# 背景

 Lambda架构的核心理念是“流批一体化”，因为随着机器性能和数据框架的不断完善，用户其实不关心底层是如何运行的，批处理也好，流式处理也罢，能按照统一的模型返回结果就可以了，这就是Lambda架构诞生的原因。现在很多应用，例如Spark和Flink，都支持这种结构，也就是数据进入平台后，可以选择批处理运行，也可以选择流式处理运行，但不管怎样，一致性都是相同的。

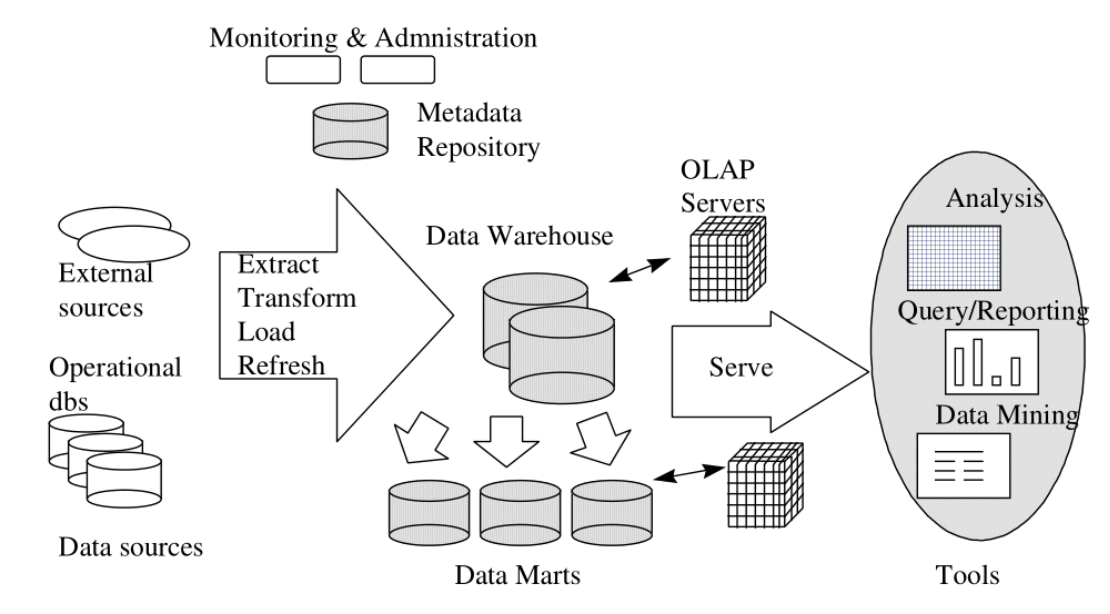
# 概述

联机实时分析OLAP（On-Line Analytical Processing），OLAP是面向数据分析的，也称为面向信息分析处理过程。它使分析人员能够迅速、一致、交互地从各个方面观察信息，以达到深入理解数据的目的。其特征是应对海量数据，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。例如数据仓库是其典型的OLAP系统。

