多行更新设计文档

修订历史

版本	修订日期	修订描述	作者	备注
0.1	2015-12-31	多行更新设计文档	王嘉豪	

1 需求分析

1.1 综述

OceanBase是可扩展的关系数据库，支持的是海量数据的查询和存储。但在做更新事物时，仅支持条件包含主键的单行数据的更新操作，而在某些应用场景下需要数据库支持多行的更新。

本文档介绍了原OceanBase数据库更新操作的流程，并对其原有流程进行了修改，在保留原主键更新执行流程不变的基础上，增加了对多行更新的支持，方便了用户使用。

但是，在这个版本中，不希望引入太多的改动量，实现方案对于更新数据量的大小有限制，一次更新的数据量不能太多，而且隔离级别仅仅支持到了Read Commit，不会出现“脏读”，但是可能出现“不可重复读”和幻读现象。

1.2 名词解释

RS：RootServer，管理集群内所有服务器，以及Tablet的分布和副本管理，在三集群架构中，负责集群间选主。

UPS：UpdateServer，存储更新数据，生成CommitLog，并将其同步到备机和写入到本地磁盘。

MS：MergeServer，接收客户端的链接，并生成相应的执行计划并执行，最后将结果返回给客户端。

CS：ChunkServer，存储基线数据。

1.3 功能描述

1.3.1 原OceanBase0.4版本的功能描述

- 功能：
  更新或删除一条数据
- 语法：
  `UPDATE tab1 SET col1=val1 [col2=val2] ... WHERE cond1 [and cond2] …`
  `DELETE FROM tab1 WHERE cond1 [and cond2] …`
- 限制：
  仅支持单表删除或更新
  仅支持给定主键的删除或更新（condition必须包含全部的主键信息）
  不支持where子查询

1.3.2 需求功能描述

- 功能：
  更新或删除多条数据
- 语法：
  `UPDATE tab1 SET col1=val1 [col2=val2] ... [ WHERE cond1 [and cond2] …]`
  `DELETE FROM tab1 [ WHERE cond1 [and cond2] …]`
- 限制：
  仅支持单表删除或更新
  不支持where子查询
  更新的数据大小有上限

