

# 基于ClickHouse的DataOps实践

北京万山数据科技有限公司

鲁四海

2024-07-20

# ：创始人

## 高级工程师、研究生导师

- 北大CIIIM电子商务大数据开放实验室副主任
- 成都信息工程大学产业互联网研究院大数据分析实验主任
- CIO时代可信数据空间专委会秘书长
- 西南科技大学研究生导师
- ClickHouse技术爱好者



系统架构设计师、TOGAF鉴定级企业架构师，国内首批大数据社区核心发起成员，积极推动ClickHouse在传统行业落地。支持了多家企业数据中台建设；主持参与了多个国家、省、市级大数据项目。出版了《中国智慧城市实践指南》、《云计算技术与标准化》、《赋能数字经济：大数据创新创业启示录》、《中国互联网发展报告》、《中国白酒行业电子商务大数据分析报告》、《中国大数据企业排行榜》、《中国大数据产业地图》等书籍和报告。曾获得全国信息技术标准化技术委员会优秀个人。

# 目录

什么是DataOps

为什么是Clickhouse

如何实现基于Clickhouse的湖仓一体

如何实现“去ETL”

如何实现数据治理可视化

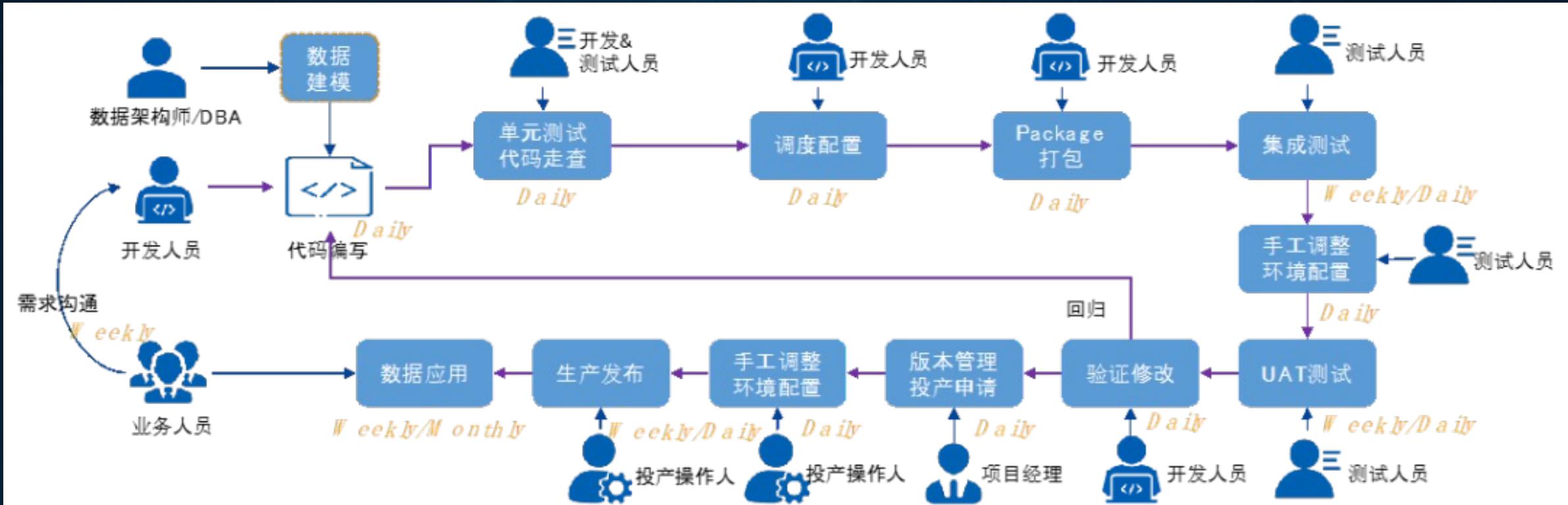
# DataOps基本概念

DataOps是“数据操作”的缩写，DataOps是一种面向流程的自动化方法，由分析和数据团队使用，旨在提高数据分析的质量并缩短数据分析的周期。DataOps的这一定义会随着时间的推移而变化，但其关键目标非常明确：提高数据分析的质量并缩短数据分析的周期。在2018年Gartner发布的。

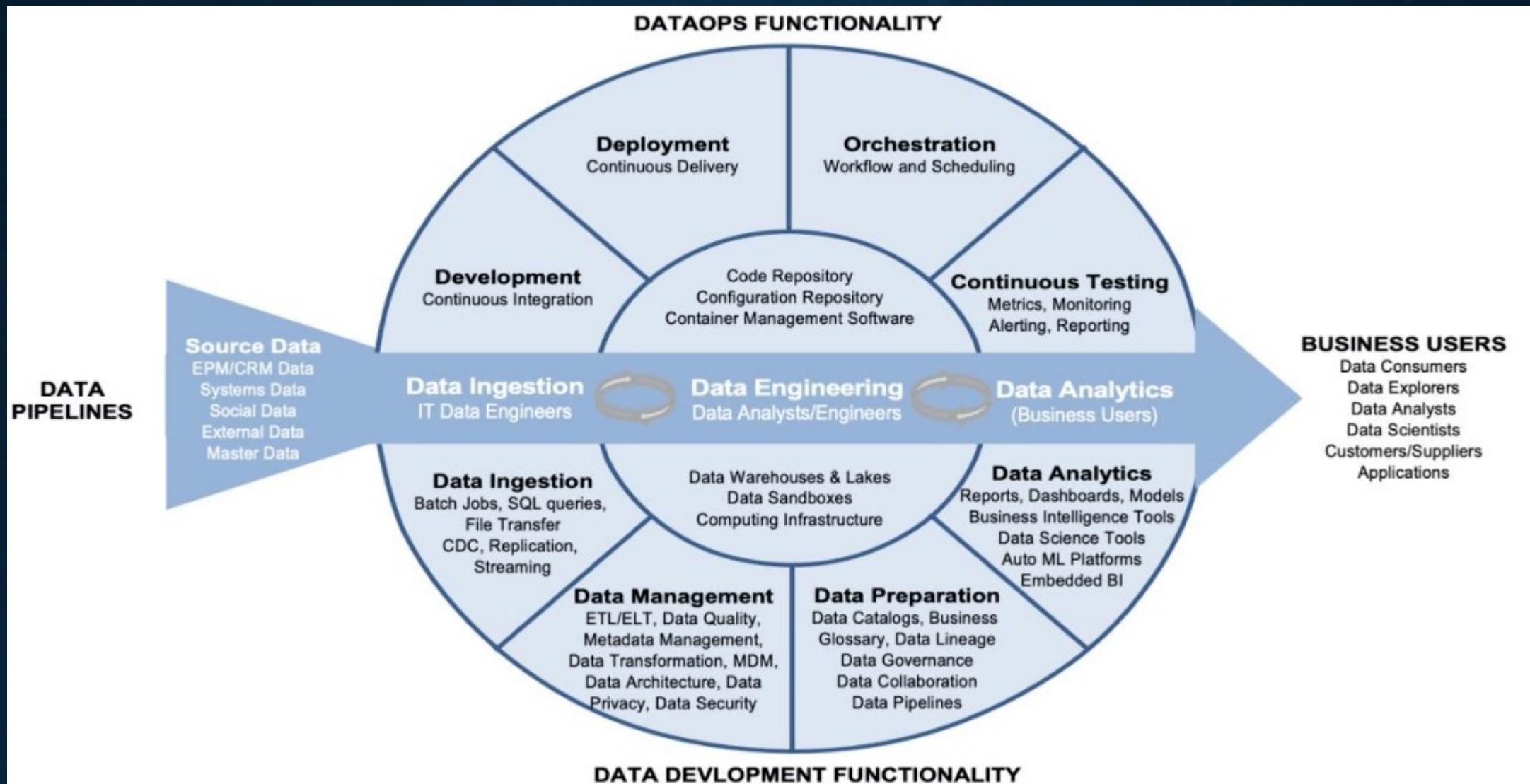
《数据管理技术成熟度曲线》报告中，DataOps的概念被首次提出。DataOps平台定位于“大数据视窗操作系统”，为大数据、数据仓库的开发、运维提供更直接、简单的操作界面。通过可视化的操作界面，利用湖仓一体平台的计算能力，实现简单化可控的数据开发，保障从数据集成到数据应用全过程中，每一步是可见的，数据是可溯源的。