## 数据仓库

这是商业智能（BI）的核心部分，主要是将不同数据源的数据整合在一起，通过多维分析为企业提供决策支持、报表生成等。存入数据仓库的资料必定包含时间属性。

数据仓库和OLAP关系：

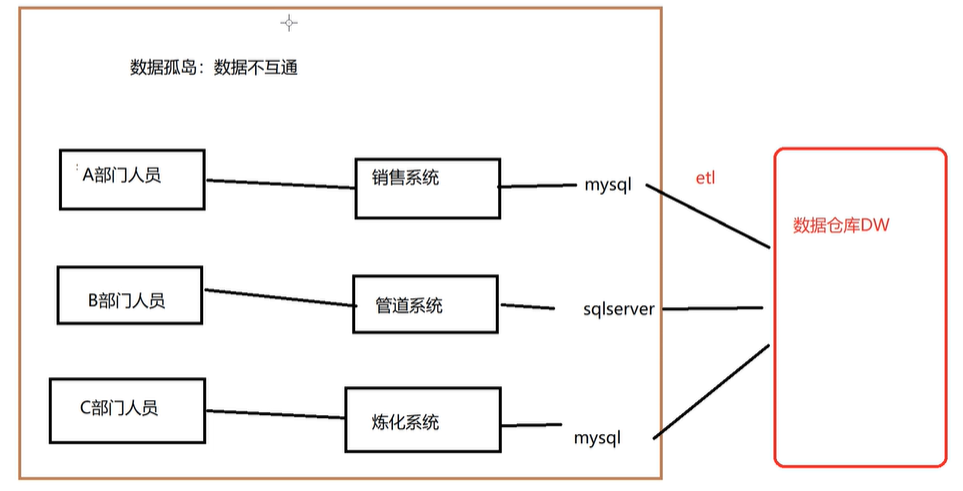


数据仓库和数据库的主要区别：用途不同。

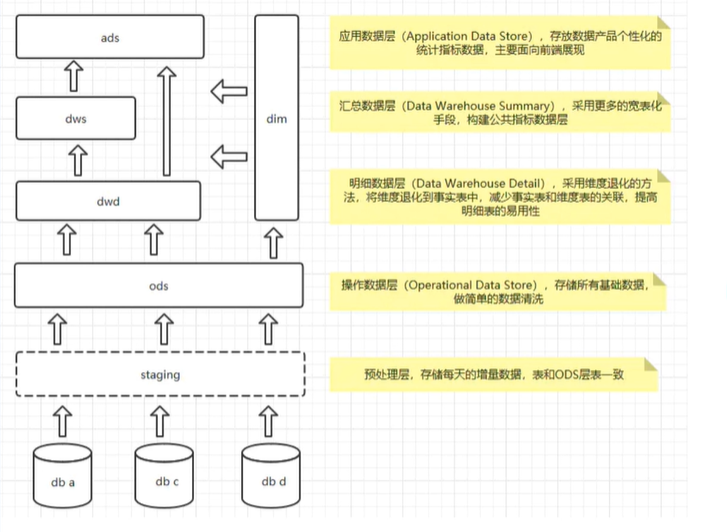
|  |  |
| --- | --- |
| 数据库 | 数据仓库 |
| 面向事务  存储在线的业务数据，对上层业务改变作出实时反映，遵循三范式设计 | 面向分析  历史数据，主要为企业决策提供支持，数据可能存在大量冗余，但是利于多个维度分析，为决策者提供更多观察视角 |

一般来说，在传统BI领域里，数据仓库的数据同样是存储在MySQL这样的数据库中的。大数据领域常用的数据仓库是Hive，Kylin是以Hive作为默认的数据源。

数据仓库的出现就是为了解决部分之间数据分散的问题（数据孤岛）。



数据仓库分层设计：



## OLTP/OLAP

OLAP（Online Analyse Process），联机分析处理，大量历史数据为基础，配合时间点的差异，以多维度的方式分析数据，一般带有主观的查询需求，多应用在数据仓库。

OLTP（Online Transaction Process），联机事务处理，侧重于数据库的增删改查等常用**业务**操作。

## 维度和度量

这两个是数据分析领域常用的概念。

维度（Dimension）简单来说就是你观察数据的角度，也就是数据记录的一个属性，例如时间、地点等。

度量（Measure）就是基于数据所计算出来的考量值，通常就是一个数据，比如总销售额，不同的用户数量。我们就是从不同的维度来审查度量值，以便我们分析找出其中的变化规律。

