FLINK FORWARD #ASIA 2021 ONLINE

实时即未来 REAL-TIME IS THE FUTURE

Apache Flink 集成 Apache Iceberg 最佳实践

胡争 Apache Iceberg & HBase PMC

FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

#1 Hive表面临的挑战

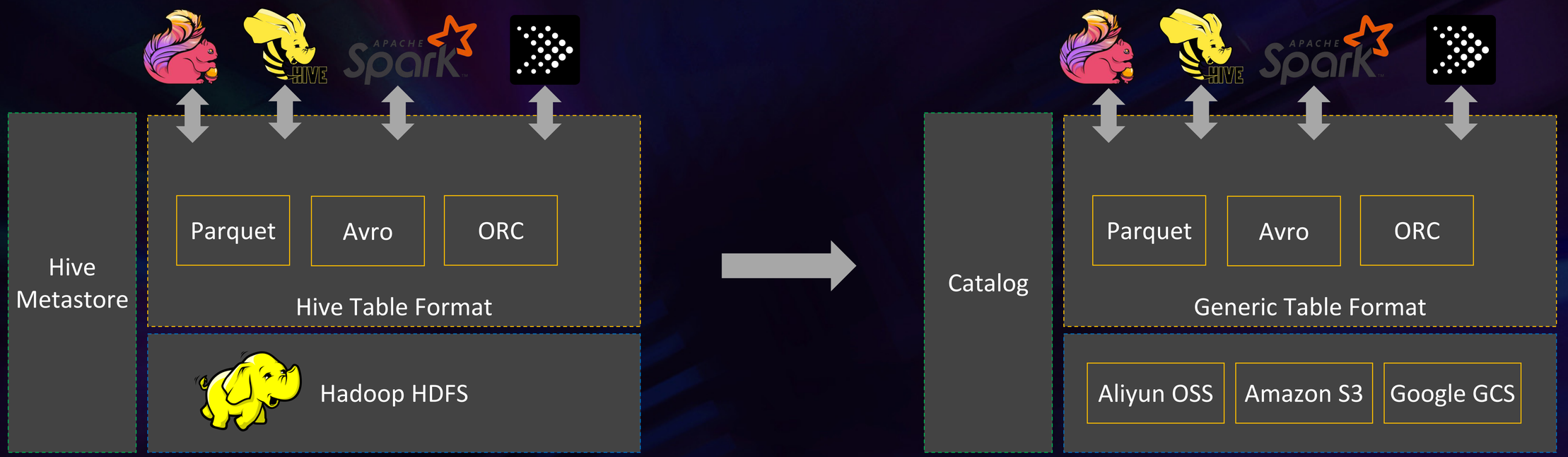
#2 Iceberg的解决方案

#3 Flink和Iceberg最佳实践

#4 现状及规划

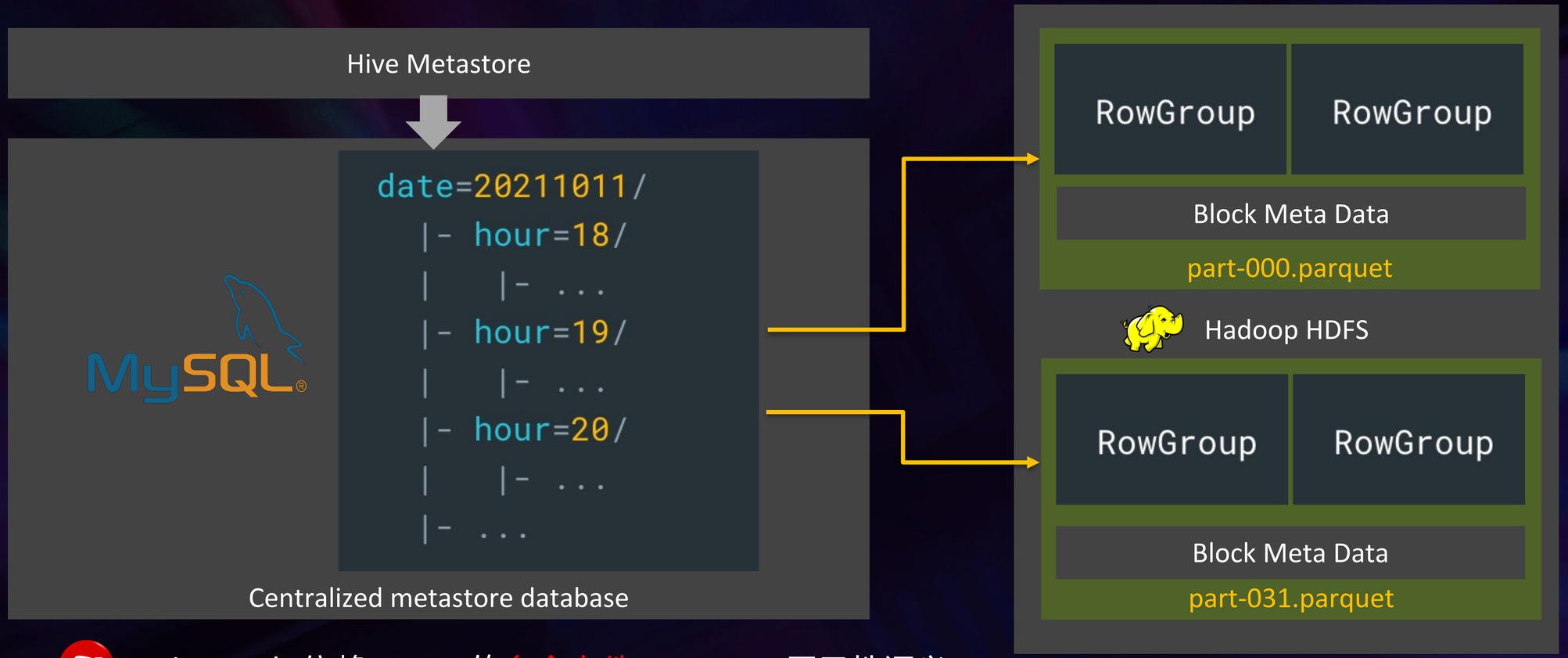
#1 Hive 表格式面临的挑战

挑战#1：上云



- HMS 信息冗杂(Schema/表级统计信息/分区信息等)，边界不清。无法扩展到云厂商 Catalog 服务。
- HMS 存储中心化，扩展性差。
- HDFS 成本高，缺乏弹性。
- Hive 表格式抽象不清晰，暴露太多差异化细节给上层。

挑战#1：上云



- Write Path 依赖 HDFS 的多个文件 RENAME 原子性语义
- Read Path 先查 MySQL 获取分区列表，再 LIST 目录获取文件
- 中心化的 Metastore 数据库，扩展性差
- 表级统计信息更新不及时，缺乏有效的文件统计信息