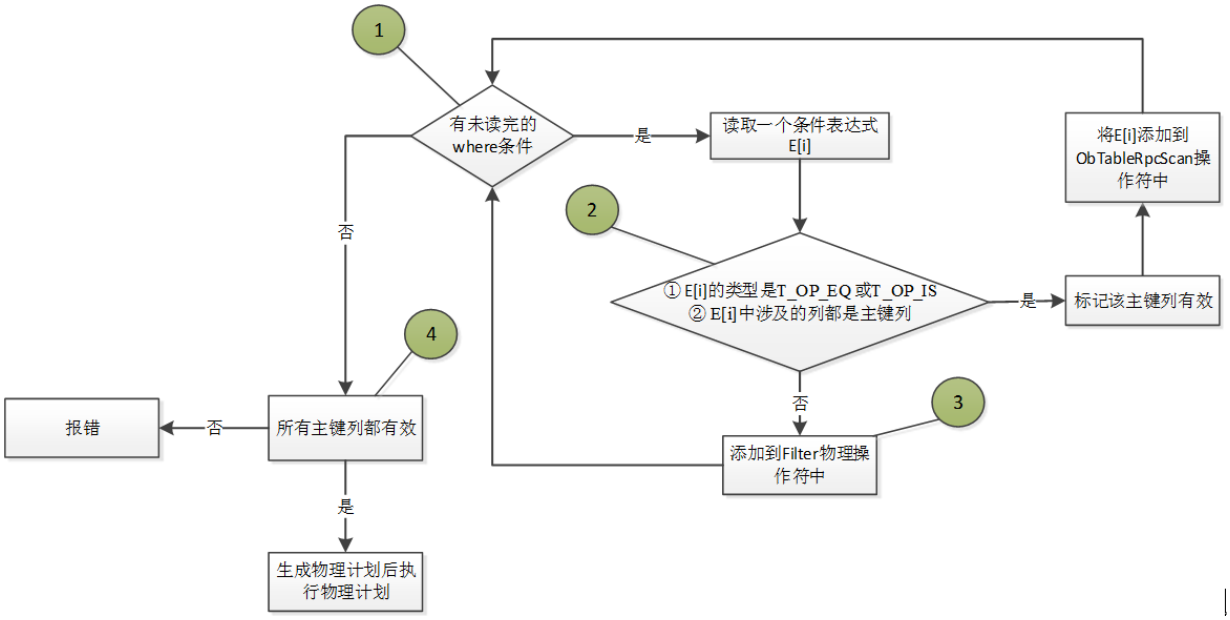
2 概要设计

2.1 原有OceanBase的更新计划

2.1.1 计划概述

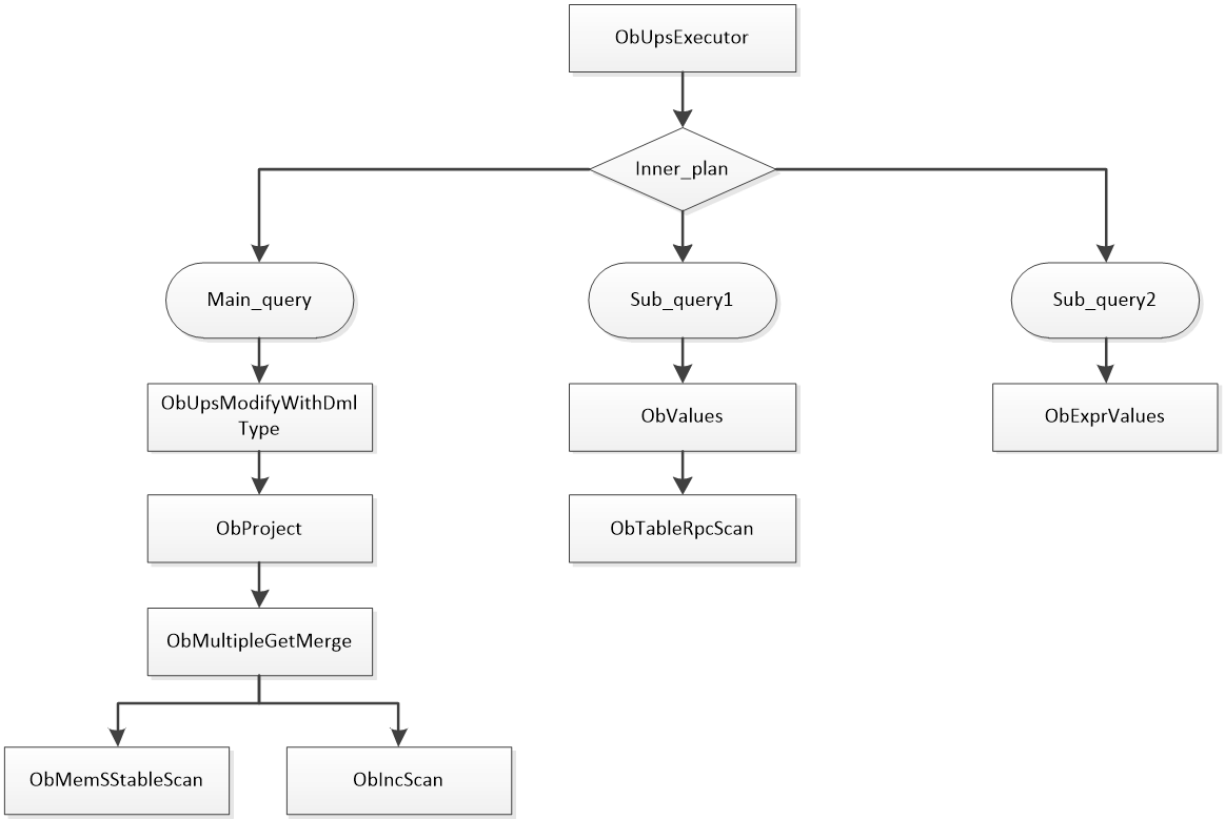
客户端首先连接MS，并发送更新或删除请求，MS收到请求之后，生成相应的逻辑计划和物理计划。之后执行MS端的物理计划，首先从CS拉取静态数据，CS将静态数据和UPS上的冻结数据融合之后，返回给MS，然后MS把静态数据和在UPS端执行的物理计划序列化之后发给UPS变更请求。UPS收到请求后，把动态数据和静态数据结合后，进行更新操作。

2.1.2 生成物理计划时where条件判断



1. 读取一条where条件表达式E[i]，如果已经读取完毕，则转(4)
2. 对E[i]进行检查，如果E[i]同时满足：a)它的类型是T_OP_EQ或者T_OP_IS。 b)它涉及的列是主键。则将这个表达式添加到ObTableRpcScan物理操作符中。如果两个条件有一个不满足，则转(3)，如果都满足，则转(1)。
3. 将这个条件表达式添加到ObFilter物理操作符中
4. 检查where条件中是否包含所有的主键列，全部包含，则继续处理其他内容，如果没有，则设置错误码。
   由于(2)和(4)的检查，所以有OB更新where条件的限制。