对应我们的SQL查询，GROUP BY的属性通常就是我们考量的维度，所计算出来的比如sum(字段)就是我们需要的度量。

## 事实表和维度表

事实表包含业务过程中的度量值、指标或者事实，例如：销售表，它包含卖家ID，买家ID，商品ID，价格，交易时间。

查询表：查询表用于将索引关联到一系列信息，例如卖家表就是查询表，包含卖家ID到卖家名称，卖家所在城市、国家等的映射。

## 多维数据模型

### 星型模型(star schema)

### 雪花模型(snowflake schema)

### 区别

# 特点

其具备以下特点：

1. 本身不产生数据，其基础数据来源于生产系统中的操作数据
2. 基于查询的分析系统；复杂查询经常使用多表联结、全表扫描等，牵涉的数量往往十分庞大
3. 每次查询设计的数据量很大，响应时间与具体查询有很大关系
4. 用户数量相对较小，其用户主要是业务人员与管理人员
5. 由于业务问题不固定，数据库的各种操作不能完全基于索引进行
6. 以SQL为主要载体，也支持语言类交互
7. 总体数据量相对较大

Codd提出了关于OLAP的12条准则：

1：Multidimensional conceptual view OLAP 模型必须提供多维概念视图；

2：Transparency 透明性；

3：Accessibility 存取能力准则；

4：Consistent reporting performance 稳定的报表能力；

5：Client/server architecture 客户/服务器体系结构；

6：Generic dimensionality 维的等同性准则；

7：Dynamic sparse matrix handling 动态的稀疏矩阵处理；

8：Multi-user support 多用户支持能力；

9：Unrestricted cross-dimensional operations 非受限的跨维操作；

10：Intuitive data manipulation 直观的数据操纵；

11：Flexible reporting 灵活的报表生成；

12：Unlimited dimensions and aggregation levels. 不受限的维与聚集层次。

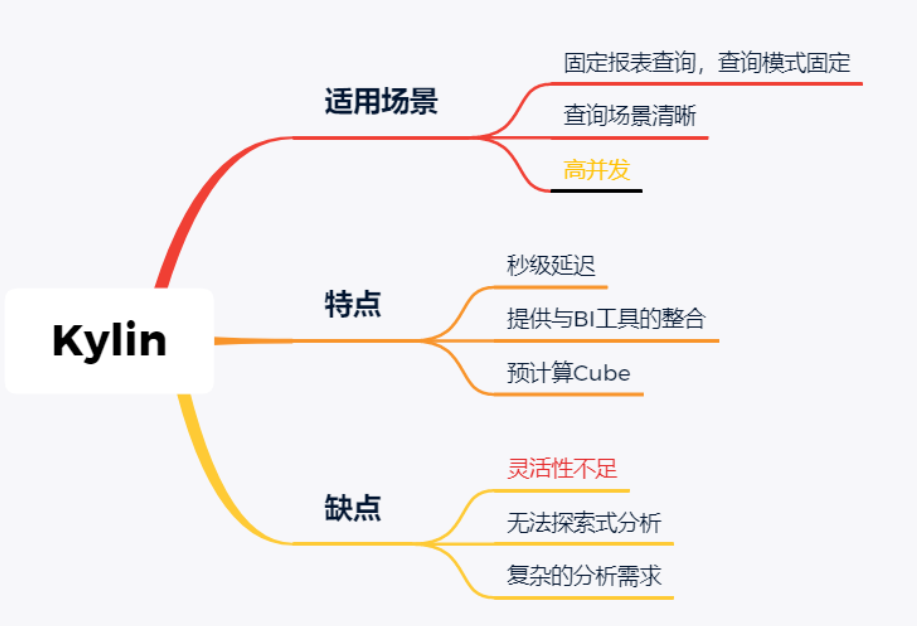
综上所述，OLAP系统强调了数据分析在系统中的重要性，对于速度等要求有着极高的要求。

# 服务工具

## Kylin

    Kylin的主要特点是预计算，提前计算好各个cube，这样的优点是查询快速，秒级延迟；缺点也非常明显，灵活性不足，无法做一些探索式的，关联性的数据分析。

    适合的场景也是比较固定的，场景清晰的地方。



## ClickHouse

Clickhouse由俄罗斯yandex公司开发。专为在线数据分析而设计。

Clickhouse最大的特点首先是快，为了快采用了列式储存，列式储存更好的支持压缩，压缩后的数据传输量变小，所以更快；同时支持分片，支持分布式执行，支持SQL。