挑战#1：上云

要求一

支持多种对象存储

特点：弹性、低廉、稳定

要求二

统一的Table语义

抽象度高，ACID，多种文件格式

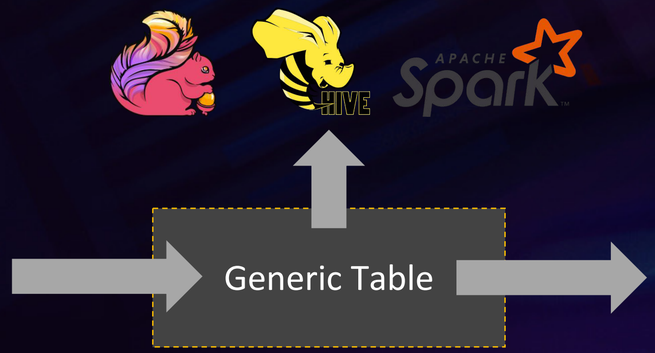
要求三

计算引擎互连互通

支持Hive,Spark,Flink,Presto读写

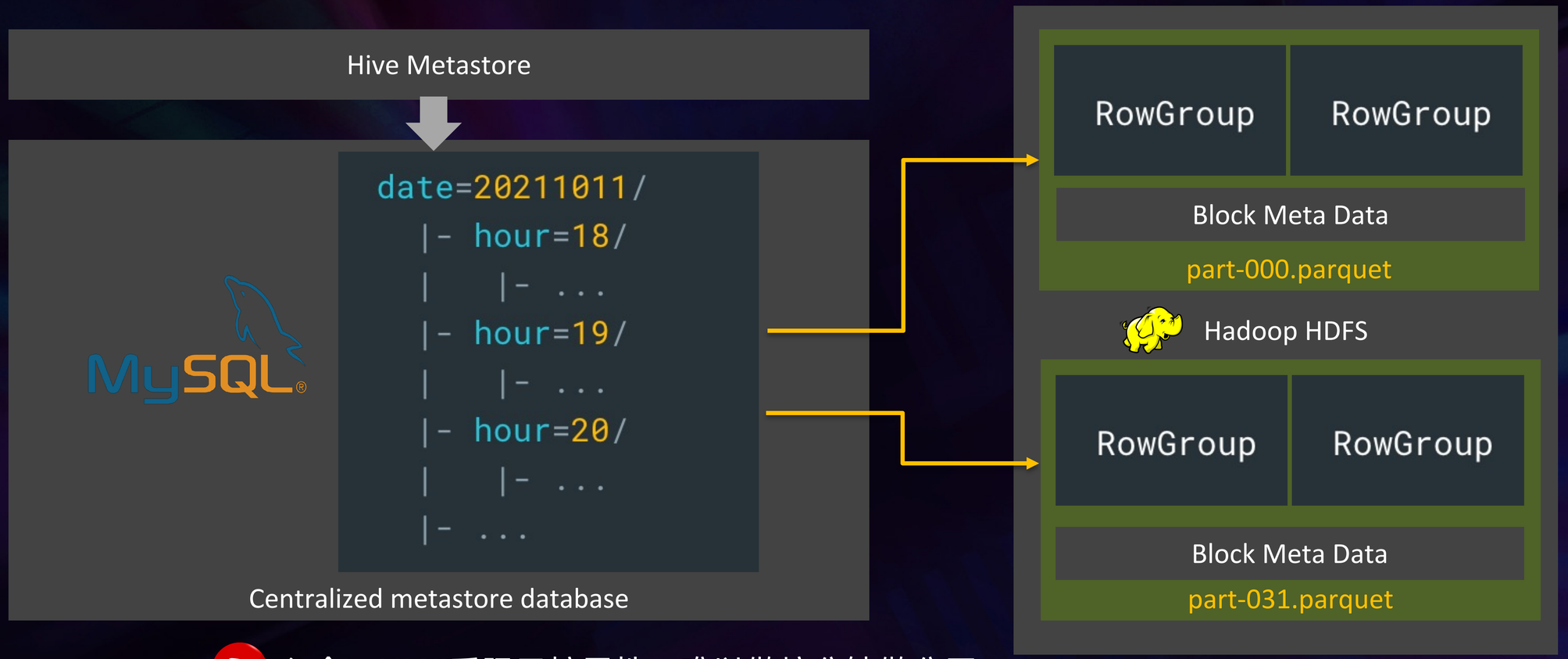
挑战#2: 近实时数仓

小时级时效性体验



分钟时级时效性体验

挑战#2：近实时数仓



- 入仓：HMS受限于扩展性，难以做按分钟做分区
- 查询：先查MYSQL找分区，再list分区目录找文件，元数据index效率低
- 查询：缺乏文件级全局统计信息
- 出仓：不支持增量数据查询

挑战#2: 近实时数仓

要求一

分钟级入湖入仓

湖仓内数据更实时

要求二

更高效索引加速数据分析

查询响应更快

要求三

增量出湖出仓

下游ETL响应更快

挑战#3：变更

问题一：Schema 变更（如新增一个字段）

ID	Name
1001	Alex
1002	Bob

ID	Name	Address
1001	Alex	BJ
1002	Bob	SH

挑战#3: 变更

问题二：分区变更 (从月级分区改成天级分区)

