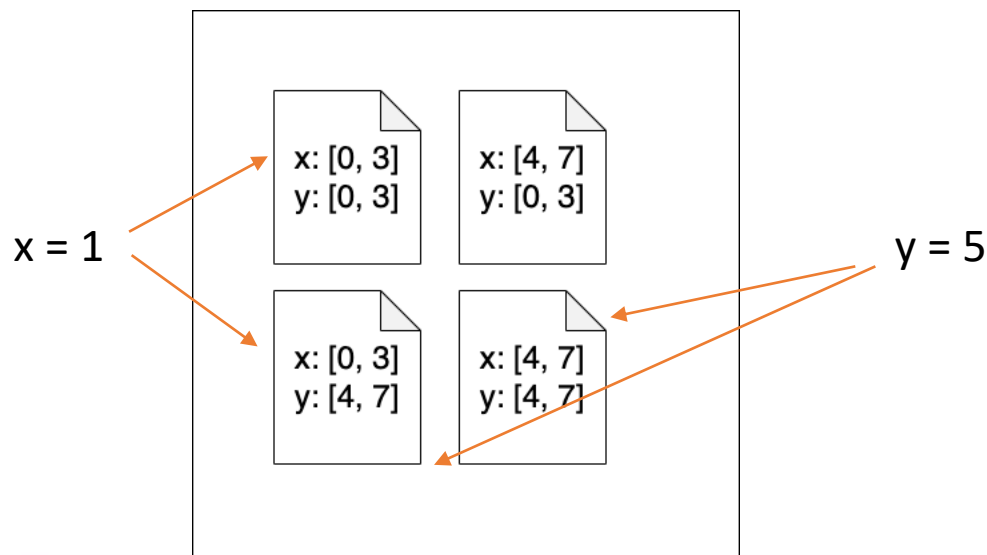


数据组织排序

➤ Z-Order 排序

原理：多个待排序字段，分别按照大小进行二进制编码，每组字段根据编码后的值按 bit 位交错生成一个 Z-Value 值，使得生成的 Z-Value 沿着特定空间轨迹是有序的

	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y:	0	000000	000001	000100	000101	010000	010001	010100	010101
	000								
	1	000010	000011	000110	000111	010010	010011	010110	010111
	001								
	2	001000	001001	001100	001101	011000	011001	011100	011101
	010								
	3	001010	001011	001110	001111	011010	011011	011110	011111
	011								
	4	100000	100001	100100	100101	110000	110001	110100	110101
	100								
	5	100010	100011	100110	100111	110010	110011	110110	110111
	101								
	6	101000	101001	101100	101101	111000	111001	111100	111101
	110								
	7	101010	101011	101110	101111	111010	111011	111110	111111
	111								



每个文件根据 X、Y 都有比较好的聚集性

数据组织排序

➤ Z-Order 排序

基于 2~4 个字段进行分区内 Z-Order 空间排序

```
ALTER TABLE catalog.schema.table  
WRITE DISTRIBUTED BY col1, col2, col3, col4  
WITH ZORDER LOCALLY ORDERED BY col1
```

典型应用场景:

根据 up_id、avid、time 等多个字段进行点查/范围查询

```
SELECT *  
FROM catalog.schema.table  
WHERE col1 = 'xxx';
```

```
SELECT *  
FROM catalog.schema.table  
WHERE col2 = 'xxx';
```



二级索引支持

➤ Bloomfilter 索引

很小的存储开销，支持任意普通类型字段的点查

```
CREATE INDEX coll_bf USING BLOOMFILTER  
ON catalog.schema.table(coll)  
WITH (fpp = 0.01);
```

典型应用场景:

存在较多不同字段过滤场景，适用于相对较高基数的字段

```
SELECT *  
FROM catalog.schema.table  
WHERE coll = 'xxx';
```



```
SELECT *  
FROM catalog.schema.table  
WHERE coll > 'xxx' and coll < 'yyy';
```



二级索引支持

➤ Bitmap 索引

支持多个字段的组合查询，并且支持范围查询

```
CREATE INDEX coll_bm USING BITMAP  
ON catalog.schema.table(coll);
```