信通院：数据研发运营一体化（DataOps）：是数据开发的新范式，将敏捷、精益等理念融入数据开发过程，通过对数据相关人员、工具和流程的重新组织，打破协作壁垒，构建集开发、治理、运营于一体的自动化数据流水线，不断提高数据产品交付效率与质量，实现高质量数字化发展。

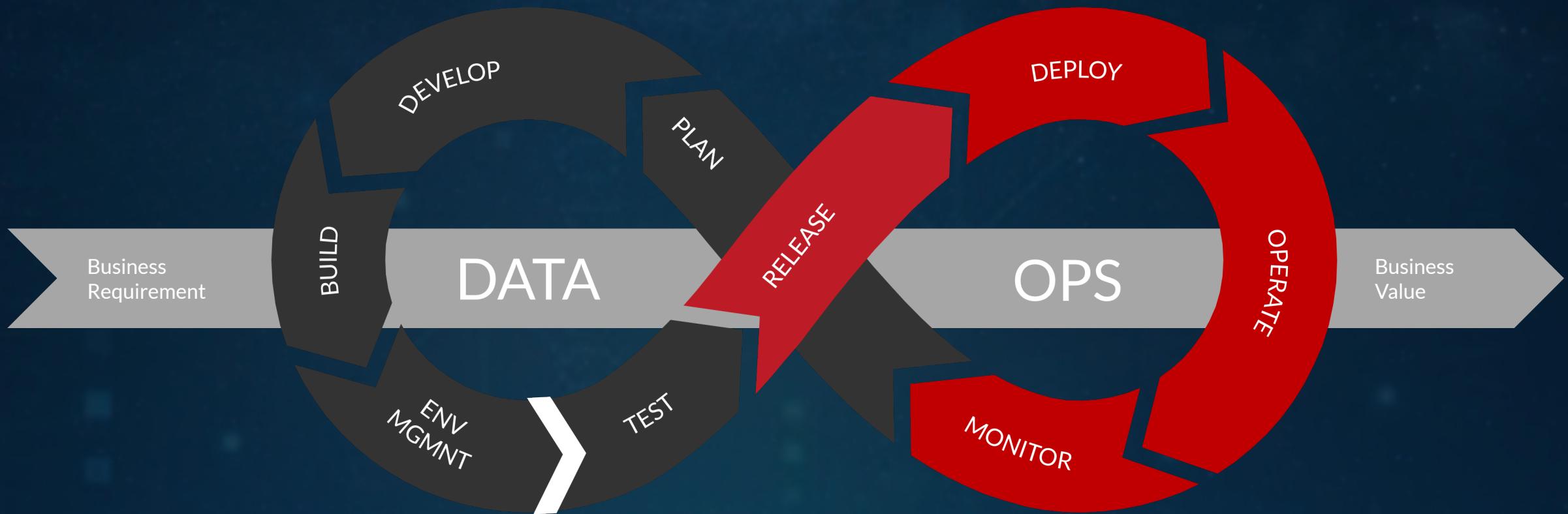
# 传统数据开发模式



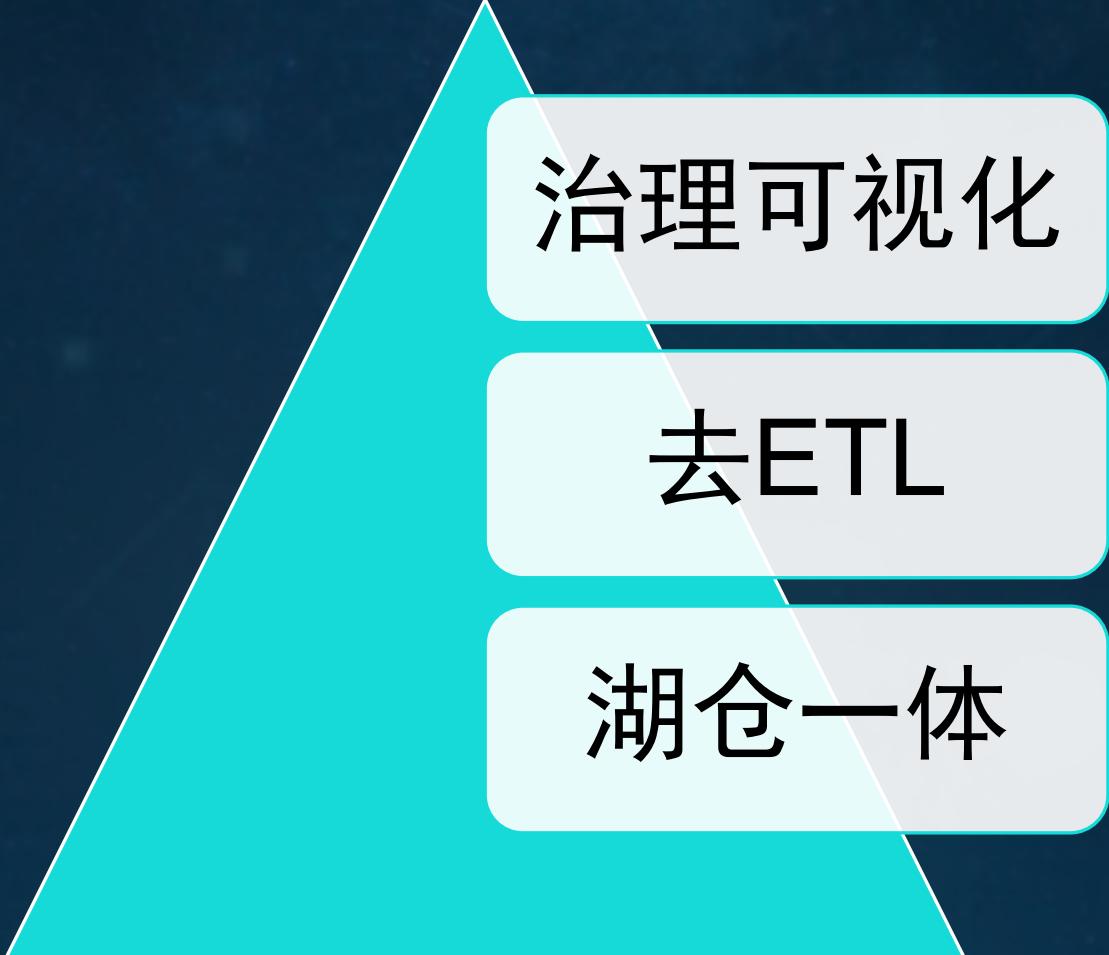
# DataOps数据管道



# DataOps双轮模型



# ■ DataOp技术平台应该有的三个特点



治理可视化

去ETL

湖仓一体

# 目录

什么是DataOps

为什么是Clickhouse

如何实现基于Clickhouse的湖仓一体

如何实现“去ETL”