挑战#3：变更

问题三：CDC 数据变更

挑战#3: 变更

要求一

Schema 变更

表结构随业务变动而变更

要求二

分区变更

调整分区策略适配不同分析诉求

要求三

数据变更

表级/分区级/文件级/行级不同粒度变更

#2 Iceberg 的解决方案

Iceberg 数据湖系统架构

阿里云EMR

MaxCompute

Hologres

DLF Catalog

- Database
- Import Data
- ETL
- Authority
- Security
- Lake SaaS

ICEBERG

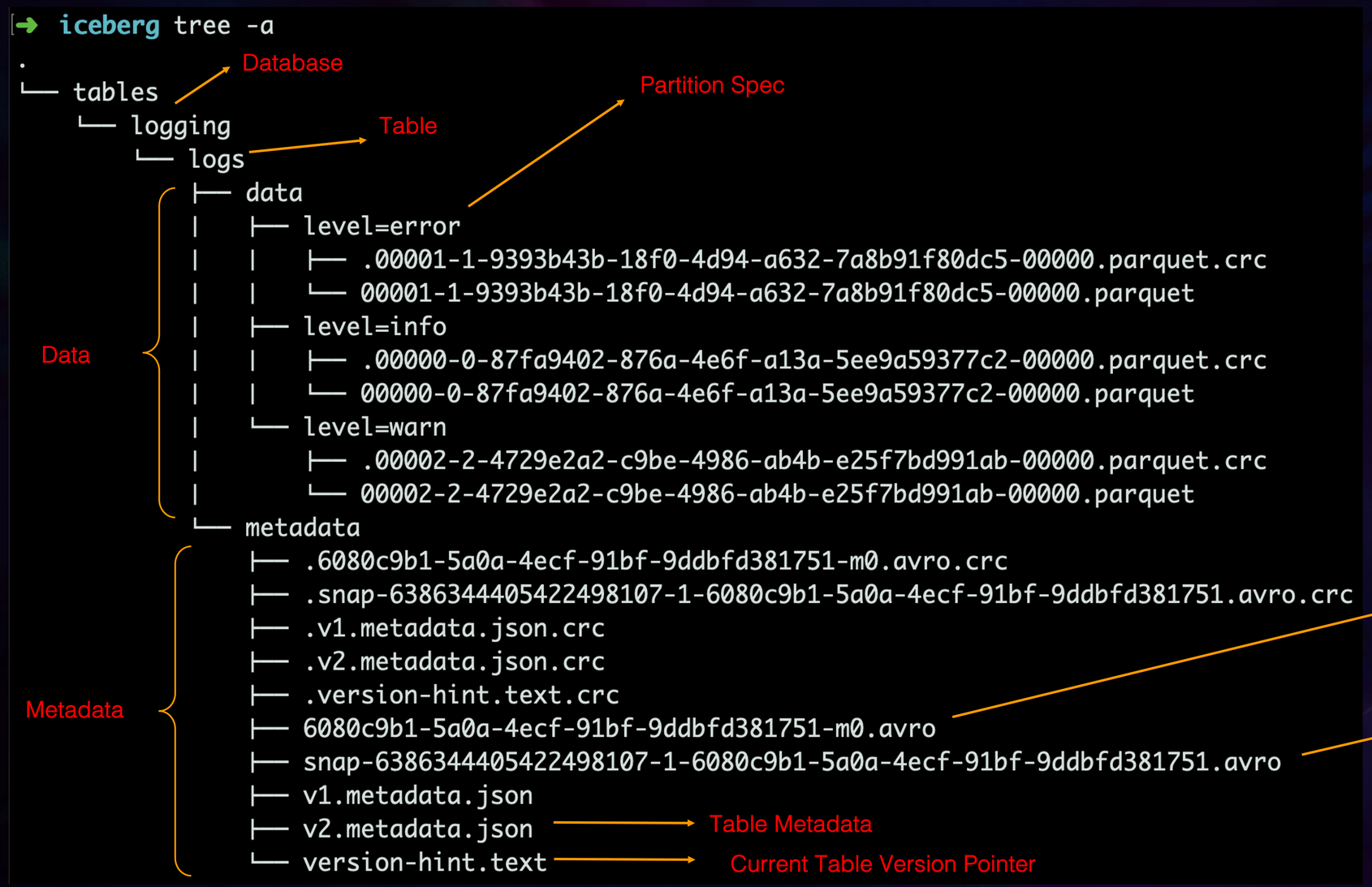
JindoFs Cache

Aliyun OSS

Data Lake Manager

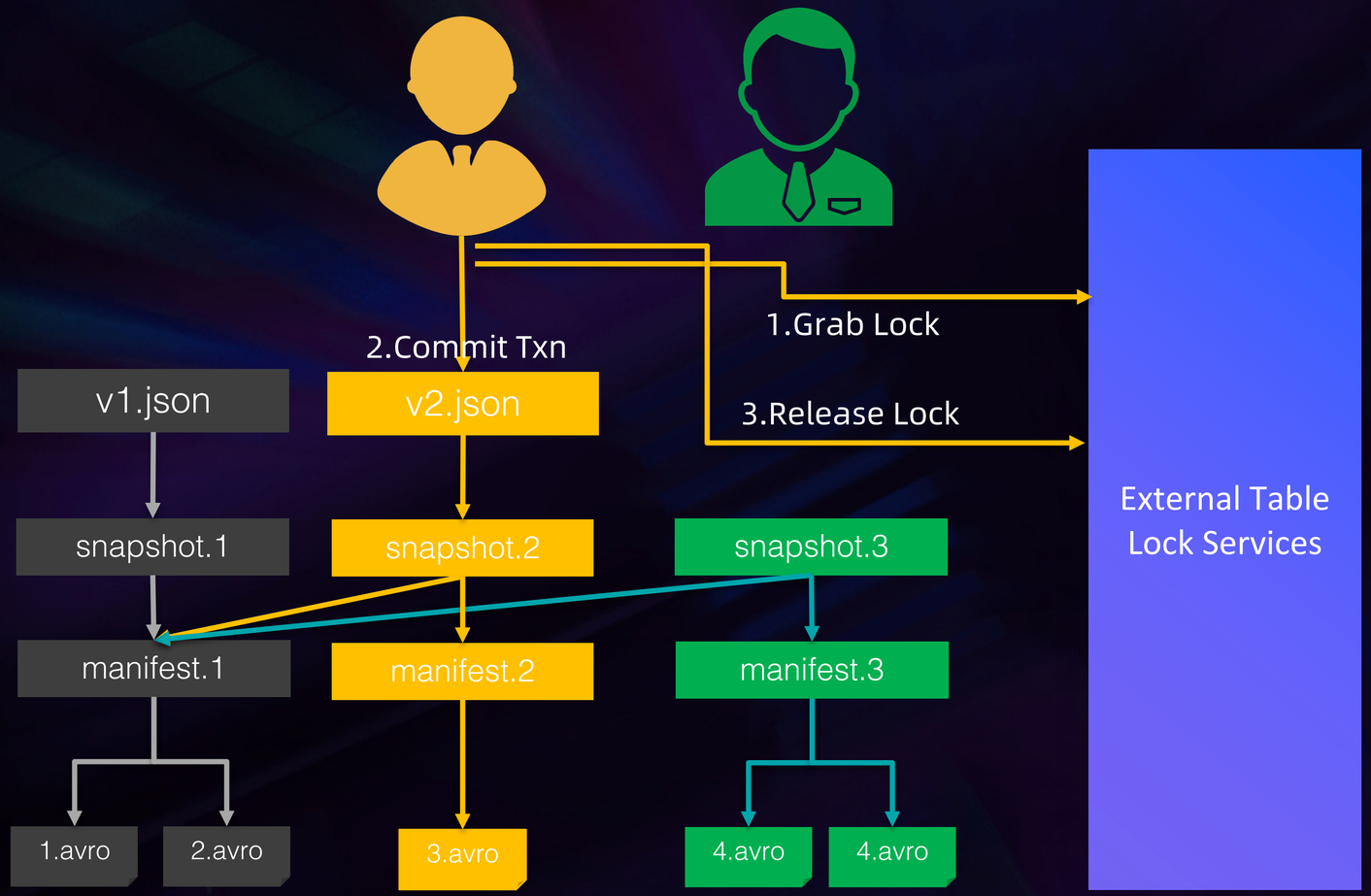
- Compaction
- Data Sort
- Expire Data
- Snapshots
- Metadata
- Metrics