典型应用场景:

单个字段 或 多个字段组合的 点查或范围查询，
字段基数不宜过高

```
SELECT *  
FROM catalog.schema.table  
WHERE coll = 'xxx';
```



```
SELECT *  
FROM catalog.schema.table  
WHERE coll > 'xxx' and coll < 'yyy';
```



二级索引支持

➤ BloomRF 索引

较小的存储开销，支持字段的点查和范围查询

```
CREATE INDEX coll_brf USING BLOOMRF  
ON catalog.schema.table(coll);
```

典型应用场景:

高基数字段的点查和范围查询

```
SELECT *  
FROM catalog.schema.table  
WHERE coll = 'xxx';
```



```
SELECT *  
FROM catalog.schema.table  
WHERE coll > 'xxx' and coll < 'yyy';
```



二级索引支持

➤ Tokenbf、ngrambf 索引

针对 String 类型数据进行检索

```
CREATE INDEX coll_tbf USING TOKENBF  
ON catalog.schema.table(coll);
```

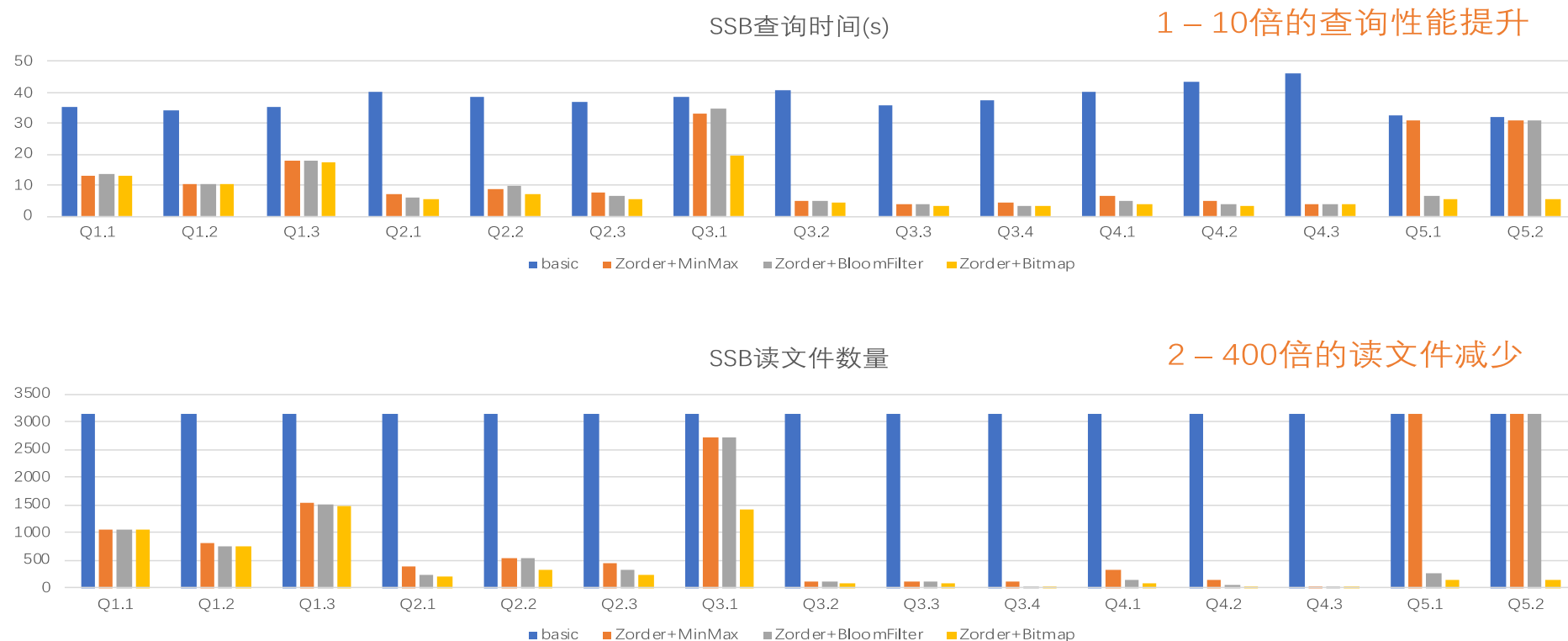
典型应用场景:

