1. Export时采用show table产生CREATE语句，存入CREATE.sql文件中。

不能直接采用DBC.tables中的requesttext字段中的语句，因为此字段中存的语句未必是CREATE语句，这可能导致表不一致，或者存在表依赖时无法创建成功等问题。

1. Export生成CREATE语句的同时构造相应的DROP语句，用于Rollback。DROP语句的顺序是CREATE的反序；CREATE按建表时间戳顺序。
2. 生成CREATE和DROP时，需要去掉语句中的数据库名信息。

如：create dbname.tablename…形式需要变换为create tablename形式，因为导入和导出的数据库名可能不相同。

1. SystemFE中的表均由DBC创建，pct\_hist\_usr对systemfe无R权限，但是对DBC的ALL有R权限，因而可以访问systemfe的表。
2. SYSLIB中的表（其实是FUNCTION）也均由DBC创建。因此在实现权限check时，先check对DBC的ALL是否有R权限，若有，则直接跳过对单个表的检查。
3. SystemID, SystemName, DatabaseID, DatabaseName, TableID, 分别指的是什么？

20130328

ExportRun 参数：

第一个参数EF表示不导出PhysicalConfiguration和VirtualConfiguration

第二个参数F表示通过读取保存的CSV文件的记录存入PostgreSQL中，不加F则分别导出到CSV和PostgreSQL

目前支持的DDL类型：

|  |
| --- |
| switch(TableKind) {  case "T":  sql = CommonConfig.sqlShowTable + "\"" + TableName + "\"";  this.sqlListDrop.add(CommonConfig.sqlDropTable +  "\"" + TableName + "\"");  break;  case "V":  sql = CommonConfig.sqlShowView + "\"" + TableName + "\"";  this.sqlListDrop.add(CommonConfig.sqlDropView +  "\"" + TableName + "\"");  break;  case "M":  sql = CommonConfig.sqlShowMacro + "\"" + TableName + "\"";  this.sqlListDrop.add(CommonConfig.sqlDropMacro +  "\"" + TableName + "\"");  break;  case "P":  case "E":  sql = CommonConfig.sqlShowProcedure + "\"" + TableName + "\"";  this.sqlListDrop.add(CommonConfig.sqlDropProcedure +  "\"" + TableName + "\"");  break;  case "D":  logger.info(" -- TableKind is D, SKIP. -- ");  break;  case "R":  case "F":  sql = CommonConfig.sqlShowFunction + "\"" + TableName + "\"";  this.sqlListDrop.add(CommonConfig.sqlDropFunction +  "\"" + TableName + "\"");  break;  default:  break;  } |