版本说明

版本号：0.3.1
版本发布时间：2017年9月29日

Cedar是基于开源的OceanBase 0.4.2的开发版本，而Cedar0.3是继16年开源版本0.2之后的又一稳定且功能更为丰富的系统版本。该版本保留了0.2版本的所有特性，并在此基础上增加了一些全新的功能模块。

特征列表

1 新功能特征

1.1 Decimal数据类型

优化DECIMAL基本数据类型，该数据类型支持38位有效数字，可有效减少数据库在存储、计算数据时精度的丢失。

- **DECIMAL基本数据类型的功能**
  支持insert、replace、update、delete、select等操作；
  支持聚合函数avg( )、count( )、sum( )、max( )、min( )、 mod( )等操作；
  支持系统函数cast( )、coalesce( )等函数；
  支持逻辑运算符(=、>=、>、<=、<、!=(<>))；
  支持运算符(between and、in、is null、is not null、not、and、or)；
  支持集合操作(union、except、intersect)；
  Decimal类型数据可以作为连接属性的支持情况；
  Decimal类型数据可以作为数据表和二级索引的主键。
- **DECIMAL基本数据类型的类型转换**
  支持其他数据类型与DECIMAL数据类型的相互转换，例如
  bool_decimal( )、 decimal_bool( )
  int_decimal( )、decimal_int( )
  float_decimal( )、 decimal_float( )
  double_decimal( )、 decimal_double( )
  varchar_decimal( )、 decimal_varchar( )
- **DECIMAL基本数据类型的使用**

建立含有DECIMAL数据类型的表：
CREATE TABLE *table_name* (c1 INT, c2 DECIMAL(30,5), PRIMARY KEY(c1))

插入DECIMAL列数据：
INSERT INTO *table_name* VALUES (1, 100.001)
REPLACE INTO *table_name* VALUES (2, 200.001)

修改DECIMAL列数据：
UPDATE *table_name* SET c2 = c2 + 10.123 WHERE c1 = 2

删除DECIMAL列数据：
DELETE FROM *table_name* WHERE c1 = 2

1.2 支持主键自增(auto_increment)

增加数据库主键自增的功能，该功能是指在插入新数据行时自动生成特定主键列的值。

- **主键自增的功能**
  允许用户定义表结构时给主键添加AUTO_INCREMENT关键词标记；
  主键列被定义为自增列后，且在没有给定相应主键值的情况下，进行插入数据操作（REPLACE或INSERT）时，该主键列值会进行自动生成（默认从1开始自增）。
- **主键自增的机制**
  新增系统表 __all_auto_increment 记录每张表自增的主键当前最大值，每次插入新数据行时，互斥地读取出该值，并自增1做为新数据行的主键值，同时更新系统表中的记录。
- **主键自增的使用**
  增加SQL语句关键词标记，以满足主键自增的使用：

构建主键自增条件：
CREATE TABLE *table_name* (id int primary key AUTO_INCREMENT, col int)

主键自增使用：
情形一：使用自增的主键值
REPLACE(或INSERT) INTO user (*us_name*) VALUES (*name_value*)
情形二：指定主键值
REPLACE(或INSERT) INTO user (*us_id* ,*us_name*) VALUES (*id_value*, *name_value*)
指定的主键值若大于最大主键，则会更新系统表，下次插入会从新的主键值开始自增。

删除主键自增：
删除表时将自动删除系统表 __all_auto_increment 中与该表相关的所有记录。

使用限制：

1. 只能标记主键列；
2. 最多只能标记一个列；
3. 只能标记数值类型 (bigint, int, integer, mediumint, smallint, tinyint, float, double, real)列。

1.3 清空表(Truncate Table)

增加truncate table 功能后，在如下场景可能会得到应用：

> 表中数据量非常大，且不需要回滚，删除数据后保留表的结构。

- **Truncate Table的功能**
  允许用户大量删除数据时保留表的结构。
- **Truncate Table的机制**
  仿照UPS中行Btree数据结构+行锁，实现表级Btree+表锁，用于保存表级truncate信息。
- **Truncate Table的使用**
  增加SQL语句，以满足truncate table的使用：

创建二级索引：
truncate table [if exists] table_name1，table_name2, …, table_nameN
允许truncate多张表、成功返回”0 rows affected”、失败返回错误代码。

使用限制：
truncate之后表不允许更新直至下一次memtable冻结完成（不等合并完成即可恢复写），memtable 冻结手动触发。

手动版本合并：

1. 主动发起大版本合并(工具在oceanbase/bin目录下)：
   ./ups_admin -a ip -p port -t major_freeze
2. 主动发起小版本合并：
   ./ups_admin -a ip -p port -t minor_freeze

1.4 事务传输量—支持单次32M包

该功能可以支持超过2MB的事务执行，弥补数据库系统应用场景的局限性。

- **可配置的拓展2MB事务的功能**
  允许用户配置定义事务的写数据量的大小限制(最大可配置到1GB)；
  支持以存储过程定义的限制大小内的事务执行。
- **可配置的拓展2MB事务的机制**
  新增UpdateServer系统配置项max_log_buffer_size；
  新增UpdateServer系统配置项log_buffer_max_size；
  新增UpdateServer系统配置项block_bits；
  新增存储过程调用(CALL)语句hint(LONG_TRANS)；
  增加长事务执行流程。
- **可配置的拓展2MB事务的使用**

修改系统配置项：
ALTER SYSTEM SET max_log_buffer_size = 33554432 SERVER_TYPE = UPDATESERVER //33554432 =32 MB（默认1.875M）
ALTER SYSTEM SET log_buffer_max_size = 67108864 SERVER_TYPE = UPDATESERVER //67108864=64MB（默认2M）
ALTER SYSTEM SET block_bits = 26 SERVER_TYPE = UPDATESERVER //1<<26=64MB（默认22）
ALTER SYSTEM SET log_cache_block_size = '64MB' SERVER_TYPE = UPDATESERVER （默认32M）
ALTER SYSTEM SET commit_log_size = '128MB' SERVER_TYPE = UPDATESERVER（默认64M）

长事务调用：
CALL producename()/*+LONG_TRANS*/；
CALL producename()/*+NO_GROUP LONG_TRANS*/ //如果使用了for循环，在循环次数很多时需要增加NO_GROUP不成组执行优化。

1.5 哈希连接(HashJoin)

增加哈希连接(HashJoin)，支持两张或两张以上的大表进行连接，首先通过对前表构建Hash Table，然后流水线处理后表每行数据做连接，一方面不再需要对后表进行内存中的缓存和排序，另一方面对于选择率较高的查询明显降低了连接计算的时间，生产环境中检验证明Hash Join的应用显著提高了多表连接查询的性能。

- **哈希连接查询机制**
  增加物理操作符ObHashJoin；
  修改查询处理流程，使其支持哈希连接查询。
- **哈希连接的使用**
  增加hint语法，用户可以通过在SQL语句中添加hint来决定是否使用哈希连接优化。

/*+JOIN(HASH_JOIN，MERGE_JOIN，，，***join_type***)*/