日志场景中的关键字检索

```
SELECT *  
FROM catalog.schema.table  
WHERE has_token(coll) = 'xxx';
```



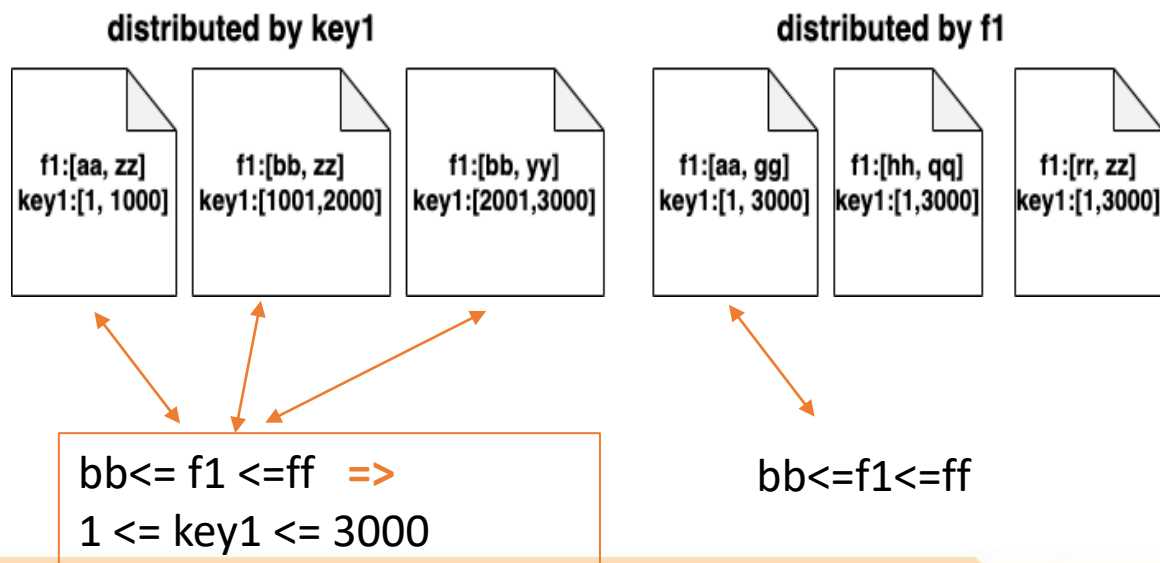
优化效果（SSB 1000 大宽表模式）



预计算、星型模型支持

➤ 星型模型

Join 场景下 根据维表过滤字段 组织数据，
谓词下推



```
ALTER TABLE T1  
ADD CORRELATED COLUMN (T2.f1, T2.f2)  
FROM (LEFT JOIN T2 ON T1.key1 = T2.key2  
WITH unique)
```

```
ALTER TABLE T1  
WRITE DISTRIBUTED BY key1, f1, f2  
WITH ZORDER;
```

```
SELECT *  
FROM T1 left join T2 on t1.key1 = t2.key2  
WHERE t2.f1 = 'xxx';
```



预计算、星型模型支持

➤ 预计算

预计算解决聚合、多表关联场景下大量数据摄入、
计算密集导致查询耗时的问题

支持多种聚合函数：count、avg、max、min、
sum、count_distinct、approx_count_distinct、
percentile、top_n