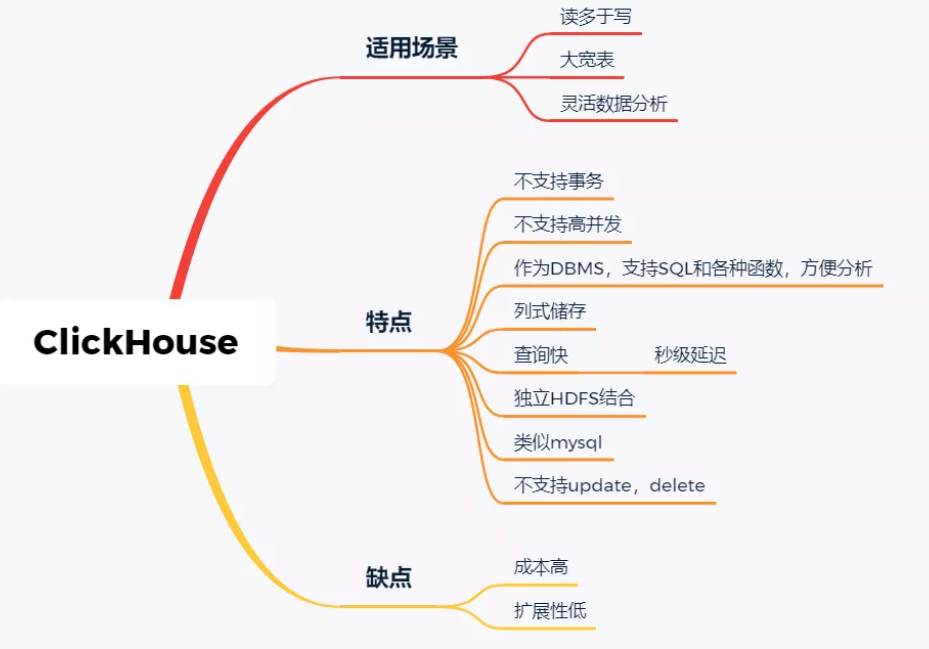
ClickHouse很轻量级，支持数据压缩和最终数据一致性，其数据量级在PB级别。

