QianBase 管理员手册 1.6.6

2020 年 12 月

目录

前言

本文简介

本文有针对性地并且详尽地描述了如何管理 QianBase 数据库。

目标读者

本指南的目标读者为 QianBase 管理员和用户。

修订历史

版本	日期	说明
1.6.6	2020 年 12 月	升级版本号，将 QianBase Manager 改成 Esgyn DBManager。
1.5.4	2020 年 3 月	新增 11.共享缓存管理 新增 15.数据库初始化命令
1.1.1	2019 年 12 月	修改 1.2 QianBase 组件与进程 改正了一些错别字等 新增 TRUNCATE 和 DUMP 语句
1.1.0	2019 年 7 月	第一版

符号约定

描述语法时，本指南采用以下符号约定。

- 大写字母

  表示关键字和保留字。未在方括号中的内容是必选的语法项。

  示例

  ```
  SELECT
  ```

- 小写字母

  表示变量。未在方括号中的内容是必选的语法项。

  示例

  ```
  file-name
  ```

- [] 方括号

  表示可选的语法项。

  示例

  ```
  DATETIME [start-field TO] end-field
  ```

  如果方括号中包括多个语法项，您可以选择某一语法项或不选。

  多个语法项可以垂直排列，每个语法选项用方括号括起；也可以水平排列，用竖线隔开。

  示例

  ```
  DROP SCHEMA schema [ CASCADE ][ RESTRICT ]
  DROP SCHEMA schema [ CASCADE | RESTRICT ]
  ```

- {} 大括号

  表示必选的语法项。

  示例

  ```
  FROM { grantee [, grantee ] ... }
  ```

  如果大括号中包括多个语法选项，您必须选择某一语法项。

  多个语法项可以垂直排列，每个语法项用大括号括起；也可以水平排列，用竖线隔开。

  示例

  ```
  INTERVAL { start-field TO end-field }{ single-field }
  INTERVAL { start-field TO end-field | single-field }
  ```

- | 竖线

  隔开方括号或大括号中的多个语法项。

  示例

  ```
  {expression | NULL}
  ```

- … 省略号

  - 紧跟在方括号或大括号之后，表示括号内的语法项可以重复任意次。

    示例

    ```
    ATTRIBUTE[S] attribute [, attribute] ...
    {, sql-expression } ...
    ```

  - 紧跟在单个语法项之后，表示该语法项可以重复任意次。

    示例

    ```
    expression-n ...
    ```

- 标点符号
  - 圆括号、逗号、分号和其它符号，请参照以下示例输入。

示例

```
DAY (datetime-expression)
@script-file
```

  - 符号（例如，方括号或大括号）周围的引号表示该符号是必选项。

示例

```
"{" module-name [, module-name] ... "}"
```

- 语法项的间距
  - 语法项之间必须有空格。如果某一语法项是圆括号或逗号，则不需要空格。

示例

```
DAY (datetime-expression) DAY(datetime-expression)
```

  - 如果语法项之间没有空格，则不允许空格。在以下示例中，句点与任何其他语法项之间不允许有空格。

示例

```
myfile.sh
```

- 行距

  如果一条命令过长（超过一行），则每个连续行需缩进三个空格，并通过空行与前一行分隔。

  行距用于区分连续行中的语法项与垂直列表（多个语法项）中的项。

示例

```
match-value [NOT] LIKE _pattern
   [ESCAPE esc-char-expression]
```

批评与建议

我们支持您对本指南做出的任何批评与建议，并尽力提供符合您需求的文档。

若您发现任何错误、或有任何改进建议，请发邮件至 support@esgyn.cn。

相关文档

本指南为 QianBase 文档库的一部分，QianBase 文档库**包括但不限于**以下文档:

文档名称	说明
QianBase 安装部署指南	本文介绍安装 QianBase，包括安装前准备、安装 Hadoop 发行版、故障排除、配置、启用安全功能、提高安全性和卸载 QianBase 等。
易鲸捷 Designer 用户指南	本文介绍易鲸捷图形化数据库管理工具。
易鲸捷迁移工具用户指南	本文介绍如何安装和使用易鲸捷迁移工具。
ODB 用户指南	本文介绍了如何使用 odb（一种基于 ODBC 的多线程命令行工具）在易鲸捷数据库上执行各种操作。
易鲸捷加载转换指南	本文介绍如何将数据加载转换到易鲸捷数据库。
QianBase 技术白皮书	本文介绍 QianBase 技术架构，组件介绍，技术特点等。
QianBase 数据库规划文档	本文介绍节点数量规划、数据目录和安装部署目录规划、集群角色分配规划等。
QianBase 管理员手册	本文介绍 QianBase 的日常运维常用系统命令、常用检查 SQL，用户权限配置，连接设置等内容。
QianBase 常见问题提排查与解决	本文介绍如何排查和解决 QianBase 的常见问题。
QianBase 灾难恢复手册	本文介绍 QianBase 灾难恢复设计原理，方案建议以及使用手册。