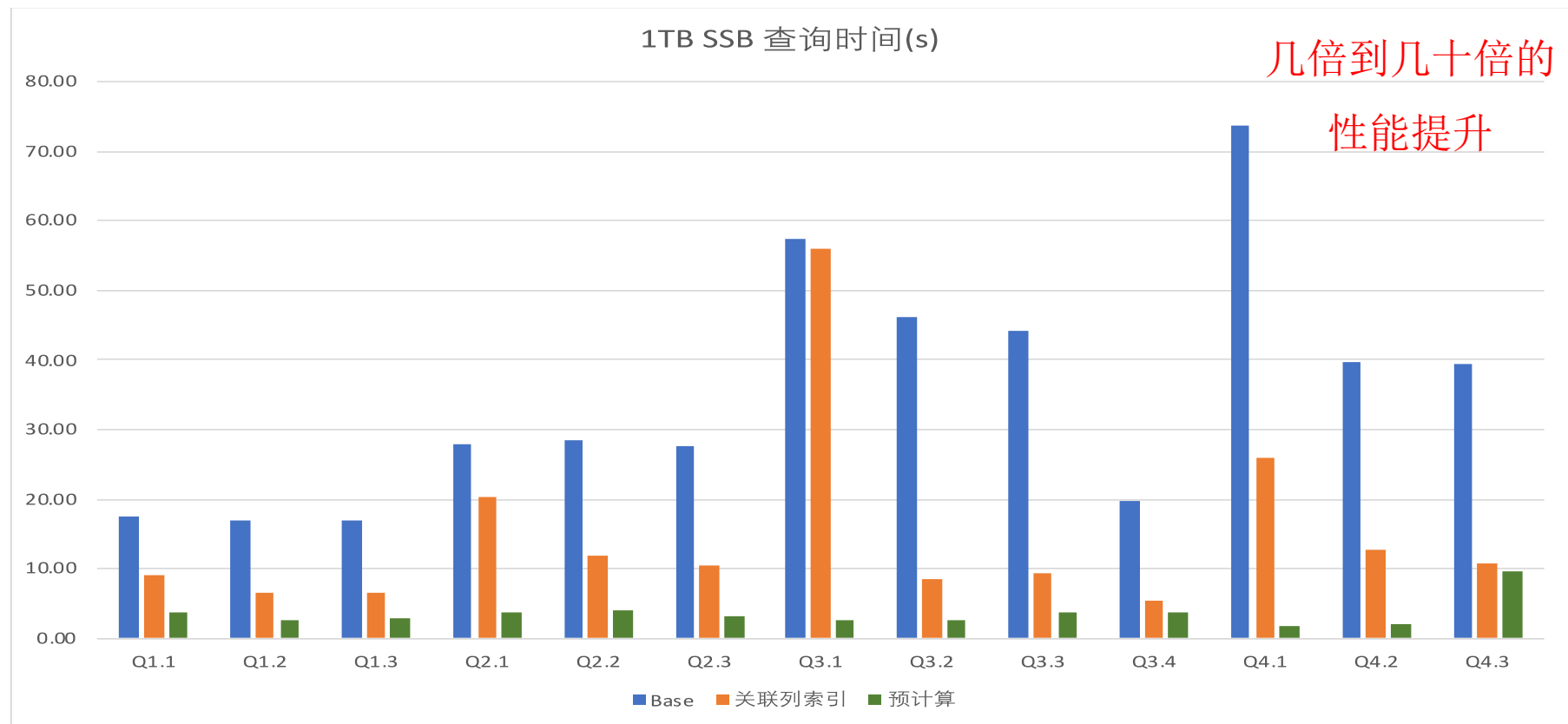
支持查询部分 cube 文件、部分数据文件

```
CREATE AGG_INDEX cube1 on T1
WITH DIMENSIONS (T2.f1, T2.f2)
WITH AGGREGATIONS (sum(T1.f3))
```

```
SELECT SUM(T1.f3), f1
FROM T1 left join T2 on t1.key1 = t2.key2
WHERE t2.f1 = 'xxx'
group by f1;
```

```
SELECT SUM(cube1.f3), cube1.f1
FROM cube1
WHERE t2.f1 = 'xxx'
group by f1;
```

优化效果



小结

内核优化核心：只读取查询所需的数据、尽量减少无用数据的摄入



智能化管理平台 Magnus

智能化管理平台 Magnus 的背景

- Iceberg 只提供了数据与元数据读、写 API 的实现，缺乏统一的数据、元数据操作、管理的服务
- 用户习惯了使用 Hive 表，怎么减少用户使用 Iceberg 表的成本，使用起来跟普通的 Hive 表一样

