OceanDB重建索引

OceanDB官方文档

常用sql命令

```
--- 删除所有的表，将查询结果执行一遍
SELECT CONCAT('DROP TABLE IF EXISTS ', table_name, ';') FROM
information_schema.tables WHERE table_schema = '数据库名';
--- 查看版本号，目前版本号：5.7.25-OceanBase-v3.2.4.5
SELECT version();
--- 获取每个表的字段
show columns from zenger.algojr_tanlzindex2;
--- 查看表索引
select * from information_schema.statistics a
where table_schema=database() AND lower(a.`TABLE_NAME`) =
lower('thisstp_makerduty') AND a.`INDEX_NAME` = 'PRIMARY';
--- 表数据备份
INSERT INTO table_name SELECT * FROM table_bak;
```

OceanDb3.x主键修改

由于OceanDb3.x不支持索引修改，每次都需要删表重建，为保留原有数据，需要进行原表的数据和结构的拷贝：

1. 获取要修改的主键，判断该主键是否已存在
2. 获取要进行主键修改的表（以`thisstp_makerduty`为例），获取其表创建的sql语句
    - `show create table zenger.algojr_tanlzindex;`
3. 根据`thisstp_makerduty`创建sql,新建一个备份表`thisstp_makerduty_bak`,将主键设置为要修改的主键
4. 将原表的数据拷贝一份到备份表
    - `INSERT INTO table_b SELECT * FROM table_a;`
    - 这个语句在满足如下两个条件会进行批量复制，速度非常快，10w的数据量几s搞定
        - 由于两个表的结构完全一致，不用进行类型检查与转换
        - 且备份表为空，直接将数据库拷贝到新页，也不用进行页的合并与分裂
5. 删除原表，将备份表的的名字重命名为原表
    - `DROP TABLE IF EXISTS table_name`
    - `ALTER TABLE algojr_tanlzindex1 RENAME TO algojr_tanlzindex2;`

脚本样例

在新建索引时，注意检查主键字段的顺序：

```
set @hs_sql1 = 'select 1 into @hs_sql1;';
set @hs_sql2 = 'select 1 into @hs_sql2;';
set @hs_sql3 = 'select 1 into @hs_sql3;';
set @hs_sql4 = 'select 1 into @hs_sql4;';
```

```
set @v_rowcount = 0;
SELECT count(1) INTO @v_rowcount from dual where (select count(1) from
information_schema.statistics a where table_schema=DATABASE() and
lower(a.`TABLE_NAME`) = 'thisalgojr_toperatelog' AND lower(a.`INDEX_NAME`) =
'primary' and lower(a.`COLUMN_NAME`)='business_date' and a.`SEQ_IN_INDEX` = 1)=1
and (select count(1) from information_schema.statistics a where
table_schema=DATABASE() and lower(a.`TABLE_NAME`) = 'thisalgojr_toperatelog' AND
lower(a.`INDEX_NAME`) = 'primary' and lower(a.`COLUMN_NAME`)='position_str' and
a.`SEQ_IN_INDEX` = 2)=1
and (select count(1) from information_schema.statistics a where
table_schema=DATABASE() and lower(a.`TABLE_NAME`) = 'thisalgojr_toperatelog' AND
lower(a.`INDEX_NAME`) = 'primary' and lower(a.`COLUMN_NAME`)='input_date' and
a.`SEQ_IN_INDEX` = 3)=1;
select 'CREATE TABLE `thisalgojr_toperatelog_bak` (
  `business_date` int(11) NOT NULL DEFAULT ''0'',
  `company_id` int(11) DEFAULT ''0'',
  `create_time` int(11) DEFAULT ''0'',
  `extsystem_id` int(11) DEFAULT ''0'',
  `input_date` int(11) NOT NULL DEFAULT ''0'',
  `message` varchar(4000) COLLATE utf8mb4_bin DEFAULT '''',
  `operate_type` char(1) COLLATE utf8mb4_bin DEFAULT '''',
  `operator_no` int(11) DEFAULT ''0'',
  `position_str` varchar(128) COLLATE utf8mb4_bin NOT NULL DEFAULT '''',
  `scheme_batch_no` int(11) DEFAULT ''0'',
  `scheme_code` varchar(64) COLLATE utf8mb4_bin DEFAULT '''',
  `scheme_ins_code` varchar(64) COLLATE utf8mb4_bin DEFAULT '''',
  `strategy_id` int(11) DEFAULT ''0'',
  `third_no` int(11) DEFAULT ''0'',
  `third_remark` varchar(256) COLLATE utf8mb4_bin DEFAULT '''',
  `algobus_front_id` int(11) NOT NULL DEFAULT ''0'',
  `log_type` int(11) NOT NULL DEFAULT ''0'',
  `log_level` int(10) DEFAULT NULL,
  PRIMARY KEY (`business_date`, `position_str`, `input_date`),
  KEY `idx_histoperatelog_date_scheme` (`business_date`, `scheme_code`) BLOCK_SIZE
16384 LOCAL
);' into @hs_sql1 from dual where @v_rowcount = 0;
select 'insert into thisalgojr_toperatelog_bak (business_date, company_id,
create_time, extsystem_id, input_date, message, operate_type, operator_no,
position_str, scheme_batch_no, scheme_code, scheme_ins_code, strategy_id,
third_no, third_remark, algobus_front_id, log_type, log_level)
SELECT business_date, company_id, create_time, extsystem_id, input_date, message,
operate_type, operator_no, position_str, scheme_batch_no, scheme_code,
scheme_ins_code, strategy_id, third_no, third_remark, algobus_front_id, log_type,
log_level
from thisalgojr_toperatelog;' into @hs_sql2 from dual where @v_rowcount = 0;
select 'DROP TABLE IF EXISTS thisalgojr_toperatelog;' into @hs_sql3 from dual
where @v_rowcount = 0;
select 'ALTER TABLE thisalgojr_toperatelog_bak RENAME TO thisalgojr_toperatelog;'
into @hs_sql4 from dual where @v_rowcount = 0;
PREPARE stmt FROM @hs_sql1;
EXECUTE stmt;
PREPARE stmt FROM @hs_sql2;
EXECUTE stmt;
PREPARE stmt FROM @hs_sql3;
```

```
EXECUTE stmt;
PREPARE stmt FROM @hs_sql4;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

写个脚本生成sql脚本

为进行文本匹配、批量脚本生成、生成数据库模拟数据，建议写一个脚本进行操作，同时通过多线程加速程序运行（如go开两个协程的情况下，速度是原来的3倍）

如果用go写脚本需要先安装OceanDB驱动：

go安装OceanDB驱动

下载驱动到本地后，执行：`go install \path\to\github.com\go-sql-driver\mysql`,如`D:\GO\github.com\go-sql-driver\mysql`

建议用网络自动进行包管理：

```
# 通过代理，不行
set http_proxy=http://127.0.0.1:7890
# 通过镜像，ok
go env -w GOPROXY=https://goproxy.cn,direct
go get -u github.com/go-sql-driver/mysql
```

在 Go 中，安装的包会被下载到 $GOPATH/pkg/mod 目录下，并根据版本号存储在相应的子目录中。因此，可以通过以下步骤来查看 github.com/go-sql-driver/mysql 包的安装路径：

1. 打开命令行或终端窗口,执行命令 go env GOPATH，查看您的 GOPATH 路径。
2. 在 $GOPATH/pkg/mod 目录下找到 github.com/go-sql-driver/mysql 目录，其中的子目录名称应该与安装的版本号相同