如何实现数据治理可视化

# : OLAP发展



关系型数据库



MapReduce



SQL on Hadoop



Impala



MPP

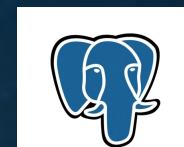


# : Clickhouse集成数据真的很简单

## MaterializedView



ClickHouse



File

URL

# : Clickhouse处理数据真的很简单

常规函数	Arithmetic, Arrays, arrayJoin, UDF, Bit, Bitmap, Comparison, Conditional, Dates and Times, Dictionaries, Distance, Embedded Dictionaries, Geo, Encoding, Encryption, Files, Hash, IN Operator, IP Addresses, Introspection, JSON, Logical, Machine Learning, Maps, Mathematical, NLP, Nullable, Other, Random Numbers, Replacing in Strings, Splitting Strings, Strings, Time Series, Time Window, Tuples, Type Conversion, UUID, URLs, UUIDs, uniqTheta
聚合函数	List of Aggregate Functions, Combinators, Parametric, GROUPING
表函数	azureBlobStorage, cluster, deltaLake, dictionary, executable, azureBlobStorageCluster, file, fileCluster, format, gcs, fuzzJSON, fuzzQuery, generateRandom, mergeTreeIndex, hdfs, hdfsCluster, hudi, iceberg, input, jdbd, merge, mongodb, mysql, nullfunction, numbers, generate_series, generateSeries, odbc, postgresql, redis, remote, s3, s3Cluster, sqlite, url, urlCluster, view, loop

# 我看到的Clickhouse



# 目录

什么是DataOps

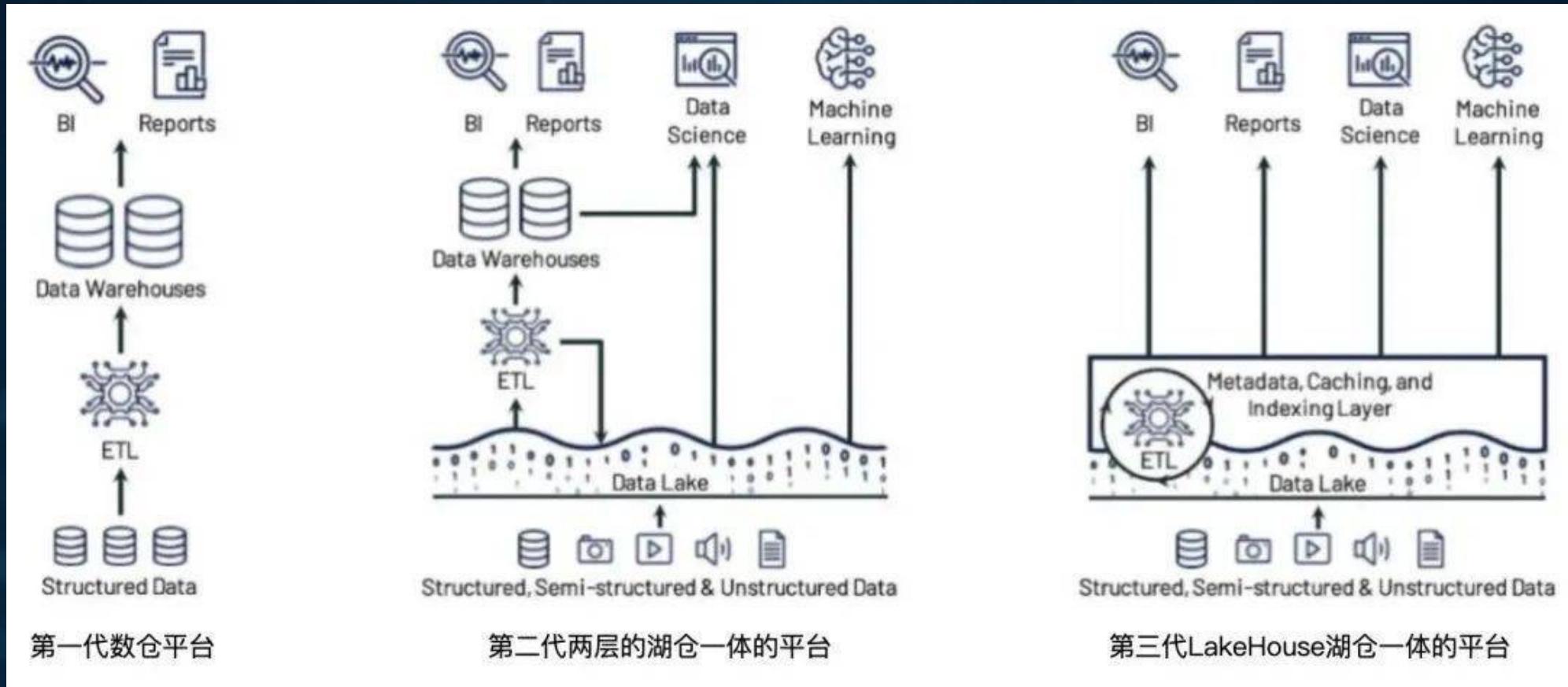
为什么是Clickhouse

如何实现基于Clickhouse的湖仓一体

如何实现“去ETL”