挑战#1：上云



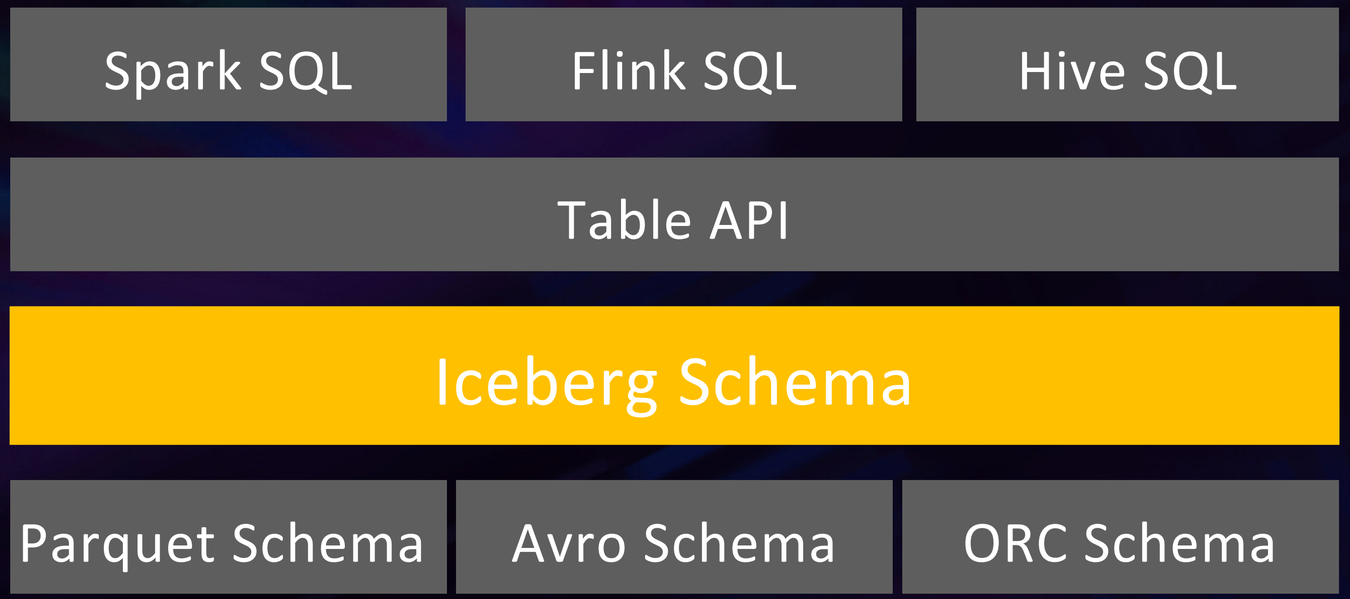
- 数据访问不使用任何LIST接口
- 可扩展的 metadata 存储

挑战#1：上云



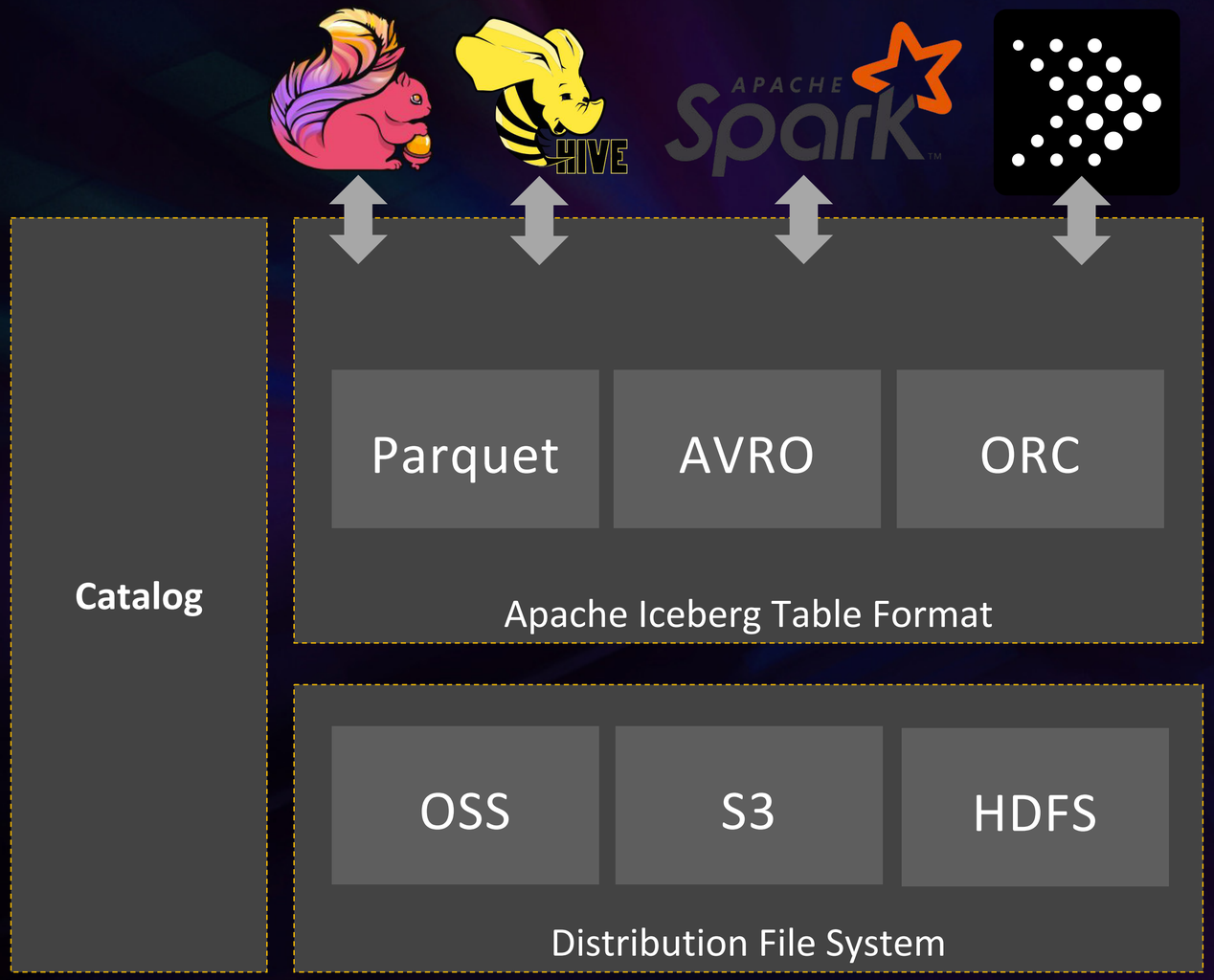
ACID不依赖 RENAME 接口

挑战#1：上云



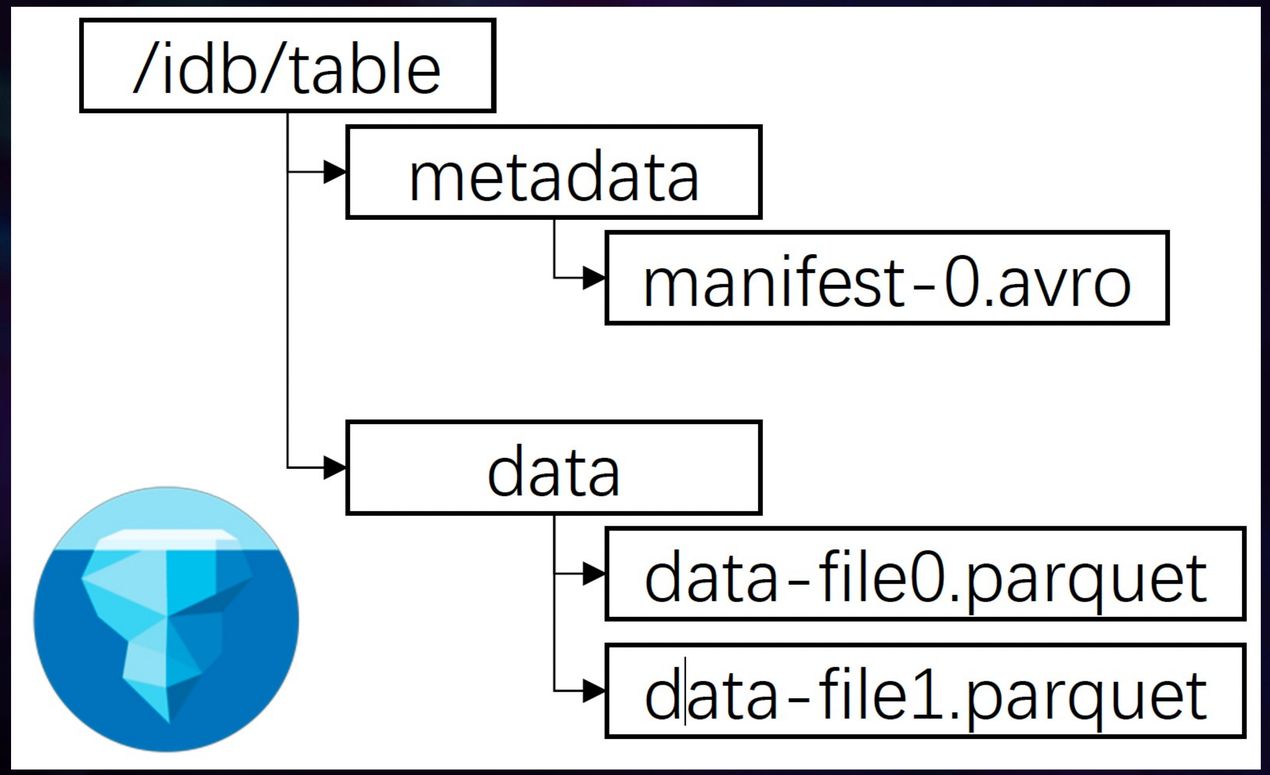
统一的 Table 语义

挑战#1：上云



完善的计算和多云生态对接

挑战#2: 近实时数仓



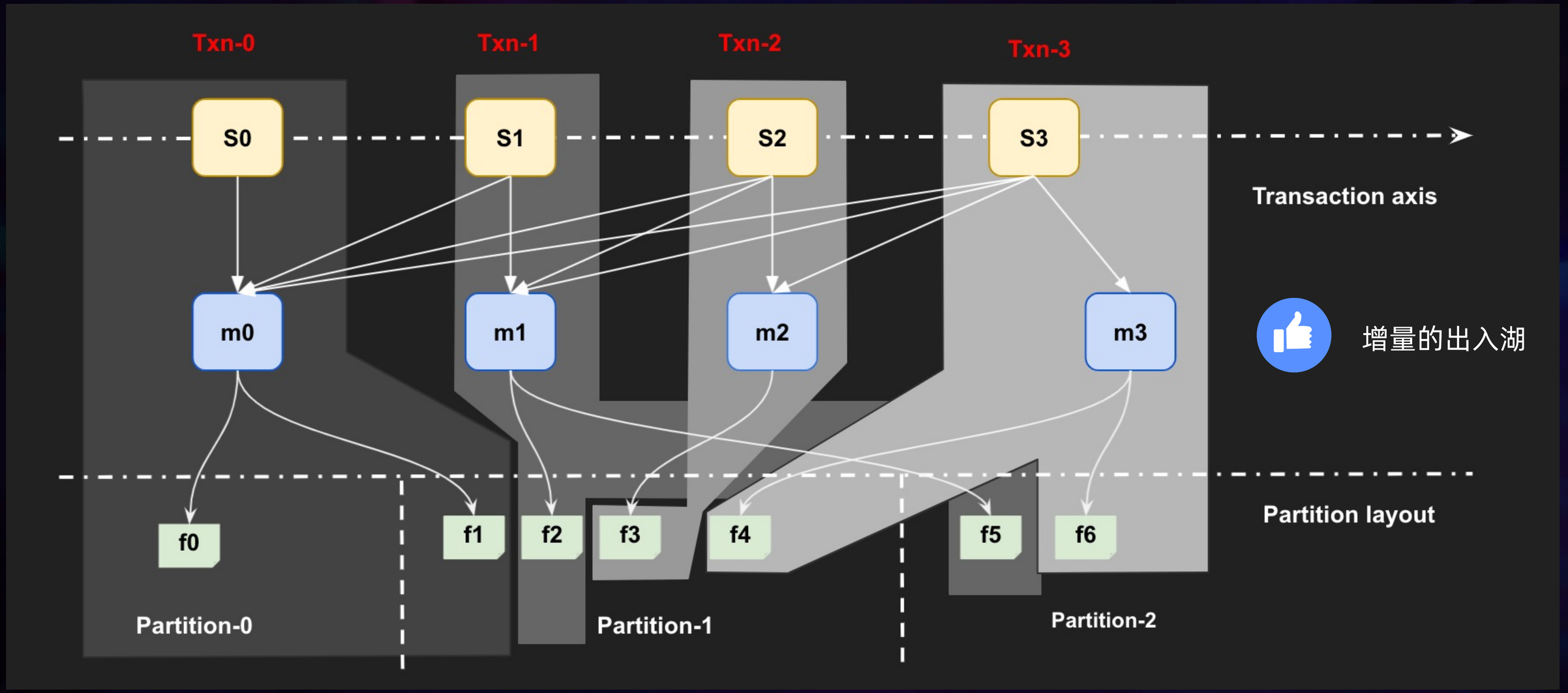
去中心化可拓展的 metadata

挑战#2: 近实时数仓

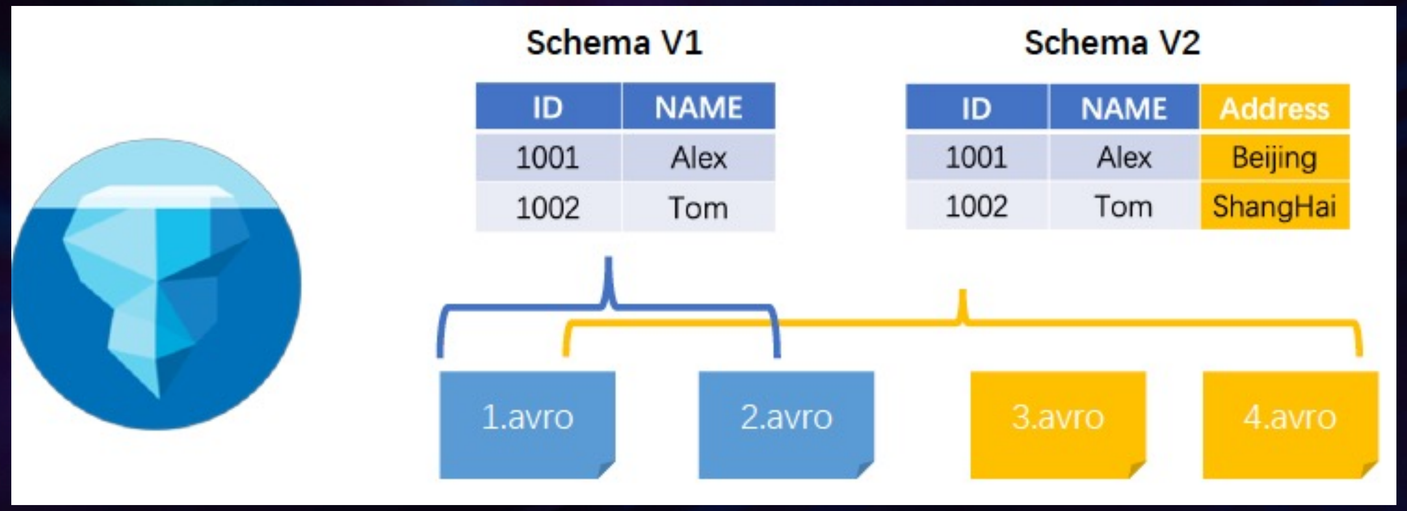
file_path	partition		lower_bounds		upper_bounds	
			device_id	event_time	device_id	event_time