另外Clickhouse不是为关联分析而生，所以多表关联支持的不太好。

同样Clickhouse不能修改或者删除数据，仅能用于批量删除或修改。没有完整的事务支持，不支持二级索引等等，缺点也非常明显。

与Kylin相比ClickHouse更加的灵活，sql支持的更好，但是相比Kylin，ClickHouse不支持大并发，也就是不能很多访问同时在线。

总之ClickHouse用于在线数据分析，支持功能简单。CPU利用率高，速度极快。最好的场景用于行为统计分析。



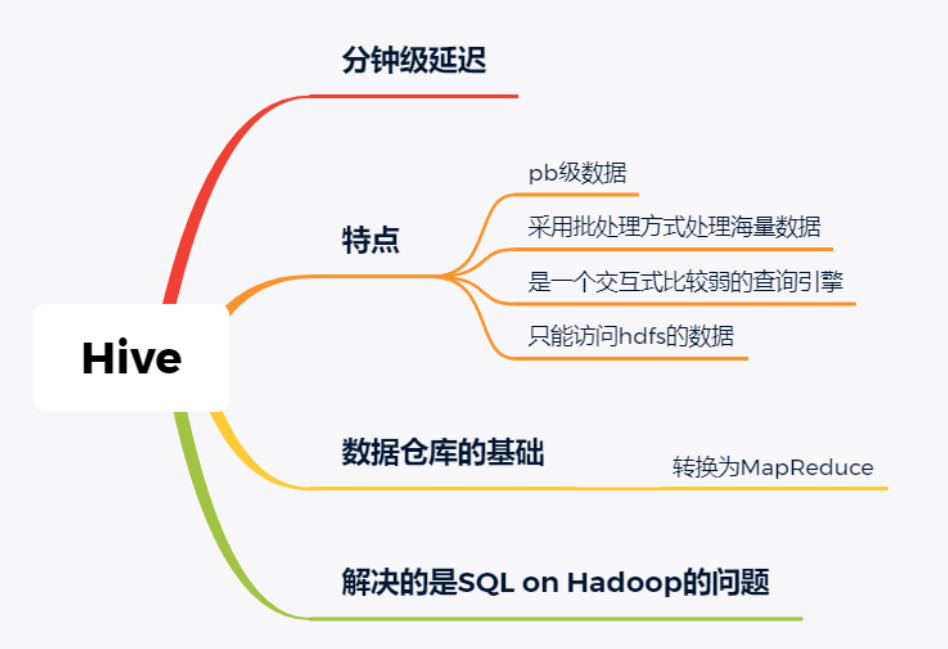
## Hive

Hive这个工具，大家一定很熟悉，大数据仓库的首选工具。可以将结构化的数据文件映射为一张数据库表，并提供完整的sql查询功能。

主要功能是可以将sql语句转换为相对应的MapReduce任务进行运行，这样可能处理海量的数据批量，

Hive与HDFS结合紧密，在大数据开始初期，提供一种直接使用sql就能访问HDFS的方案，摆脱了写MapReduce任务的方式，极大的降低了大数据的门槛。

当然Hive的缺点非常明显，定义的是分钟级别的查询延迟，估计都是在比较理想的情况。但是作为数据仓库的每日批量工具，的确是一个稳定合格的产品。



## Presto

Presto极大的改进了Hive的查询速度，而且Presto本身并不存储数据，但是可以接入多种数据源，并且支持跨数据源的级联查询，支持包括复杂查询、聚合、连接等等。

Presto没有使用MapReduce，它是通过一个定制的查询和执行引擎来完成的。它的所有的查询处理是在内存中，这也是它的性能很高的一个主要原因。

Presto由于是基于内存的，缺点可能是多张大表关联操作时易引起内存溢出错误。

另外Presto不支持OLTP的场景，所以不要把Presto当做数据库来使用。

Presto相比ClickHouse优点主要是多表join效果好。相比ClickHouse的支持功能简单，场景支持单一，Presto支持复杂的查询，应用范围更广。