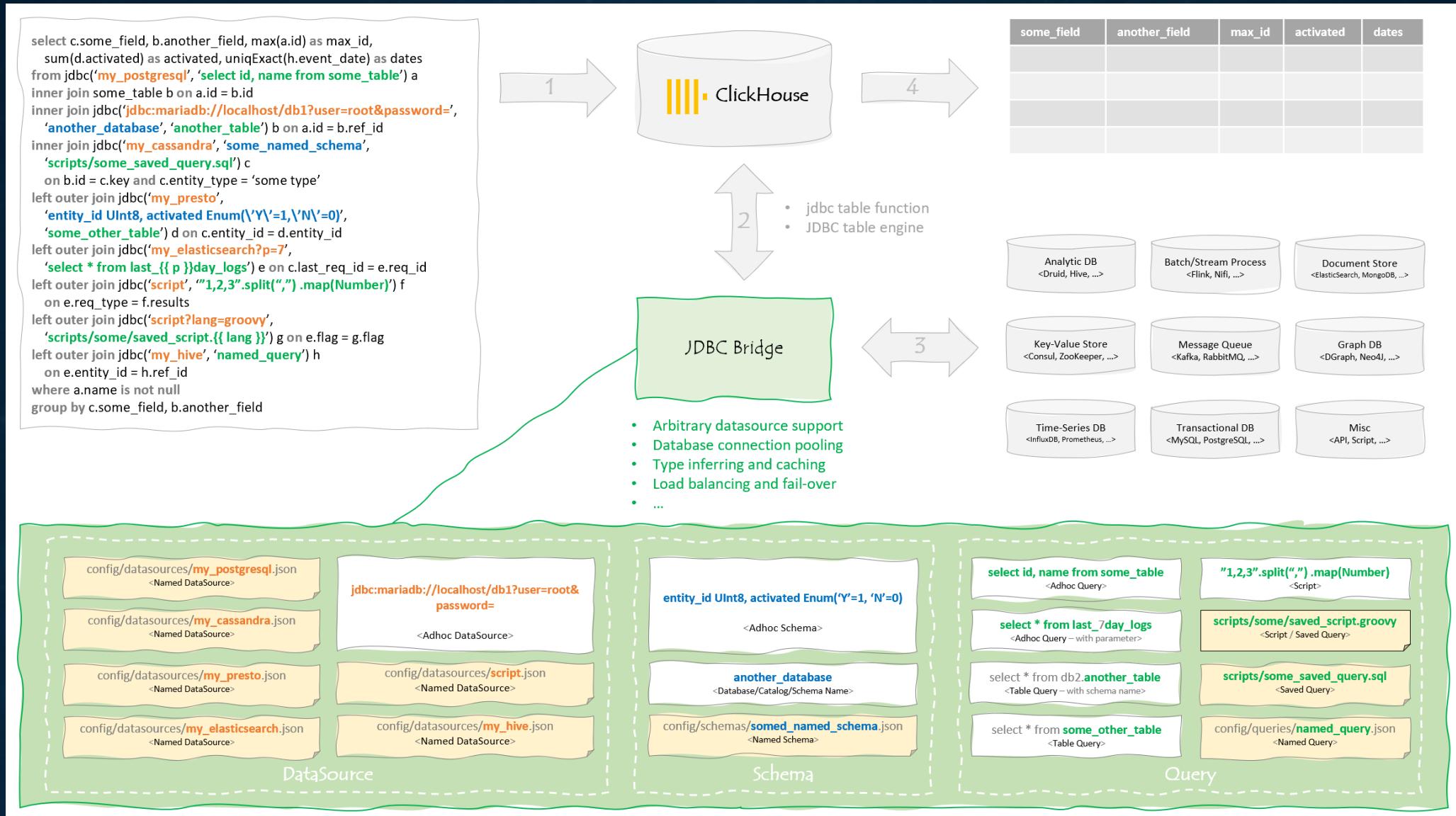
如何实现数据治理可视化

# 湖仓一体



操作入口？数据入口？湖仓一体=湖+仓？

# Clickhouse JDBC-Bridge



# Clickhouse DBH

基本信息    连接属性    可写表    隐私计算    测试

编辑

编辑

基本信息    开发环境    测试环境    生产环境

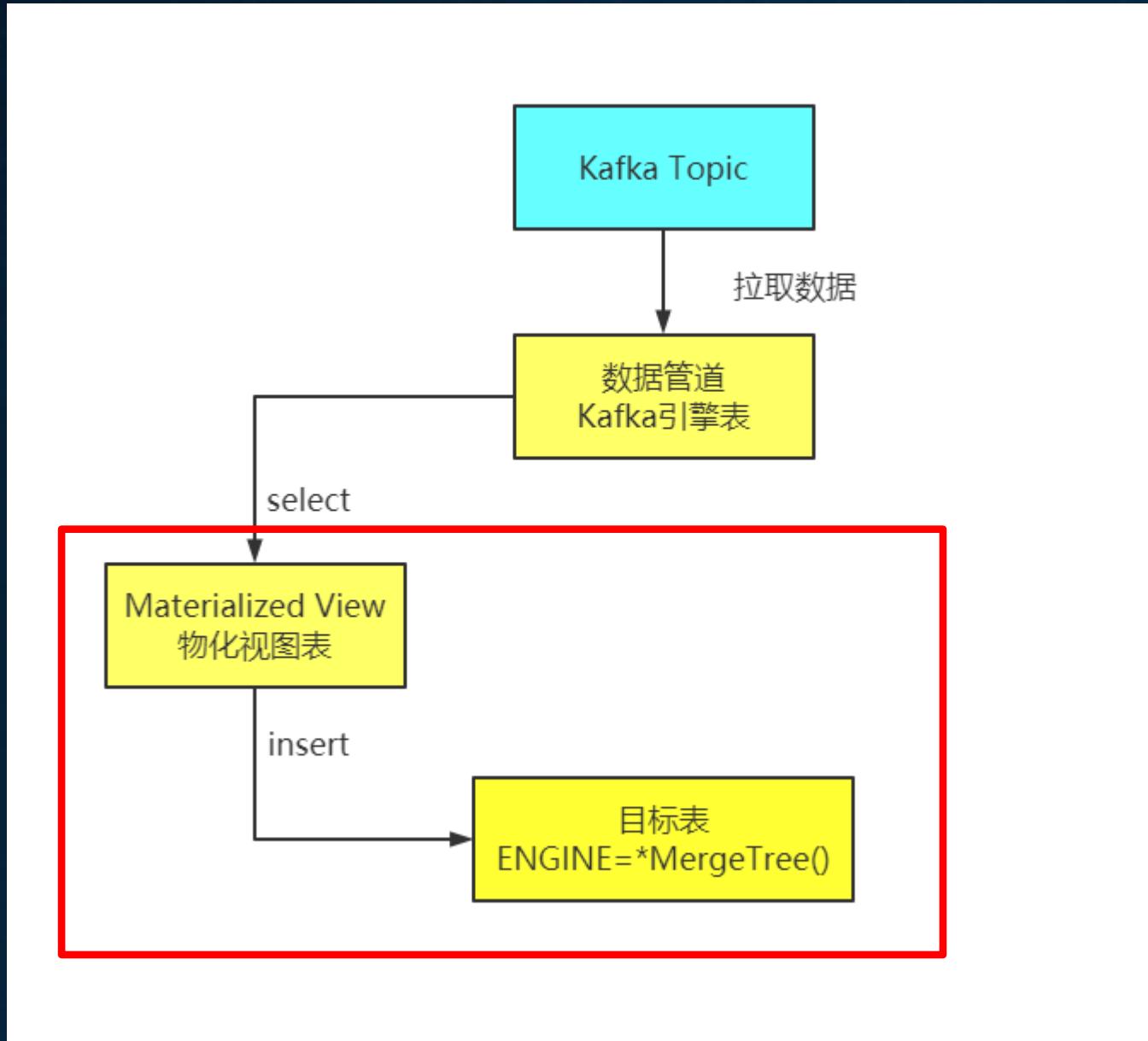
\* 标签

备注

返回    提交

- ✓ 数据类型转换优化
- ✓ 数据写入优化
- ✓ Clickhouse数据源读写优化
- ✓ 白名单控制
- ✓ 读写控制
- ✓ 脱敏配置
- ✓ 多个节点一个管理端，图形化配置管理
- ✓ 多种环境切换