01-data.parquet	2021-10-01	0	299	…	959,446	…
02-data.parquet	2021-10-01	1	186	…	960,724	…
…	…	…	…	…	…	…
64-data.parquet	2021-10-02	0	357	…	962,984	…
65-data.parquet	2021-10-02	1	65	…	959,875	…
…	…	…	…	…	…	…

丰富的 metadata index 加速

挑战#2: 近实时数仓

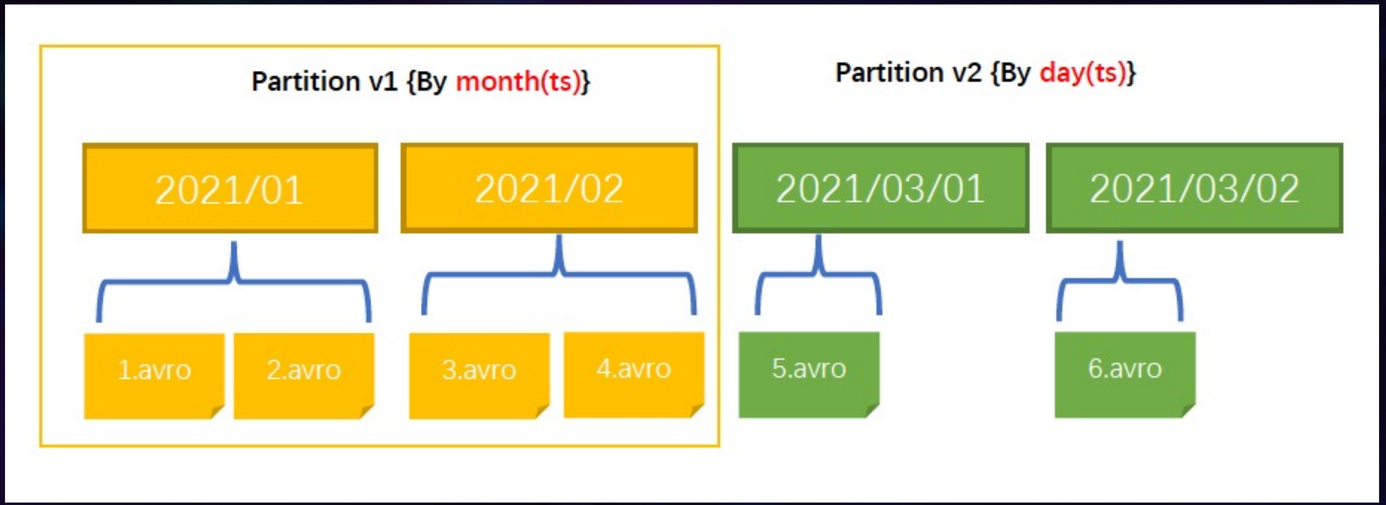

Txn-0
Txn-1
Txn-2
Txn-3
S0
S1
S2
S3
Transaction axis
m0
m1
m2
m3
增量的出入湖
f0
f1
f2
f3
f4
f5
f6
Partition layout
Partition-0
Partition-1
Partition-2