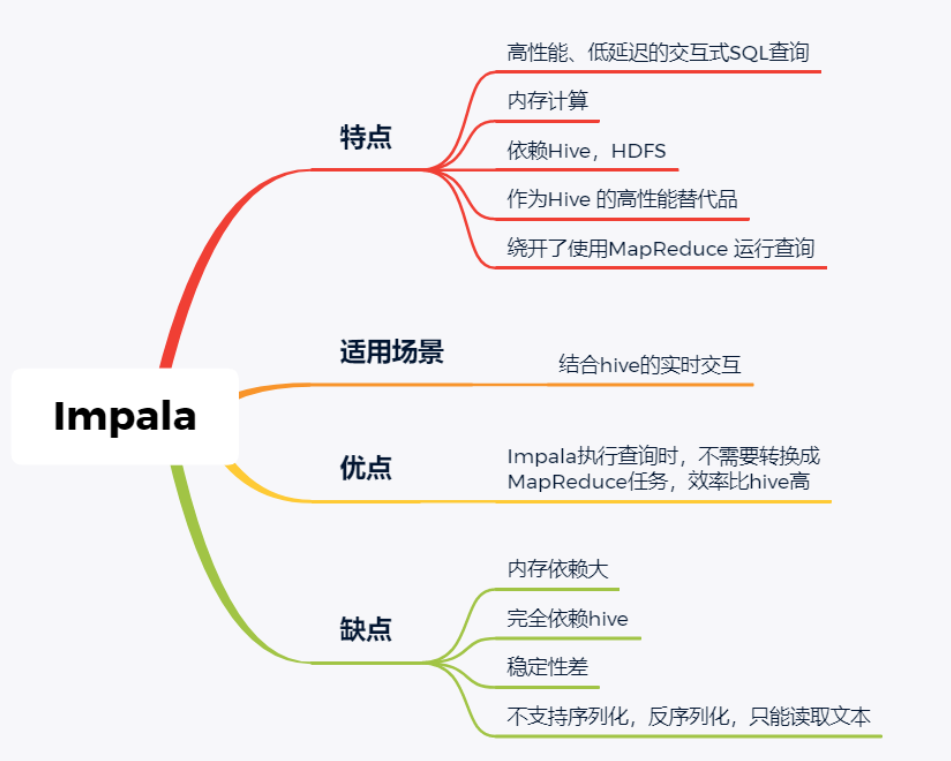


## Impala

    Impala是Cloudera 公司推出，提供对HDFS、Hbase数据的高性能、低延迟的交互式SQL查询功能。

    Impala使用Hive的元数据, 完全在内存中计算。是CDH平台首选的PB级大数据实时查询分析引擎。

Impala的缺点也很明显，首先严重依赖Hive，而且稳定性也稍差，元数据需要单独的mysql/pgsql来存储，对数据源的支持比较少，很多nosql是不支持的。但是，估计是cloudera的国内市场推广做的不错，Impala在国内的市场不错。



## SparkSQL

    SparkSQL的前身是Shark，它将SQL查询与Spark程序无缝集成，可以将结构化数据作为Spark的RDD进行查询。

    SparkSQL后续不再受限于Hive，只是兼容Hive。

    SparkSQL提供了sql访问和API访问的接口。

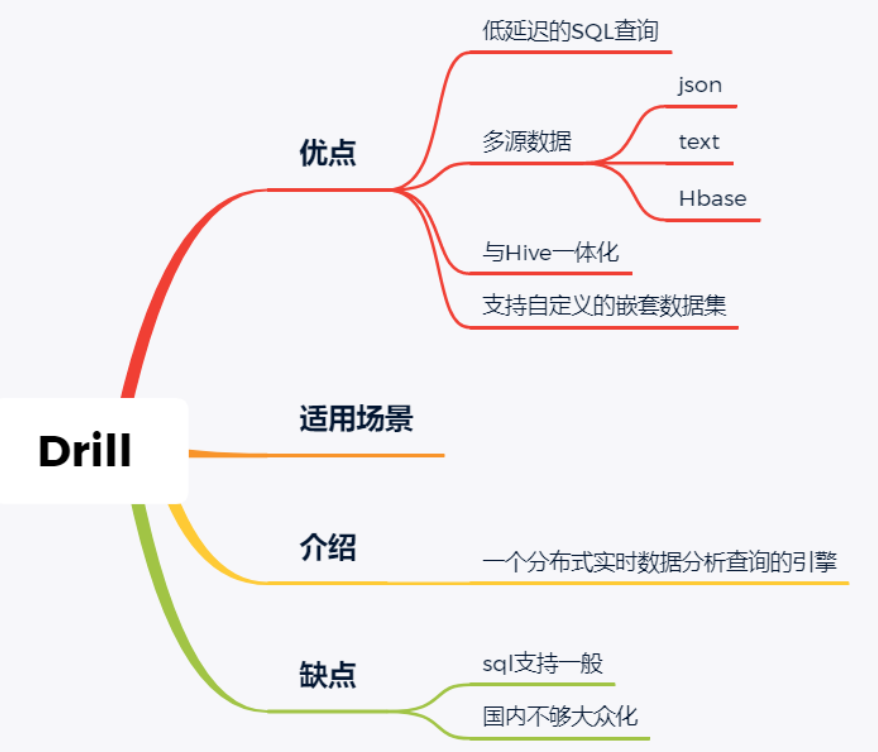
    支持访问各式各样的数据源，包括Hive, Avro, Parquet, ORC, JSON, and JDBC。



## Drill

Drill好像国内使用的很少，根据定义，Drill是一个低延迟的分布式海量数据交互式查询引擎，支持多种数据源，包括hadoop，NoSQL存储等等。

除了支持多种的数据源，Drill跟BI工具集成比较好。



## Druid

    Druid是专为海量数据集上的做高性能OLAP而设计的数据存储和分析系统。

    Druid的架构是Lambda架构，分成实时层和批处理层。

    Druid的核心设计结合了数据仓库，时间序列数据库和搜索系统的思想，以创建一个统一的系统，用于针对各种用例的实时分析。Druid将这三个系统中每个系统的关键特征合并到其接收层，存储格式，查询层和核心体系结构中。