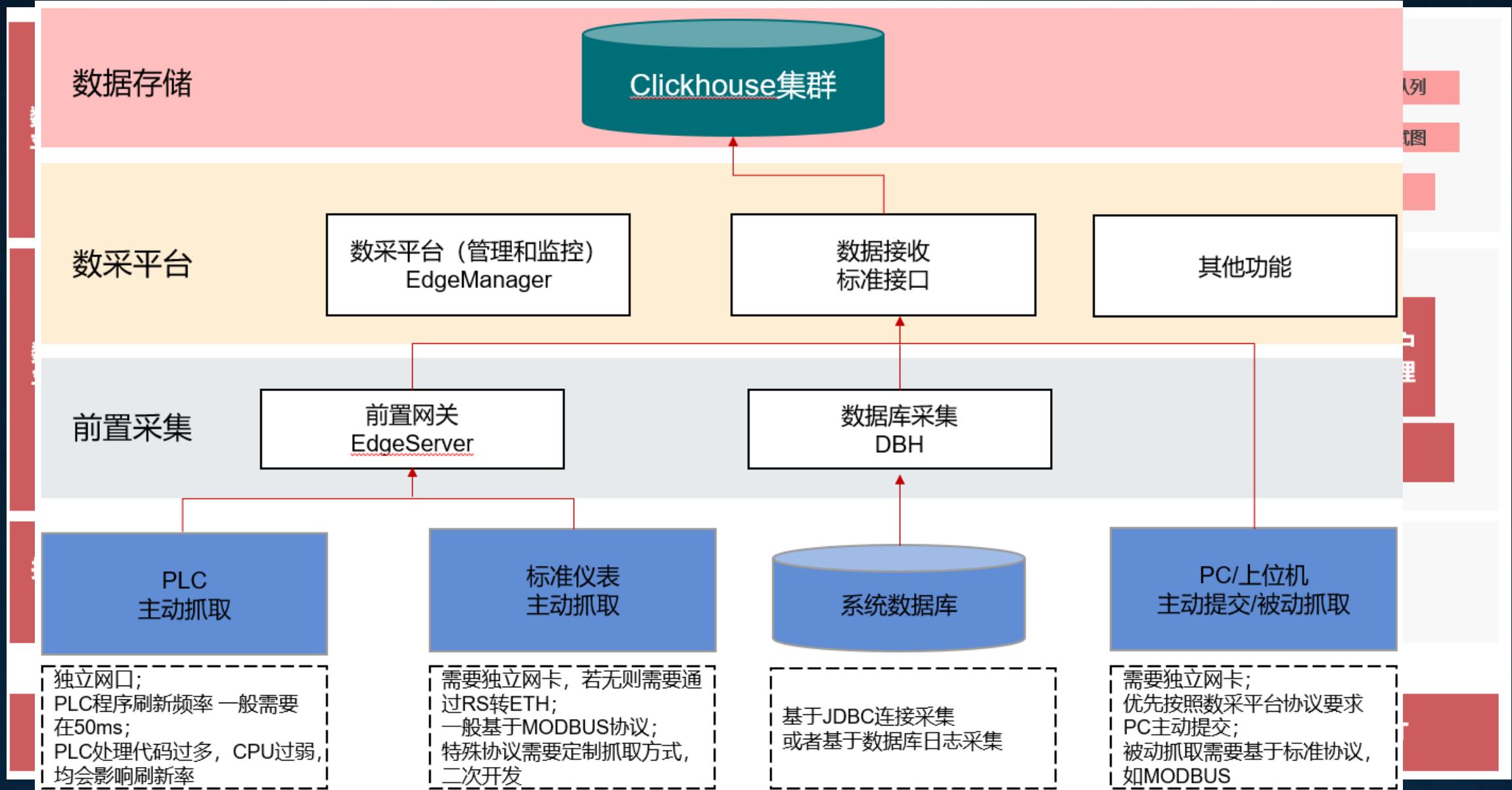
# Kafka+MV



常见问题：

- ✓ 格式出错丢数
- ✓ 聚合统计
- ✓ 数据分片
- ✓ 多副本
- ✓ Inner表

# IOT融合应用案例



# 目录

什么是DataOps

为什么是Clickhouse

如何实现基于Clickhouse的湖仓一体

如何实现“去ETL”

如何实现数据治理可视化

# ETL相关说法



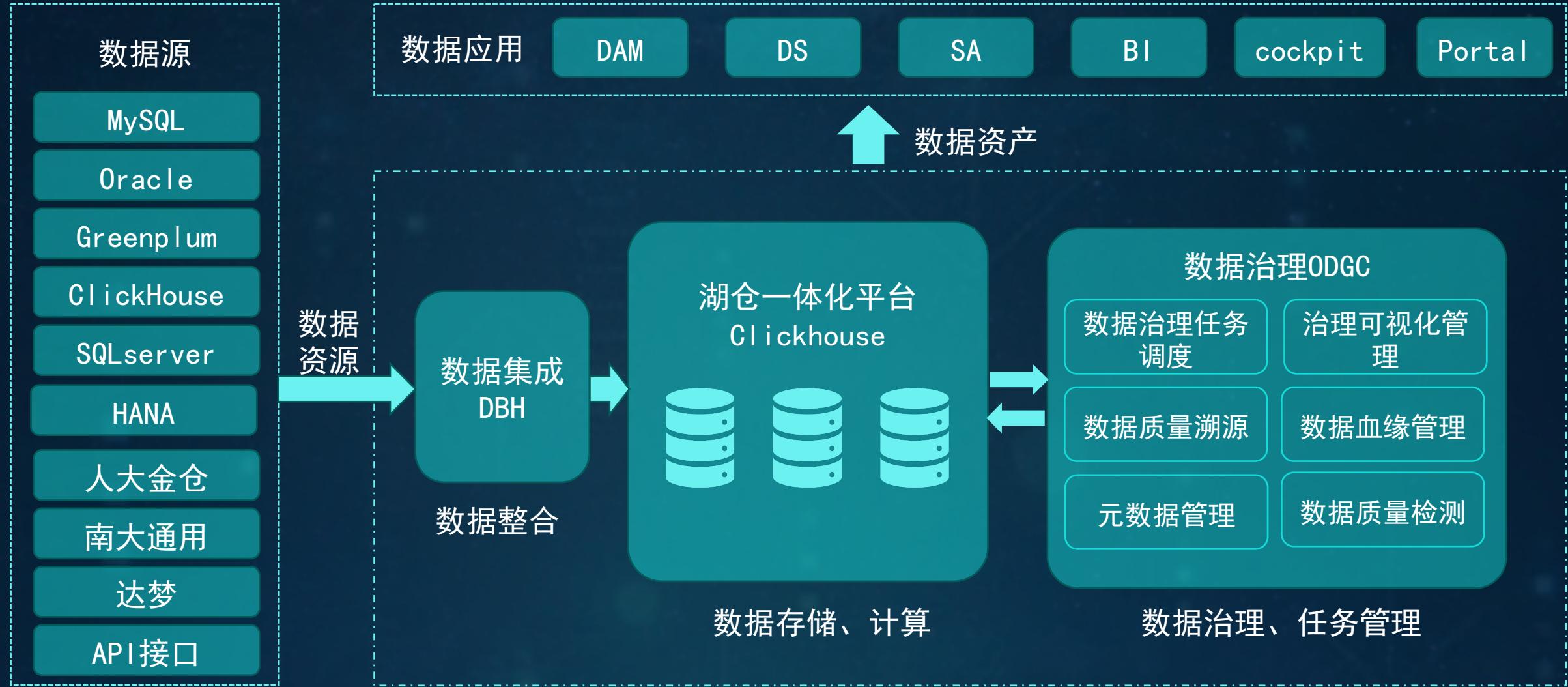
ETL与ELT



ETL的处理链路参考图

# “去ETL”

“去ETL” 去的不是ETL，而是把独立的ETL工具和ETL开发任务去掉！



# ■ 预计算

```
ALTER TABLE uk_price_paid  
  ADD PROJECTION  
town_sort_projection (  
    SELECT  
      town, price,  
date, street, locality  
    ORDER BY town  
)
```



```
ALTER TABLE uk_price_paid  
MATERIALIZE PROJECTION  
town_sort_projection
```



```
avg(price)  
140764.81274599963
```