挑战#3：变更



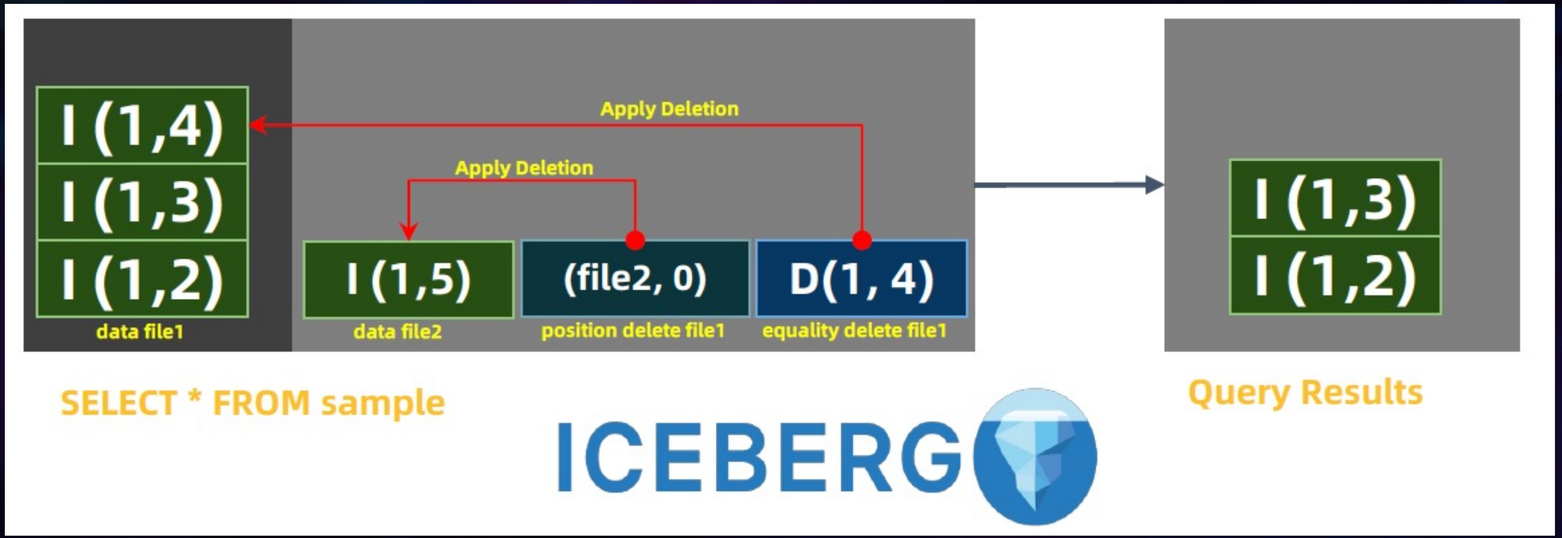
快速实现 Schema 变更

挑战#3：变更



轻量级分区变更

挑战#3：变更



V2支持 Merge-On-Read 方式更新数据

FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

#3 Flink 和 Iceberg 最佳实践

Flink写入filesystem表，还是Iceberg表？

FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

#1 Flink写入 FileSystem 表之后，不能被其他计算引擎直接读取？

#2 Flink写入到 FileSystem 的表，怎么实现 schema 变更、分区变更，数据变更？

#3 Flink写入到 FileSystem 的表，怎么存放在 OSS 以及更多云存储之上？

#4 Flink写入到 FileSystem 的表，如何追溯历史版本？

#5 Flink写入到 FileSystem 的表，如何实现增量拉取？

Flink写入filesystem表，还是Iceberg表？

#1 Flink写入 FileSystem 表之后，不能被其他计算引擎直接读取？

写Iceberg表，遵循Iceberg标准协议。Hive,Presto,Spark,Flink可正常读写。

#2 Flink写入到 FileSystem 的表，怎么实现 schema 变更、分区变更，数据变更？

Iceberg在ACID之上支持各种DDL变更和DML变更。

#3 Flink写入到 FileSystem 的表，怎么存放在 OSS 以及更多云存储之上？

Iceberg基于对象存储fs语义构建，社区支持HDFS/S3/aliyun-oss等异构存储服务。

#4 Flink写入到 FileSystem 的表，如何追溯历史版本？

Iceberg表自动维护历史版本，轻松实现历史追溯。

#5 Flink写入到 FileSystem 的表，如何实现增量拉取？

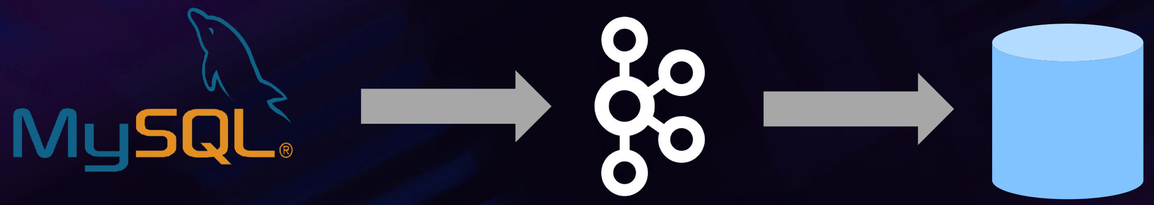
Iceberg表相邻两Snapshot之差及增量，纯粹借助元数据实现增量数据拉取。

Apache Iceberg 0.13.0 Quick Start

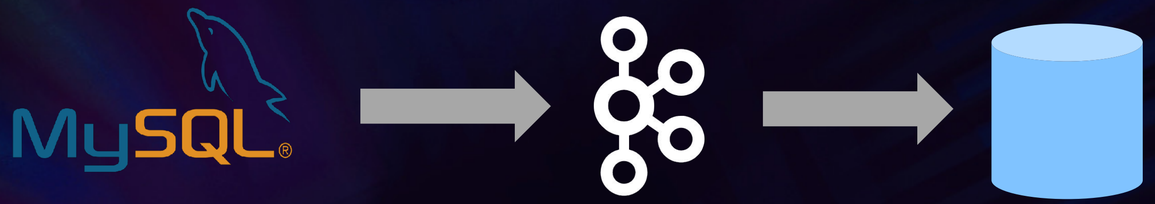

```
-- Open the Flink SQL client.
-- ./bin/sql-client.sh embedded \
--    -j /path/to/iceberg-flink-1.13-runtime-0.13.0.jar \
--    -j /path/to/flink-sql-connector-hive-2.3.6_2.12-1.13.2.jar \
--    shell

CREATE TABLE iceberg_oss(
  id BIGINT,
  data STRING
) WITH (
  'connector' = 'iceberg',
  'catalog-name' = 'hive_prod',
  'uri' = 'thrift://localhost:9083',
  'engine.hive.enabled' = 'true',
  'location' = 'oss://iceberg/warehouse',
  'io-impl' = 'org.apache.iceberg.aliyun.oss.OSSFileIO',
  'access.key.id' = '******',
  'access.key.secret' = '******',
  'oss.endpoint' = 'oss-cn-hangzhou.aliyuncs.com'
);
```

```
DESC iceberg_oss;
+------+--------+------+-----+--------+-----------+
| name |   type | null | key | extras | watermark |
+------+--------+------+-----+--------+-----------+
|   id | BIGINT | true |     |        |           |
| data | STRING | true |     |        |           |
+------+--------+------+-----+--------+-----------+
2 rows in set

INSERT INTO iceberg_oss VALUES
   (1, 'AAA'),
   (2, 'BBB'),
   (3, 'CCC');

SELECT * FROM iceberg_oss;
+----+------+
| id | data |
+----+------+
|  1 |  AAA |
|  2 |  BBB |
|  3 |  CCC |
+----+------+
3 rows in set
```

MySQL数据如何实时同步到OSS?