数据分布信息 帮助

文件间排序字段(Distribute By): 请选择

文件内排序字段(Sort By): 请选择

数据索引配置 帮助

index	字段名称	索引类型	fpp参数	操作
0	请选择字段名称	请选择索引类型	请输入数值	+ -

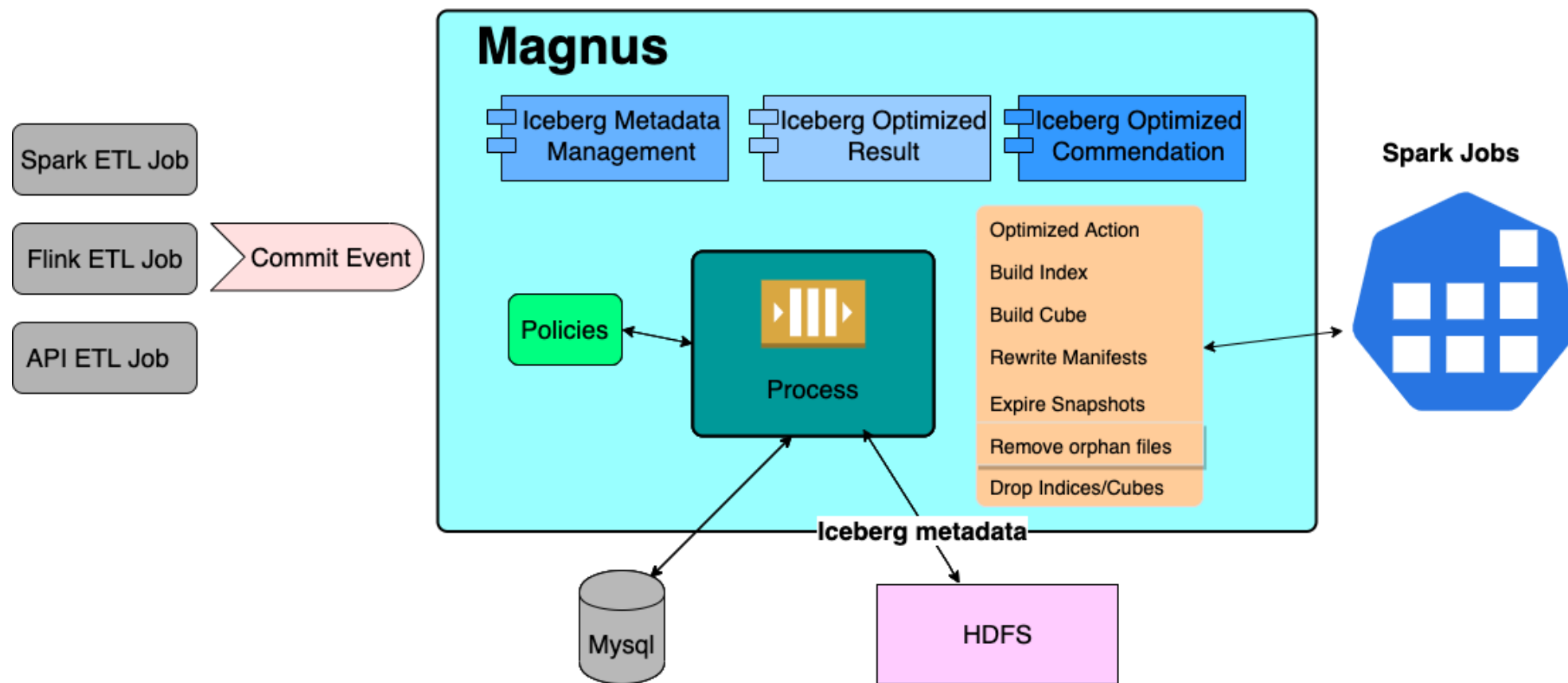
不确定常用查询 Pattern

定义不合理

查询模式随时间变化

数据组织、索引如何触发

智能化管理平台 Magnus 架构



智能化管理平台的实践

➤ Iceberg 元数据管理平台

Magnus Web

Schema

Table

Snapshot

Manifest

Partition

Optimized Detail

Optimized Summary

Analysis

Job

Job Action

Configuration

test_iceberg_bdp iceberg_bls_test01 search

test_iceberg_bdp.iceberg_bls_test01

Snapshot nums: 11, current: 6039100617355963000 Sorted By identity(log_level) ASC NULLS FIRST, identity(time) AS