2.1.3 Update语句物理计划如下



- sub_query1和sub_query2是在MS执行的物理计划，Main_query是在UPS执行的物理

计划。下面介绍在MS执行的物理操作符的作用。

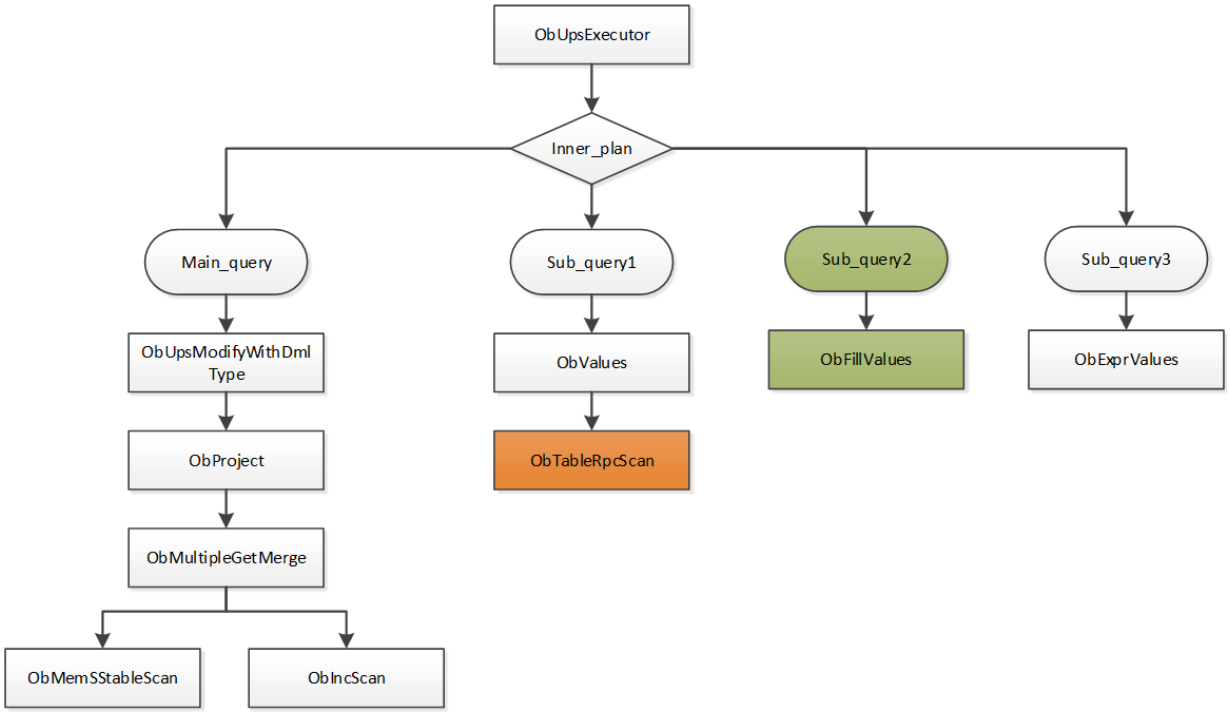
- ObValues：存储要删除或更新的静态数据。在UPS上执行物理计划时，物理操作符ObIncScan会读取这些数据。
- ObTableRpcScan：根据where条件中的各个表达式构造查询range，并将这些range发送到合适的CS，将读取到的数据发送到MS进行合并最终获得结果。
- ObExprValues：存放where条件中主键值，这些值用来在UPS中定位要删除或更新的数据。

2.2 修改后的更新计划

2.2.1 计划概述

对生成物理计划的实现中，对where条件判断流程做出小改：当条件涵盖全部主键列时，按照原有的计划生成；当条件不足时（不涵盖全部主键列），继续执行多行更新的物理计划。修改后的物理计划，会把满足条件的行的主键值全部发给UPS，UPS的事物执行计划不改变。

2.2.2 修改后的物理计划



- 新增一个sub_query，对应操作符为新增的ObFillValues。修改ObTableRpcScan的参数，由原来的get方法现改为get或scan方法（由where条件确定）。
- 修改思路大致如下：由于三个sub_query是顺序执行的，sub_query1中获取到的所有需要更新的条目数据，在sub_query2中将sub_query1中数据的主键取出，制作成表达式，填充到sub_query3中的ObExprValues操作符中。最后序列化ObValues和ObExprvalues发送给UPS进行事务执行。

3 详细设计

3.1 ObFillValues操作符的实现

操作符主要实现了将ObValues中的行生成表达式放入ObExprValues中。

- Open函数伪代码：

```
Open()
{
    For each_row in ObValues
    {
        根据each_row中的主键构建表达式output_expr
        将表达式output_expr，填充至ObExprvalues中
    }
}
```

3.2 ObTableRpcScan操作符的参数改动

- ObTableRpcScan在执行之前，需要设定参数：①根据where的条件设定cs拉数据的策略（Scan，get）以及②数据的一致性选择（static，frozen，weak，strong）。之前的update由于有指定的主键，使用get方法，数据一致性选择了frozen。改动之后，在where条件为非全主键时，由于需要更新的主键未知，故需要使用scan方法，数据一致性选择至少为weak，最好为strong。
- 读取一致性策略选择代码

```
ObString name = ObString::make_string(OB_READ_CONSISTENCY);
ObObj value;
int64_t read_consistency_level_val = 0;
hint.read_consistency_ = common::STRONG;
sql_context_->session_info_->get_sys_variable_value(name, value);
value.get_int(read_consistency_level_val);
hint.read_consistency_ = static_cast<ObConsistencyLevel>(read_consi
stency_level_val);
```

- 读取策略代码

```
int32_t read_method = ObSqlReadStrategy::USE_SCAN;
ObArray<ObRowkey> rowkey_array;
sql_read_strategy.get_read_method(rowkey_array, rowkey_objs_allocat
or, read_method);
hint.read_method_ = read_method;
```

3.3 ObTransformer的改动

- 增加`int ObTransformer::gen_phy_table_for_update_more()`方法，该方法主要是在原有方法`int ObTransformer::gen_phy_table_for_update()`上做的修改，主要功能是构建update的物理计划。