QianBase 备份恢复手册	本文介绍 QianBase 备份恢复设计原理，方案建议以及使用手册。
QianBase 数据库扩容指南	本文介绍 QianBase 如何更换节点，增加节点，删除节点等操作。
QianBase 数据库参数调优建议	本文介绍如何进行数据模型优化，CQD 参数优化等 。
QianBase 客户端安装手册	本文介绍 QianBase JDBC，ODBC 以及 Trafci 驱动安装。
QianBase JDBC 程序员参考指南	本文介绍 QianBase JDBC 驱动连接设置，开发人员指南。
QianBase ODBC 程序员参考指南	本文介绍 QianBase ODBC 驱动连接设置，开发人员指南。
QianBase SPSQL 存储过程用户手册	本文介绍 QianBase SPSQL 存储过程的使用。
Esgyn DBManager 用户手册	本文介绍图形化数据库监控运维工具 DB Manager 的使用。
QianBase 数据库迁移指南	本文介绍如何将常见关系型数据库（Oracle、MySQL、SQL server 等）迁移至 QianBase。
QianBase SQL 用户手册	本文是 QianBase 的 SQL 使用手册。
QianBase 命令行工具指南	本指南适用于维护和监管 QianBase 数据库的数据库管理员和支持人员。

1. QianBase 数据库

QianBase 基于 Apache 顶级项目 Trafodion，提供企业级运营和事务型的 SQL on Hadoop 解决方案，并具备 Trafodion 的所有功能。

本章讲述以下内容:

1.1 QianBase 架构

QianBase 结构上分为三层：客户端、SQL 数据库服务层、存储引擎层。其中存储引擎层是使用 HBase，保证数据库与 Hadoop 生态圈有良好融合的基础。

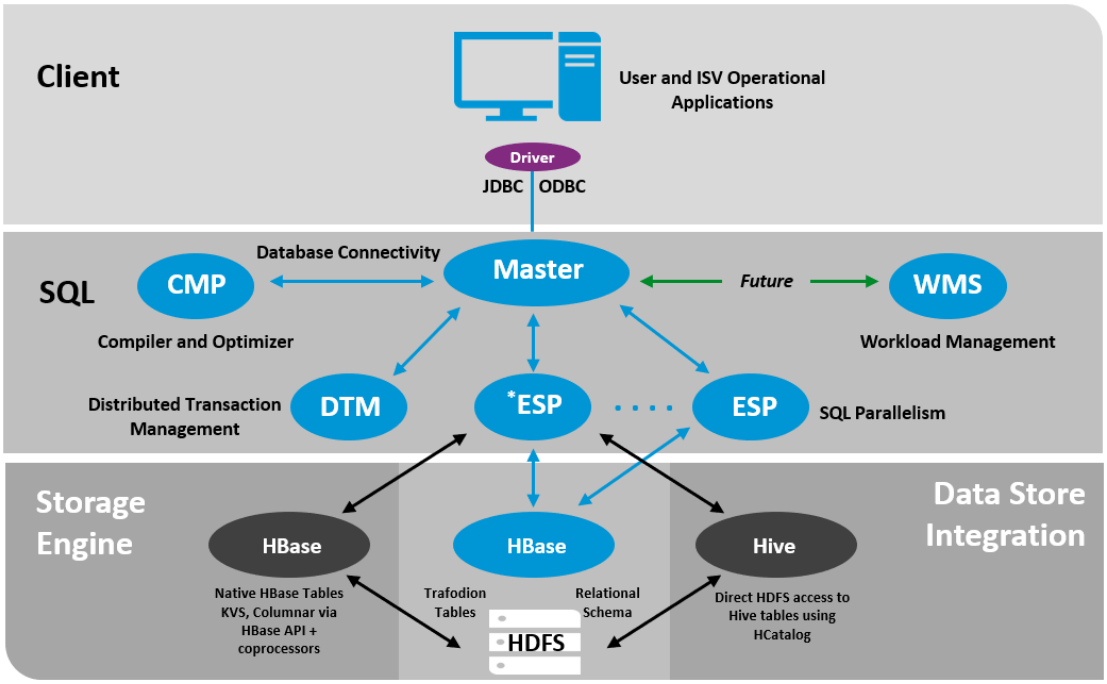

图 1-1 QianBase 架构

第一层是**客户端服务层**（Client Services Layer），客户业务应用程序位于该层。操作应用程序可以是客户编写的，也可以通过第三方 ISV 工具/应用程序启用。使用 QianBase 提供的 Windows 或 Linux 客户端驱动程序，通过标准 JDBC/ODBC 接口完成对 QianBase 数据库服务层的访问。对响应时间、连接数量和安全的不同需求以及其他因素，您可以使用 JDBC Type2 或 Type4 驱动程序。此外，您还能使用 ADO.NET 驱动程序。