Partitioned By identity(log_date) Distributed By HIBERT identity(time), identity(log_trace_id), identity(lo

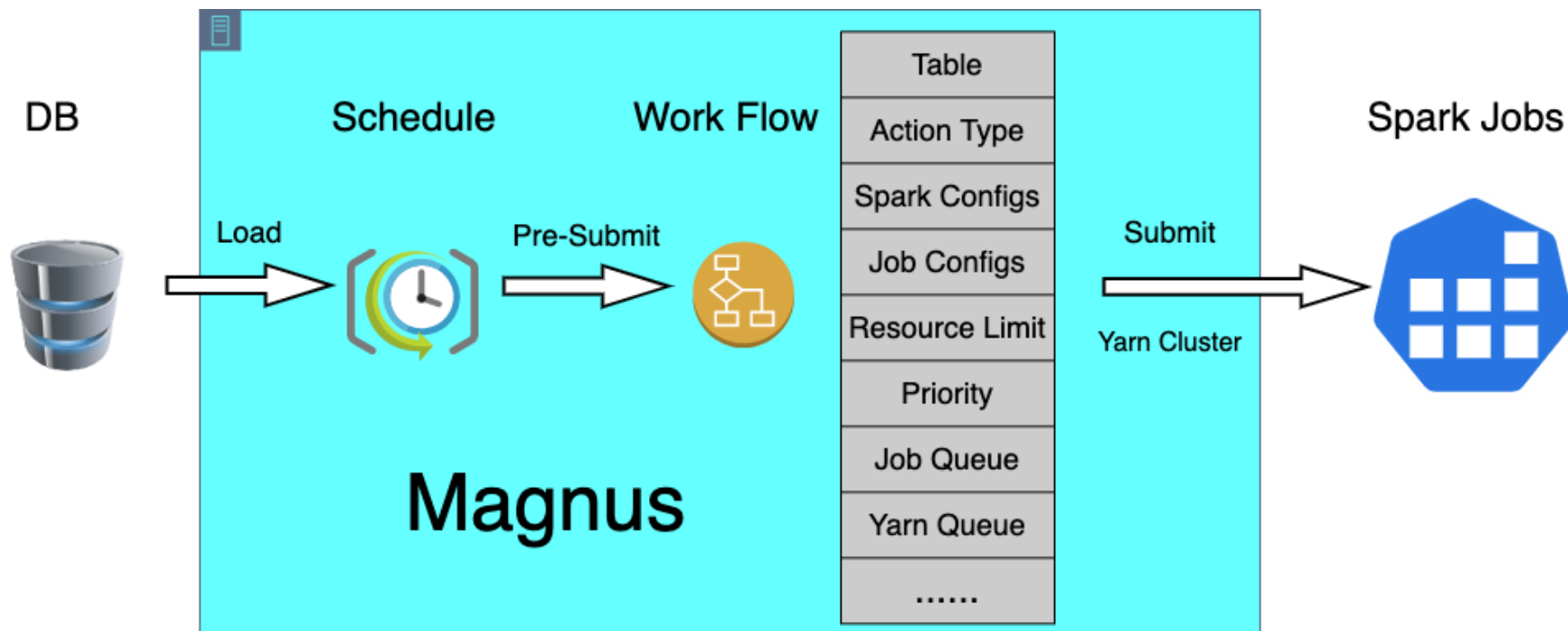
Location viewfs://warehouse/iceberg_bls_test01

Description

Column	Correlation	Index	Aggregation Index	Snapshot Summary	Properties
Id	Name	Type	Required	Desc	
1	time	long	false	time	
2	instance_id	string	false	instance_id	
3	log	string	false	log	
4	log_date	string	false	日期分区	