1 row in set. Elapsed: 0.004 sec. Processed 65.54 thousand rows, 327.74 KB  
(15.29 million rows/s., 76.48 MB/s.)



```
SELECT avg(price)  
FROM uk_price_paid  
WHERE  
  town =  
  'DURHAM'
```

# 特殊字段

```
CREATE OR REPLACE TABLE test
(
    (
        id UInt64,
        size_bytes Int64,
        size String Alias formatReadableSize(size_bytes)
    )
)
ENGINE = MergeTree
ORDER BY id;

INSERT INTO test Values (1, 4678899);

SELECT id, size_bytes, size FROM test;
+-----+-----+-----+
| id   | size_bytes | size      |
+-----+-----+-----+
| 1    | 4678899   | 4.46 MiB |
+-----+-----+-----+

SELECT * FROM test SETTINGS asterisk_include_alias_columns=1;
+-----+-----+-----+
| id   | size_bytes | size      |
+-----+-----+-----+
| 1    | 4678899   | 4.46 MiB |
+-----+-----+-----+
```

示例：

```
alter table shard_detail_xxx add
column phy_deleted UInt8 alias
if(pk in(select pk_deleted from
t3),1,0)
```

注意事项？

# Dictionary换JOIN

(1) 建表

```
CREATE TABLE gcm.Bonus
(
    `BonusID` Int32,
    `Description` String
)
ENGINE = MergeTree ORDER BY BonusID
```

(2) 创建字典

```
CREATE DICTIONARY gcm.zdict_Bonus
(
    `BonusID` UInt64,
    `Description` String
) PRIMARY KEY BonusID
SOURCE(CLICKHOUSE(HOST 'localhost' PORT tcpPort() USER 'default' PASSWORD 'yyyang'
DATABASE 'gcm' TABLE 'Bonus'))
LIFETIME(MIN 10 MAX 20)
LAYOUT(HASHED())
```

(3) 字典使用

```
select dictGet('gcm.zdict_Bonus','Description',toUInt64(111)) as Description
```

# 复杂文本提取与比对

```
VPN.txt new 1 new 3 new 4 zfcg0804.sql
1 JDBC=jdbc:clickhousenative://localhost:9000
2
3 SQL1=DROP TABLE IF EXISTS shard_proc.zfcg0804sql_1
4 SQL2=CREATE TABLE shard_proc.zfcg0804sql_1 ENGINE=MergeTree ORDER BY ID AS SELECT ID,extract(replaceRegexpAll(
replaceRegexpAll(WIN_BID_BULLETIN_CONTENT,'</[Tt][Dd]>|</[Pp]>|</?[Bb][Rr]>', ''), '<[^>]+>|&nbsp;|\s+', ''),
'中标[人单位]{1,2}[名称]{0,2}[: :]?^(0,4)([\u4e00-\u9fa5]\w{1,30})司院心局处社厂所队学校站会行场团部
台]') as BIDDER from shard_out.zhengFuCaiGou where SOURCE_TYPE='1'
5
6 SQL3=DROP TABLE IF EXISTS shard_proc.zfcg0804sql
7 SQL4=CREATE TABLE shard_proc.zfcg0804sql ENGINE=MergeTree ORDER BY ID AS SELECT ID,BIDDER,uniscid AS BIDDER_CODE
from shard_proc.zfcg0804sql_1 AS t1 INNER JOIN (select * from shard_inn.in_faren_total where fname in(select
BIDDER from shard_proc.zfcg0804sql_1))AS t2 ON t1.BIDDER=t2.fname WHERE isNotNull(BIDDER) and not empty(BIDDER)
try {
    Pattern
    p=Pattern.compile("中标[人单位]{1,2}[名称]{0,2}[: :]?^(0,4)([\u4e00-\u9fa5]\w{1,30})司院心局处社厂所队学校站会行场团部
台]");
    String cConnString="jdbc:clickhouse://localhost:8123/default?user=default&password=";
    Connection cConn.DriverManager.getConnection(cConnString);
    Statement cStmt = cConn.createStatement();
    Connection cConn2.DriverManager.getConnection(cConnString);
    Statement cStmt2 = cConn.createStatement();

    cStmt2.executeUpdate("DROP TABLE IF EXISTS shard_proc.zfcg0804");
    cStmt2.executeUpdate("CREATE TABLE shard_proc.zfcg0804(ID String,BIDDER Nullable(String)
Nullable(String))ENGINE=MergeTree ORDER BY ID");
    ResultSet rs=cStmt.executeQuery("select ID,WIN_BID_BULLETIN_CONTENT from shard_out.zhengF
1000");
    while(rs.next()) {
        String id=rs.getString("ID");
        String conn=rs.getString("WIN_BID_BULLETIN_CONTENT");
        conn=conn.replaceAll("</[Tt][Dd]>|</[Pp]>|</?[Bb][Rr]>", "").replaceAll("<[^>]+>|&nbsp;|\s+", "");
        Matcher m=p.matcher(conn);
        if(m.find()) {
            String bidder=m.group(1);
            String bidder_code="";
            ResultSet rsl=cStmt2.executeQuery("select fname,uniscid from shard_inn.in_faren");
            if(rsl.next()) {
                bidder_code=rsl.getString("uniscid");
            }
            cStmt2.executeUpdate("insert into
shard_proc.zfcg0804(ID,BIDDER,BIDDER_CODE)VALUES('"+id+"','"+bidder+"','"+bidder_
")
        }
    }
}catch(Exception e) {
    System.out.println(e);
}
```

比较项	SQL处理	JAVA编程处理
每秒执行条数	超过2万条	不到20条
代码量	6行	29行
编程语言	SQL	SQL、JAVA
是否需要编译	不需要	需要
无图形化界面调试难度	低	高
是否需要IDE支持	不需要	需要

## 复杂xml文件解析

```
SELECT arr['ticketinfo//@wagernumber '] ticketinfo_wagernumber,  
arr['ticketinfo//@betttime '] ticketinfo_betttime,  
...  
From (arrayJoin(select  
wsdXpathMapArrFromFile('/wsdck/clickhouse/POC_test_data/sbc/2024-06-  
13_page0002.xml'  
, '/betinfo'  
, ['ticketinfo//@wagernumber','ticketinfo//@betdate','ticketinfo//@betttime','ticketinfo//@sour  
cetype'...])) as arr)
```

