# 概述

# 原理

## 数据库锁

## 写时复制

## MVCC

## 读写分离

## 负载均衡

## 池化

### 线程池

### 连接池

## 热点数据

# 优秀方案

## TDSQL

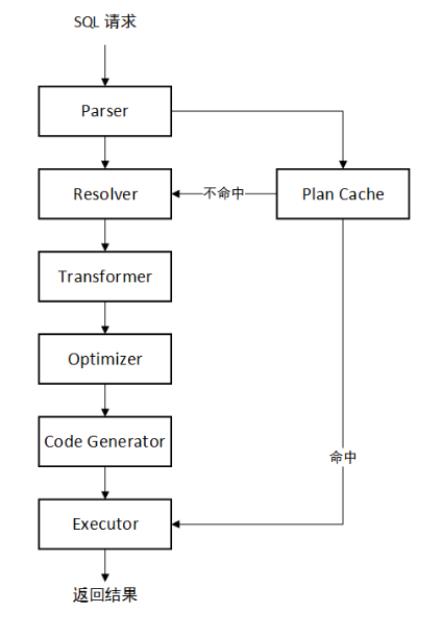
## OceanBase

### OBProxy

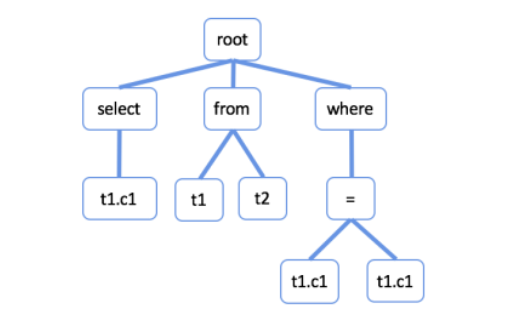
#### 读写分离

### SQL引擎

#### 快速参数化



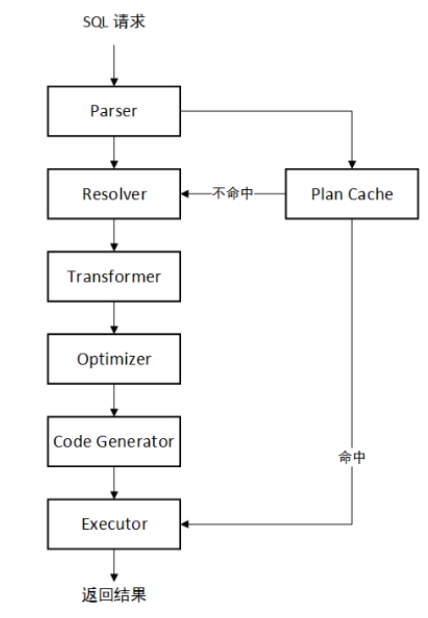
Parser（词法/语法解析模块）：  
 在收到用户发送的SQL请求串后， Parser会将字符串分成一个个的“单词”，并根据预先设定好的语法规则解析整个请求，将SQL请求字符串转换成带有语法结构信息的内存数据结构，我们称为“语法树”（Syntax Tree）。  
 为了加速SQL请求的处理速度，OceanBase对SQL请求采用了特有的**“快速参数化”**，以加速查找plan cache的速度。



#### 缓存执行计划树

#### 查询重写

#### 并行优化



Optimizer（优化器）：  
 优化器是整个SQL请求优化的核心，其作用是为SQL请求生成最佳的执行计划。  
 在优化过程中，优化器需要综合考虑SQL请求的语义、对象数据特征、对象物理分布等多方面因素，解决访问路径选择、连接顺序选择、连接算法选择、  
分布式计划生成等多个核心问题，最终选择一个对应该SQL的最佳执行计划。  
 为了充分利用OceanBase的分布式架构和多核计算资源的优势，OceanBase的查询优化器会对执行计划做并行优化：**根据计划树上各个节点的数据分布，对串行执行计划进行自底向上的分析，把串行的逻辑执行计划改造成一个可以并行执行的逻辑计划**。

#### 执行器

## TiDB

## GoldenDB

### proxy

#### LVS

#### 线程池

#### 连接池

#### 读写分离

#### 黑名单

对于某些耗时比较久的SQL，会加入黑名单中。

### SQL引擎

#### 缓存执行计划树

#### 查询重写

#### 并行优化

对于UNION可以采用并行的方式，分发到不同group。

#### 元数据缓存

Proxy本地会缓存元数据，当MDS发生元数据变更的时候会推送到proxy。

#### force index

#### MPP

### DB

#### 快同步

### Loadserver

#### 多线程

将文件拆分为多个，分别用不同线程处理。

### 存在问题

1. 没有处理group的热点数据

有对应的监控信息，但是这不能在线解决热点数据，只能作为事后处理。

1. 由于元数据推送问题导致读写分离错误

有这样的一个情况，CM会在主备关系发生变化的时候才会推送给MDS最新的元数据（主备关系），此时如果dbproxy重启，则会自动重置主备关系，这个时候可能就会发送到旧的备机（select操作），这个应该让CM定期发送（而不是变化的时候才发送），proxy可以获取最新的主备关系。