智能化管理平台的实践

➤ 智能化数据组织优化



智能化管理平台的实践

➤ Table 优化详情视图

iceberg_bdp

Partition Name

Operator

Partition Value

Q

search

Sort

SpecId: 1

identity(buvid) ASC NULLS FIRST

Distribution

SpecId: 1

HIBERT identity(platform), identity(app_id), identity(is_new_user), identity(buvid)

Index

SpecId: 6

id=1, name=index_BITMAP_ver, type=BITMAP, column=ver
id=2, name=index_BLOOMFILTER_brand, type=BLOOMFILTER, column=brand
id=3, name=index_BLOOMFILTER_buvid, type=BLOOMFILTER, column=buvid
id=4, name=index_BLOOMFILTER_chid, type=BLOOMFILTER, column=chid
id=5, name=index_BLOOMFILTER_city, type=BLOOMFILTER, column=city
id=6, name=index_BLOOMFILTER_model, type=BLOOMFILTER, column=model

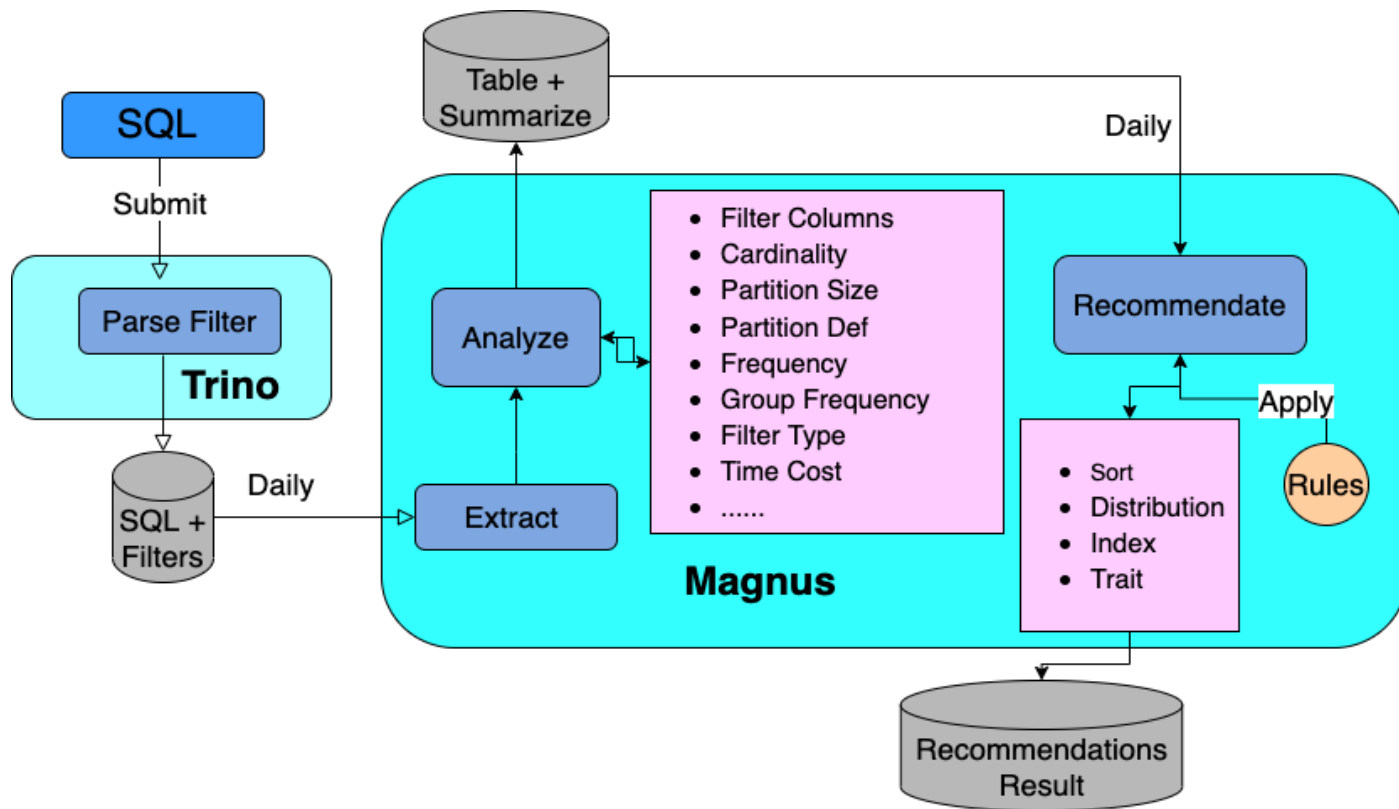
AggIndex

SpecId: 0

	Partition Name	Partition Value	Total Files	Distribution Details	Sort Details	Index Details	AggIndex Detail
>	log_date	20221126	10	[Id:1, 10/10]	[Id:1, 10/10]	[Id:1, 10/10] [Id:2, 10/10] [Id:3, 10/10] [Id:4, 10/10] [Id:5, 10/10] [Id:6, 10/10]	NONE

智能化管理平台的实践

➤ 智能查询分析、优化模式推荐



默认检测 (边界: 小表)

权重 (基数、过滤百分比、查询耗时 等)

智能化管理平台的实践

➤ 智能查询分析、优化模式推荐

The screenshot displays the Magnus Web interface. On the left is a dark sidebar with navigation items: Magnus Web, Schema, Table, Snapshot, Manifest, Partition, Optimized Detail, Optimized Summary, Analysis (highlighted in blue), Job, and Job Action. The main content area shows a search bar with 'iceberg_bdp' and a 'search' button. Below this are tabs for 'Overview' and 'Recommendation-440'. The 'Recommendation-440' tab is active, displaying a table of optimization parameters and statistics.