# 我们一直致力于：让Clickhouse更好用，一切皆SQL

- ◆ wsdGet, 以GET方法操作远程HTTP接口，支持权限验证、数据加解密；
- ◆ wsdPost, 以POST方法操作远程HTTP接口，支持权限验证、数据加解密，自定义JSON格式请求体；
- ◆ wsdSM3, 提取SM3的摘要字符串；
- ◆ wsdSM4Decode, 用于提取SM4解密字符串；
- ◆ wsdSM4Encode, 用于生成SM4加密字符串；
- ◆ wsdlIndex, 全文检索函数，插入数据时建立单条索引后，按照指定的索引类型返回UUID；
- ◆ wsdlIndexDaemon, 全文检索函数，后台执行，用于对全表或某个表中新增数据进行索引，返回执行情况字符串；
- ◆ wsdSearch, 全文检索函数，按照指定的搜索类型返回UUID数组；
- ◆ 自然语言处理函数，如分词函数及字符串处理函数包括wsdCutMost、wsdCutMostPinYin、wsdCutNlp、wsdCutNlpPinYin、wsdEnZhCut、wsdHtml2Text等；如简繁体及汉字拼音转换函数包括wsdFan2Jian、wsdJian2Fan、wsdPinYin2HanZi、wsdHanZi2PinYin、wsdHanZi2PinYinNon、wsdHanZi2PinYinNumeric

# 目录

什么是DataOps

为什么是Clickhouse

如何实现基于Clickhouse的湖仓一体

如何实现“去ETL”

如何实现数据治理可视化

# 完全SQL化，开发即在线：ODGC

The screenshot shows a software interface for configuring ODGC (Online Data Generation Configuration). On the left, there's a vertical sidebar with buttons for '新建' (New), '基本' (Basic), '参数' (Parameters), '上级' (Advanced), and '保存' (Save). The main area has tabs at the top: '基本设置' (Basic Settings), '参数设置' (Parameter Settings), '数据输入' (Data Input), '数据输出' (Data Output), and '辅助SQL' (Auxiliary SQL). The '辅助SQL' tab is active, showing sections for '前置SQL' (Pre-SQL) and '后置SQL' (Post-SQL). Each section contains a table with columns for '前置SQL' (Pre-SQL) or '后置SQL' (Post-SQL) and '操作' (Operation). There are two rows in each table, each with a '删除' (Delete) button. Below the tables are buttons for '添加前置SQL' (Add Pre-SQL) and '添加后置SQL' (Add Post-SQL). At the bottom, there are '保存' (Save) buttons.

新建

基本

参数

上级

保存

基本设置

参数设置

数据输入

数据输出

辅助SQL

前置SQL:

前置SQL	操作
	删除
	删除

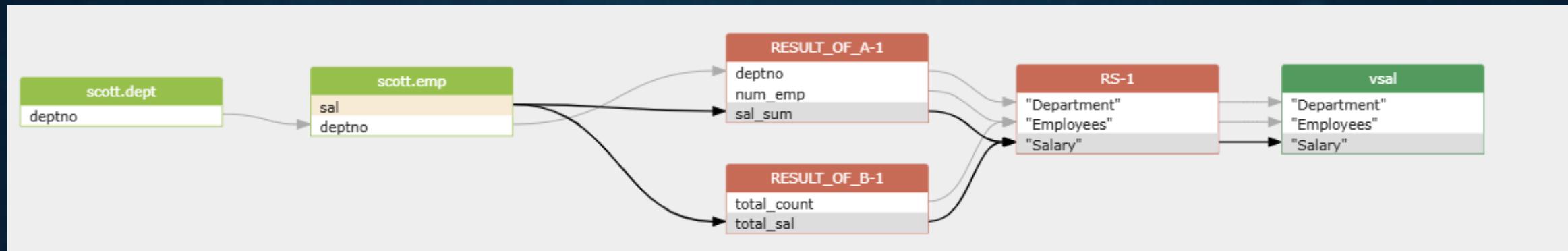
添加前置SQL

后置SQL	操作
	删除
	删除

添加后置SQL

保存

## 实时的数据血缘



副页

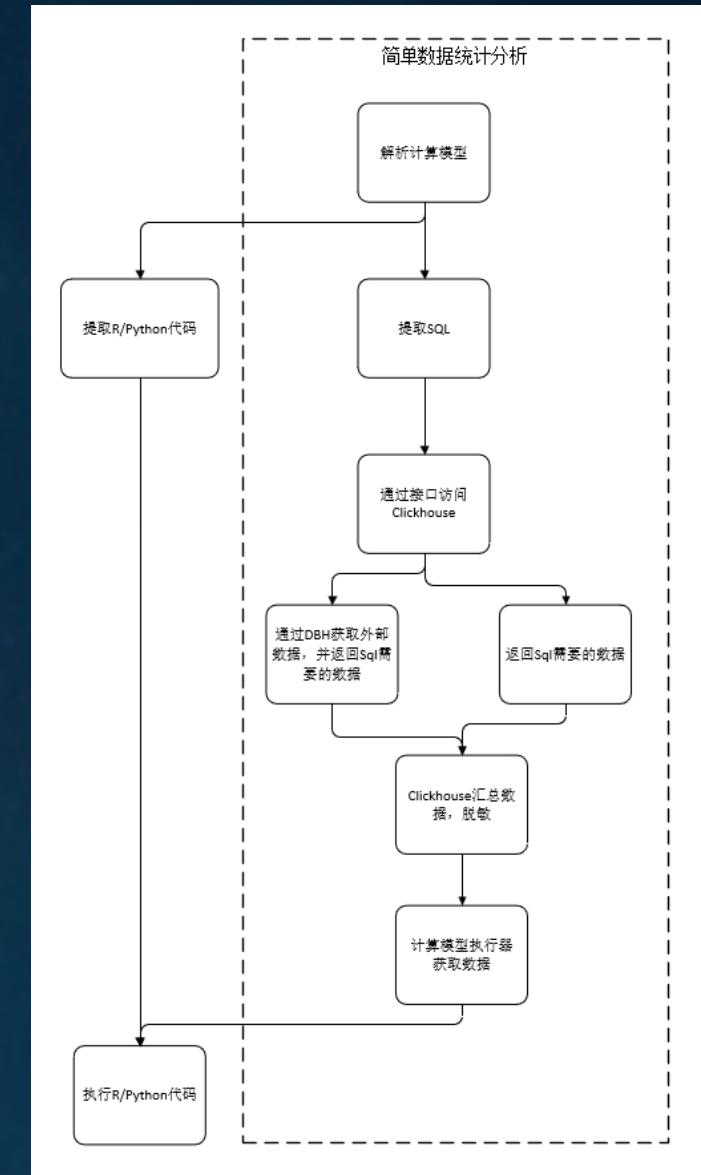
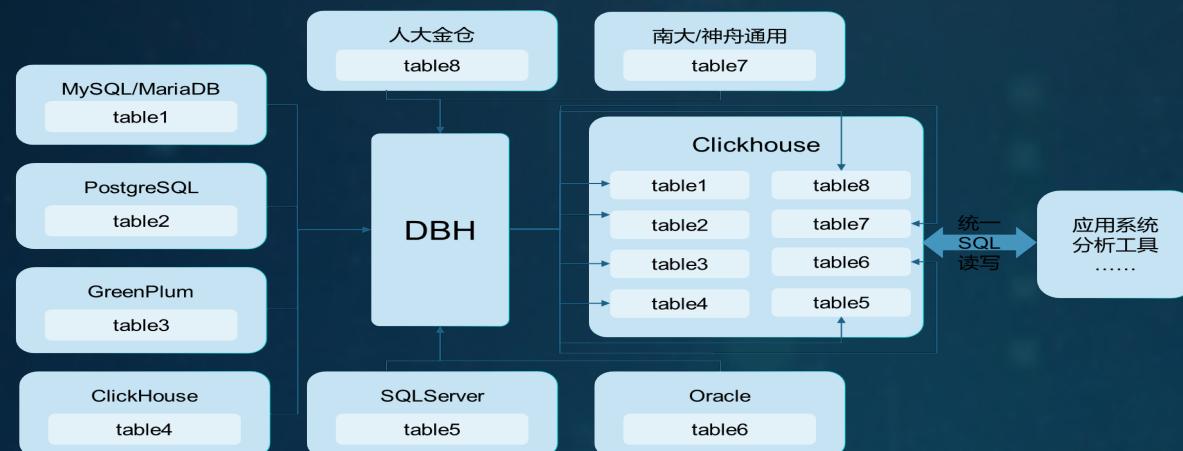
# Clickhouse在可信数据空间：异构数据统一计算