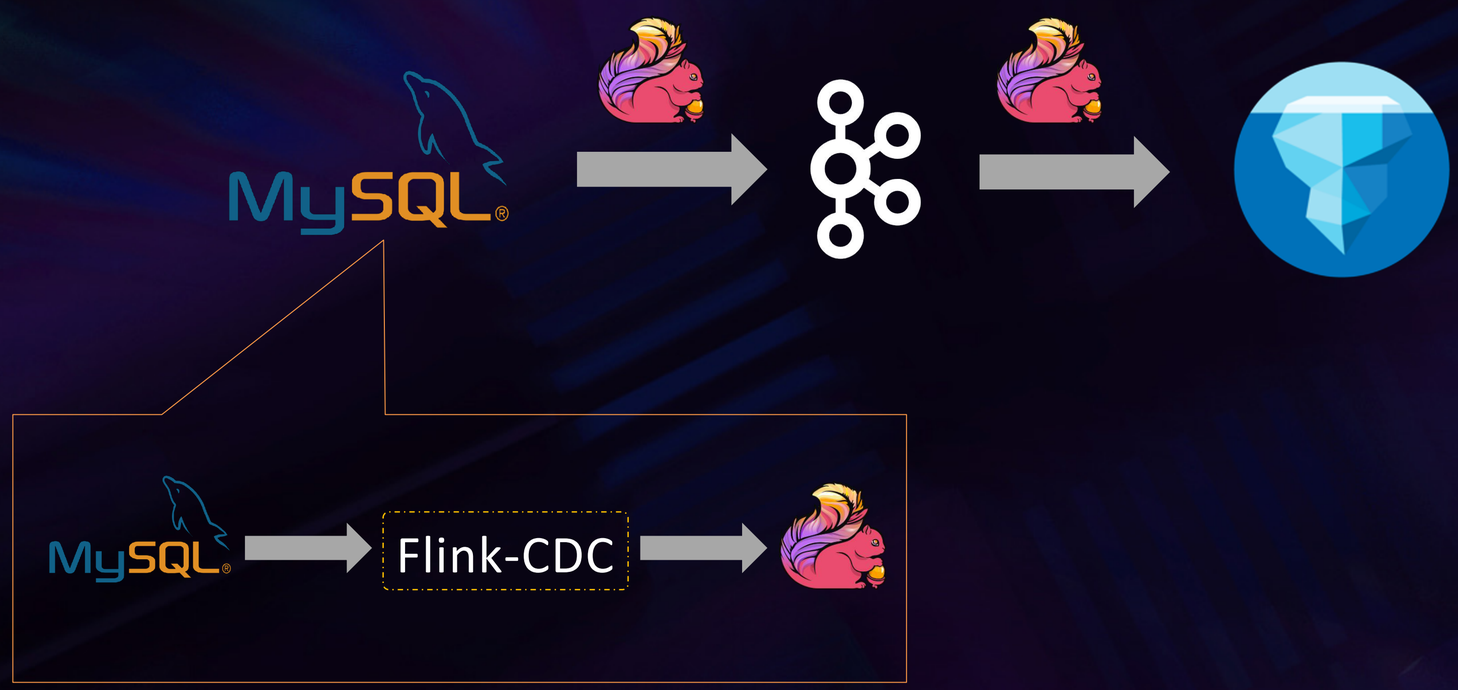


MySQL数据如何实时同步到OSS?

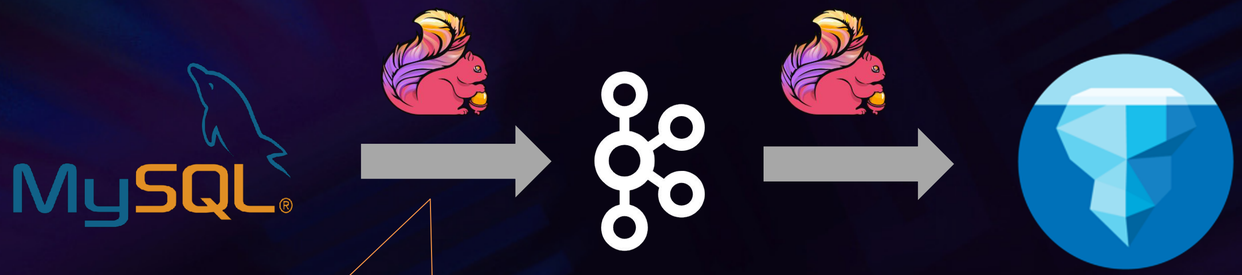


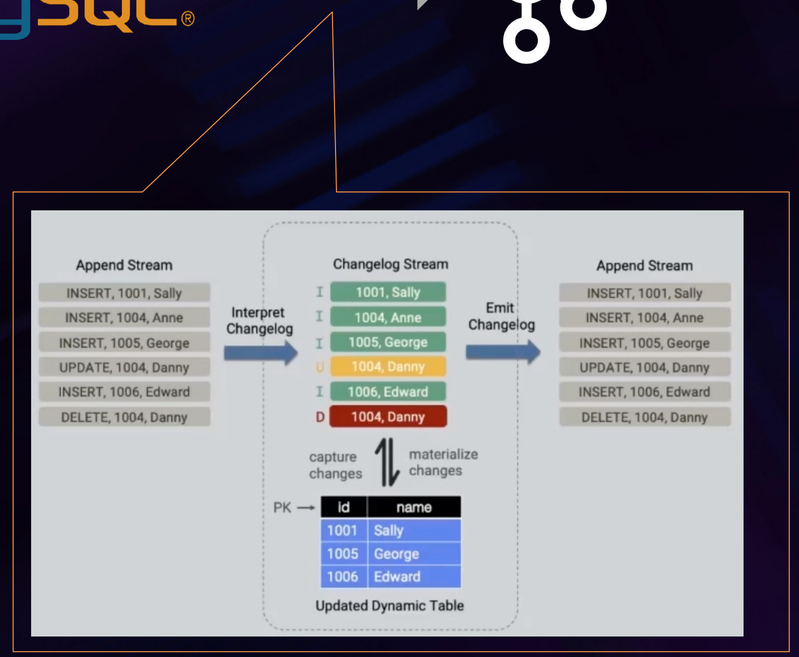
- MySQL全量和增量如何完美对齐?
- Binlog不丢不重地入湖?
- 代码开发? 门槛太高?
- 没有合适的列存存储维护变更?

MySQL数据如何实时同步到OSS?