    目前Druid的去重都是非精确的，Druid适合处理星型模型的数据，不支持关联操作。也不支持数据的更新。

Druid最大的优点还是支持实时与查询功能，解约了很多开发工作。



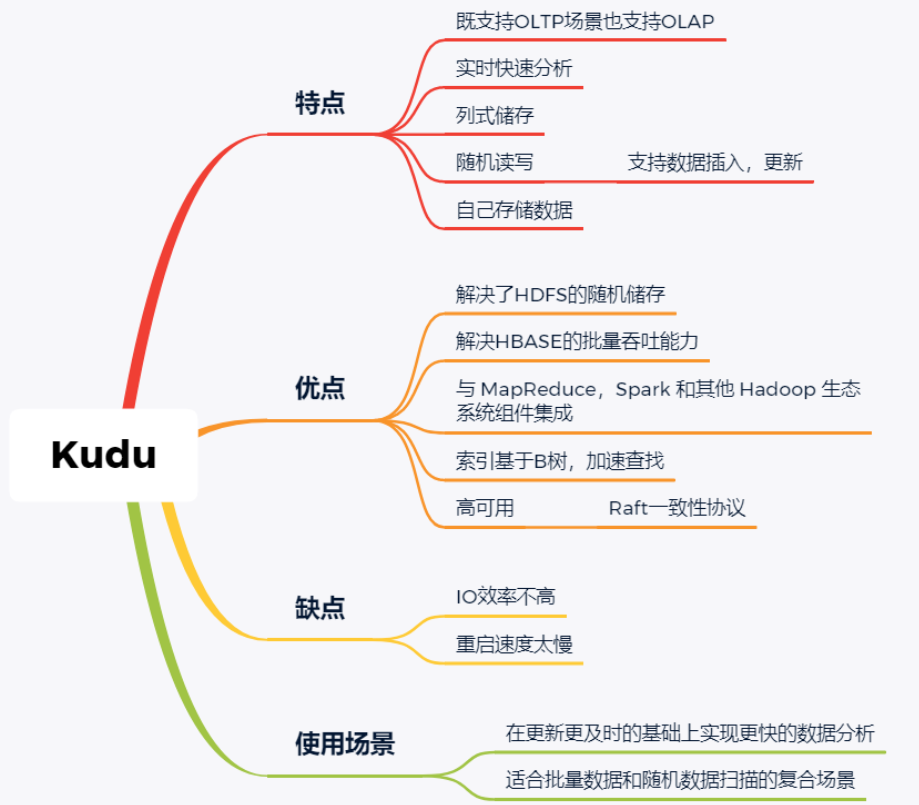
## Kudu

    kudu是一套完全独立的分布式存储引擎，很多设计概念上借鉴了HBase，但是又跟HBase不同，不需要HDFS，通过raft做数据复制；分片策略支持keyrange和hash等多种。

    数据格式在parquet基础上做了些修改，支持二级索引，更像一个列式存储，而不是HBase schema-free的kv方式。

    kudu也是cloudera主导的项目，跟Impala结合比较好，通过impala可以支持update操作。

kudu相对于原有parquet和ORC格式主要还是做增量更新的。



## Hbase

Hbase使用的很广,更多的是作为一个KV数据库来使用，查询的速度很快。



## Hawq

Hawq是一个Hadoop原生大规模并行SQL分析引擎，Hawq采用MPP架构，改进了针对Hadoop的基于成本的查询优化器。

除了能高效处理本身的内部数据，还可通过PXF访问HDFS、Hive、HBase、JSON等外部数据源。HAWQ全面兼容SQL标准，还可用SQL完成简单的数据挖掘和机器学习。无论是功能特性，还是性能表现，HAWQ都比较适用于构建Hadoop分析型数据仓库应用。