通过DBH(Data Base Hub)消除异数据的差异，所有数据源均可从Clickhouse进行访问。

利用R/Python构建统一的数据挖掘运行环境。

简单的数据计算模型仅包含SQL，直接通过Clickhouse返回结果。

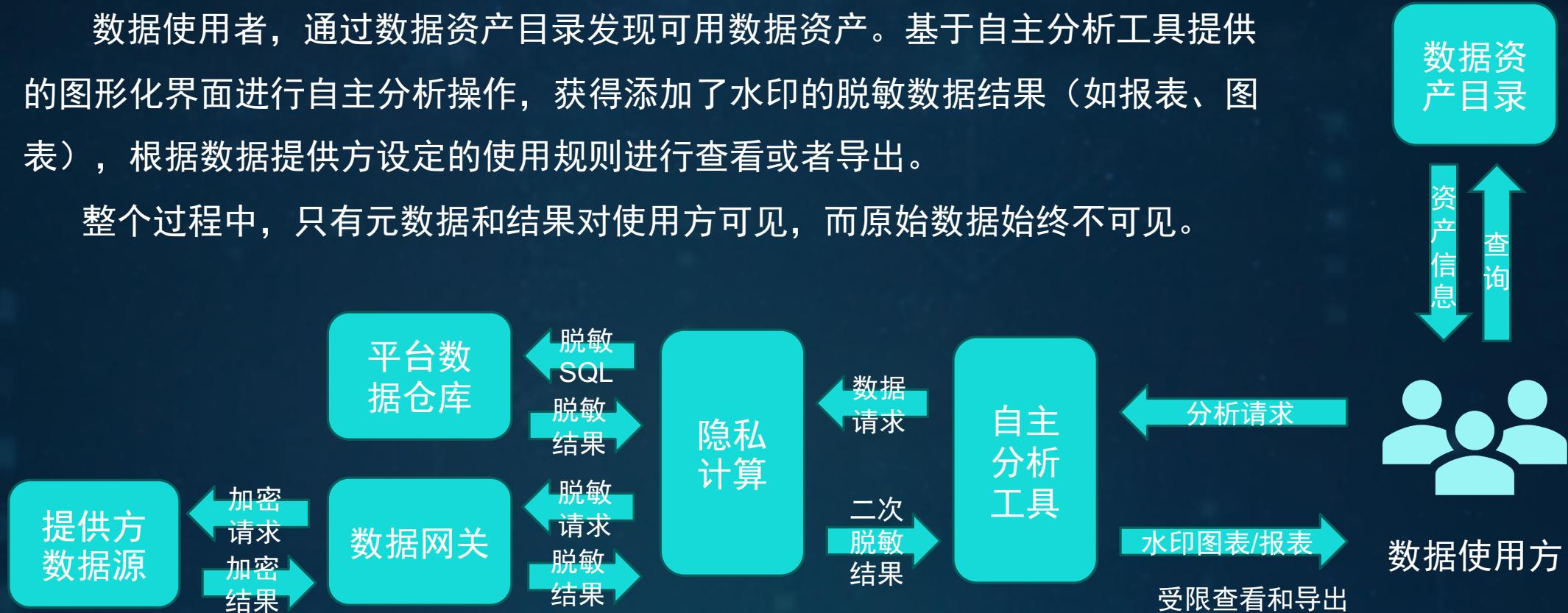
复杂的数据计算模型由SQL和R/Python代码组成，通过SQL获取数据，通过运行R/Python代码得到最终结果。



# Clickhouse在可信数据空间：数据可用不可见

数据使用者，通过数据资产目录发现可用数据资产。基于自主分析工具提供的图形化界面进行自主分析操作，获得添加了水印的脱敏数据结果（如报表、图表），根据数据提供方设定的使用规则进行查看或者导出。

整个过程中，只有元数据和结果对使用方可见，而原始数据始终不可见。

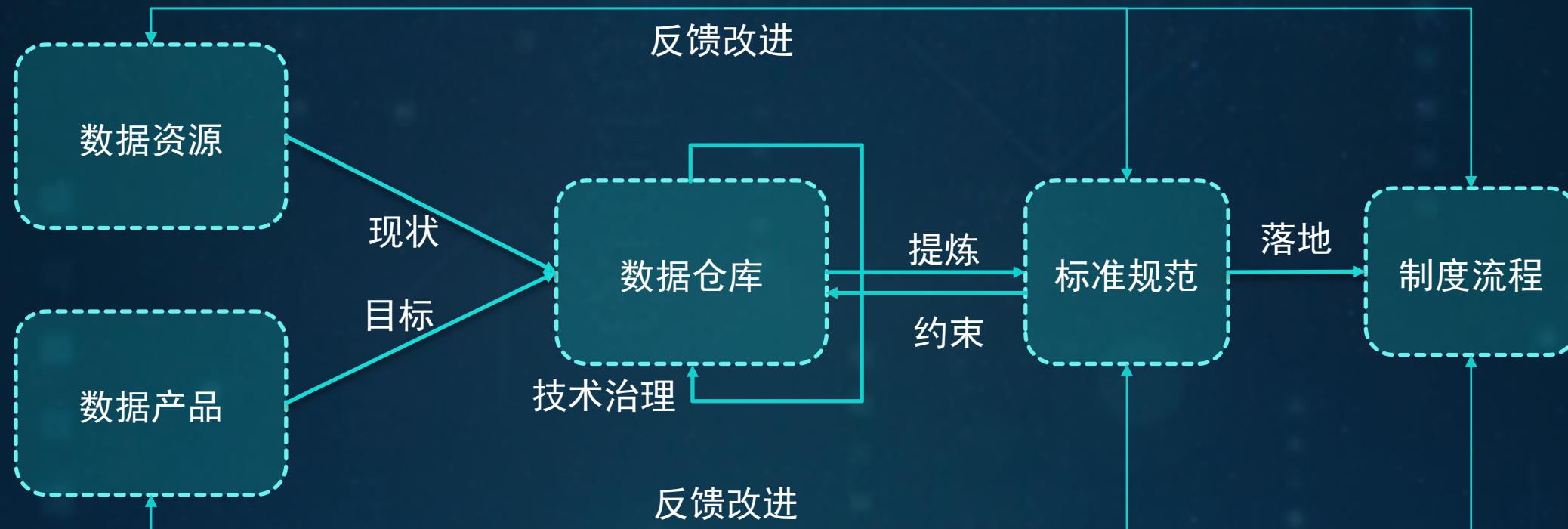


# 建议



## 建议

# 反向治理，以终为始





THANKS