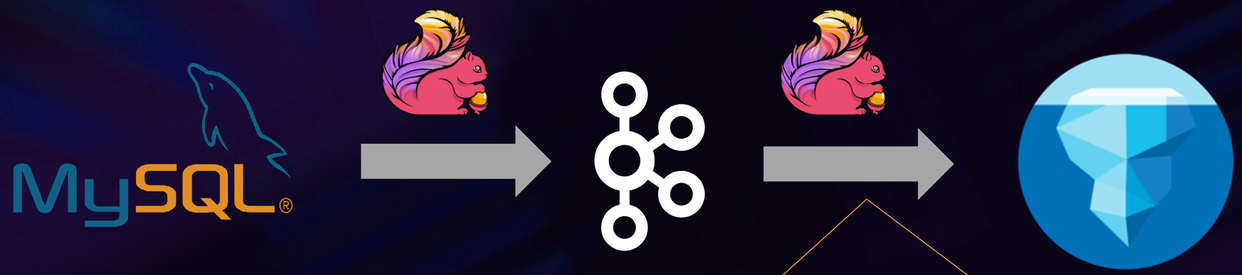


MySQL数据如何实时同步到OSS?





MySQL数据如何实时同步到OSS?



```sql
CREATE TABLE sbtest1(
    `id` INT NOT NULL,
    `k`  INT NOT NULL,
    `c`  CHAR(120) NOT NULL,
    `pad` CHAR(60) NOT NULL
) WITH (
    'connector' = 'mysql-cdc',
    'hostname' = 'localhost',
    'port' = '3306',
    'username' = '<mysql-user>',
    'password' = '<mysql-password>',
    'database-name' = 'test',
    'table-name' = 'sbtest1'
);
```

第一步：定义Source

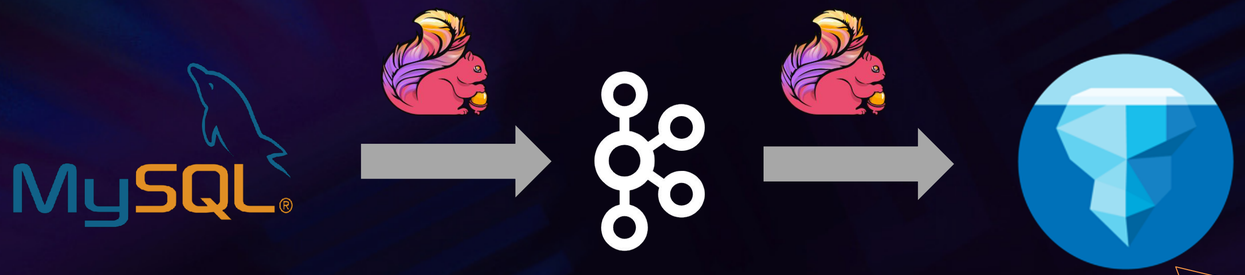
```sql
INSERT INTO iceberg_sbtest1 SELECT * FROM sbtest1;
```

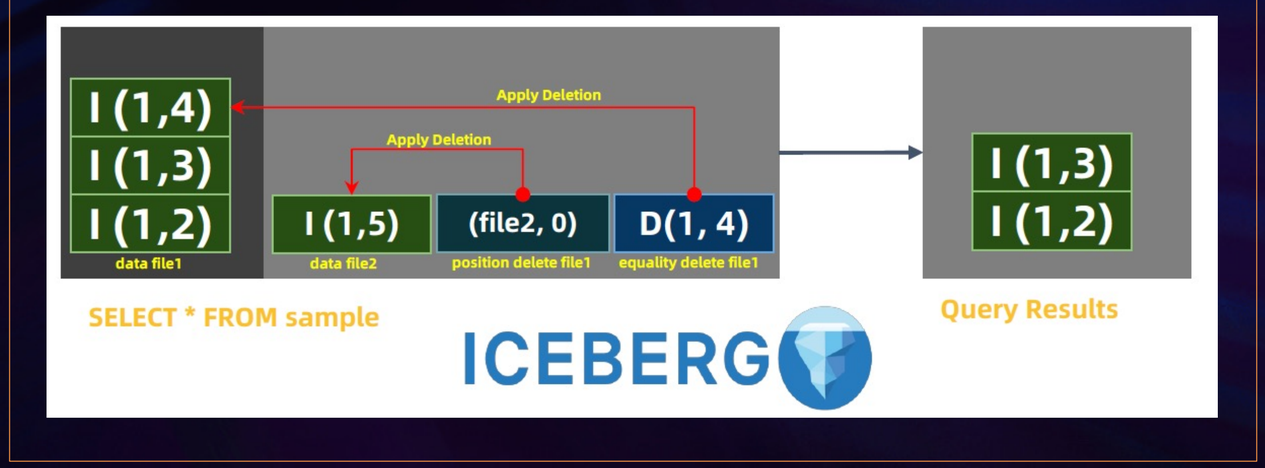
第三步：导入数据

```sql
CREATE TABLE iceberg_sbtest1(
    `id` INT NOT NULL,
    `k`  INT NOT NULL,
    `c`  CHAR(120) NOT NULL,
    `pad` CHAR(60) NOT NULL
) WITH (
    'connector' = 'iceberg',
    'catalog-name' = 'hive_prod',
    'uri' = 'thrift://localhost:9083',
    'engine.hive.enabled' = 'true',
    'location' = 'oss://iceberg/warehouse',
    'io-impl' = 'org.apache.iceberg.aliyun.oss.OSSFileIO',
    'access.key.id' = '******',
    'access.key.secret' = '******',
    'oss.endpoint' = 'oss-cn-hanghzou.aliyuncs.com'
);
```

第二步：定义Sink

MySQL数据如何实时同步到OSS?





FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

#4 现状及进展

Apache Iceberg 功能

Features

✅ Feature Available　? Feature in roadmap　⊗ No plan to support now

Category	Items	Sub-items	delta-io/delta	Apache Iceberg	Iceberg On Aliyun	Apache Hudi
Basic	ACID	-	✅	✅	✅	✅
	Time Travel	-	✅	✅	✅	✅
	Schema Evolution	-	✅	✅	✅	✅
	Source/Sink	Batch	✅	✅	✅	✅
		Streaming	✅	✅	✅	✅
Mutation	Schema Evolution	-	✅	✅	✅	✅
	Partition Evolution	-	✅	✅	✅	✅
	Copy-On-Write Update	-	✅	✅	✅	✅
	Merge-On-Read Update	Read	⊗	✅	✅	✅
		Write	⊗	✅	✅	✅
		Compaction	⊗	?	?	✅
Advanced Features	Z-Ordering	-	⊗	?	?	?
	E2E Encryption	-	⊗	✅	✅	⊗
	Secondary Indexes	-	⊗	?	?	⊗
	Local SSD Cache	-	⊗	⊗	✅	⊗
	Auto small files merge	-	⊗	⊗	✅	✅

Flink 集成 Iceberg 现状及规划

	Apache Flink	Apache Iceberg
Phase #1 (Connect to iceberg)	Apache Flink 1.11.0	Apache Iceberg 0.10.0 (Oct 2020) • Flink streaming sink • Flink batch sink • Flink batch source
Phase #2 (Replace hive table format)	Apache Flink 1.11.0	Apache Iceberg 0.11.0 (Jan 2021) • Flink source improvement - filter/limit push down • Flink streaming source • Format v2: CDC/Upsert (Phase#1) - write & read correctness data. • Major Compaction (Batch Mode).
Phase #3 (Batch/Stream row-level delete)	Apache Flink 1.12.0	Apache Iceberg 0.12.0 (~ Apr 2021) • Format v2: CDC/Upsert (Phase#2) - stability • Flink SQL imports CDC to iceberg.
	Apache Flink 1.13.2 Apache Flink 1.14.0	Apache iceberg 0.13.0 (?) • Support flink 1.13 • Support flink 1.14 • Format v2: CDC/Upsert
Phase #4	More flink versions	Apache iceberg 0.14.0 (?) • Flip-27 reader/writer • Format v2: CDC/Upsert - Improvements. • Delete Files compaction.


FLINK FORWARD
#ASIA 2021
ONLINE
实时即未来
REAL-TIME IS THE FUTURE
THANKS