Start Time	End Time	Target Date
2022-11-28 15:32:55	2022-11-28 15:33:36	2022-11-21 to 2022-11-28
Partitions for Stats	log_date>=20221125	
Parameters	{}	
Error Message		
Recommended Sort	like_rank, avid	
Recommended Distribution	avid, up_id, like_rank	
Recommended Indices	avid: BLOOMFILTER, up_id: BLOOMFILTER, like_rank: BITMAP	
Partition Statistics	fileNum: {avg: 1652, min: 1091, max: 2543} fileSize: {avg: 321.4GB, min: 258.1GB, max: 387.8GB} recordCount: {avg: 6954883651, min: 6021979490, max: 7889557338}	

智能化管理平台的实践

➤ 智能查询分析、优化模式推荐

Column Statistics

Type	#Queries				Cardinality		
	Total	Filter	Equality Filter	Range Filter	Avg	Max	Min
long	69	67 (97.10%)	67 (97.10%)	0 (0.00%)			
long	69	67 (97.10%)	0 (0.00%)	67 (97.10%)			
long	69	66 (95.65%)	66 (95.65%)	0 (0.00%)			

Pattern Statistics

Pattern Type	Pattern Desc	#Queries	Avg. Exec. Time (s)
ALL		69	2.116
FILTER	=[EQUALITY], =[RANGE], =[EQUALITY]}	66	0.965
FILTER	=[EQUALITY], =[RANGE]}	1	0.470

智能化管理平台的实践

小结

Magnus 核心价值：助力查询加速、降低用户使用湖仓门槛



未来规划

未来规划

- 预计算 支持 Star Tree
- 历史分析推荐 支持预计算
- 历史分析推荐实现 智能化
- CDC 数据落地支持

感谢收看



欢迎关注
哔哩哔哩技术
公众号

哔哩哔哩技术

微信扫描二维码，关注我的公众号

THANKS

SQL Server
vertica
D B 2
G B a s e
O r a c l e
达梦数据库
神舟通用
KingbaseES

2010

2014

2018

openGauss
OceanBase
ArkDB
RASESQL
HotDB
StellarDB
QianBase xTP
云树Shard
GoldenDB
DolphinDB
MatrixDB
DynamoDB
SinoDB
FastData
Galaxybase
KunDB
GDB
GaussDB
PolarDB
KunDB
Spacture
SequoiaDB
OushuDB
ArgoDB
开务数据库
GreatDB
MongoDB
TDSQL
TiDB
Tapdata
StarRocks
UbiSQL