第二层是 **SQL 查询引擎层**（SQL Query Engine Layer），包括 QianBase 数据库服务。

该层封装了所有管理 QianBase 数据库对象和执行 SQL 请求的服务，包括连接管理、SQL 语句编译和生成优化的执行计划、生成 SQL 执行（并行和非并行）计划、事务管理和工作负载管理。QianBase 提供透明并行 SQL 执行，从而减少复杂 MapReduce 编程开发。

第三层是**存储引擎层**(Storage Engine Layer)。

该层由QianBase使用的标准Hadoop服务组成,包括HBase、HDFS和Zookeeper。

QianBase 数据库对象可以存储在原生 Apache Hadoop 数据库结构中，包括以下格式:

- Apache HBase，提供 Big Table、宽列建值和数据模型。
- 文本文件，用于暂存数据（例如，逗号分隔值或日志数据)。
- 键值序列文件。

QianBase 支持以上格式的表作为外部表使用。

QianBase 能透明地将应用程序的 SQL 请求映射至不同存储格式所需的原生请求，它架构在 HBase 上，提供关系型 schema 抽象。QianBase 使用熟悉的 DDL/DML 语法（包括对象命名，列定义和数据类型支持等）支持传统的关系型数据库对象（schema、表、视图、二级索引、存储过程）与原生 Apache Hive 和 HBase 数据存储集成。

QianBase 拥有一个更强大的功能—使用原生存储引擎和数据格式，支持和访问存储在原生 Apache Hive 和 HBase 表中的数据，并具有以下优势:

- 对原生 Apache HBase 表执行查询，无需将其复制至 Apache Trafodion 表结构中。
- 优化访问 Apache HBase 表，无需 MapReduce 过程。
- 生成 Apache HBase 的统计信息，创建优化的查询计划。
- 支持同时连接到不同的数据源，支持的数据源有：Apache Hive 和 HBase。
- 使用 Apache HBase 原生 schema 的灵活性。
- 支持从分区 Hive 表中执行 select 和 insert 操作。insert 使用标准 SQL 语法。

1.2 QianBase 组件与进程

QianBase 安装程序包含组件:

组件名	**说明**
Monitor	监控 QianBase 进程的状态。

DTM	为 QianBase 提供分布式事务服务。
SQL	提供数据库引擎，由以下组成: 1、SQL 编译器，处理复杂查询或 DDL 操作。 2、SQL 运行环境，提供执行查询的环境。
DCS	为 QianBase 提供连接服务。
REST 服务器	发布 QianBase 关键组件、系统和查询指标的健康状况。
Esgyn DBManager	提供管理 QianBase 的服务，包括监控和报警等。

QianBase 安装程序包含开源组件:

组件名	**说明**
ElasticSearch	一个基于 Lucene 的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎
Logstash	集中、转换和存储数据 logstash 是开源的服务器端数据处理管道,能够同时从多个来源采集数据,转换数据,然后将数据发送到 ElasticSearch 中
Redis	一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库
Grafana	Grafana 是一个可视化面板(Dashboard),有着非常漂亮的图表和布局展示,功能齐全的度量仪表盘和图形编辑器。我们利用他的仪表板做图表展示。
Prometheus	Prometheus 由 Go 语言编写而成,采用 Pull 方式获取监控信息,并提供了多维度的数据模型和灵活的查询接口。
Alert Manager	Alert Manager 是 Prometheus 组件的一部分，Prometheus 专门将告警通知功能剥离出来，可用来管理告警通知，可对告警进行静默、分组、抑制等操作，功能丰富。并且通过 yaml 进行配置，可以方便的对接到 DB Manager 管理平台

	中。
Keepalived	用于高可用配置的一个 Linux 路由软件，在集群之间配置好之后,可以在集群中某个节点故障后自动迁移虚拟 IP 到另一台正常可用的节点。

QianBase 安装程序包含以下自组研发组件:

组件名	**说明**
elasticsearch adapter	Prometheus 为了保证功能的单一性，只设计了本地存储。如果需要将监控数据持久化保存到第三方存储中，需要有 adapter 的支持来实现数据的读写。我们在开源 adapter 的基础上自研了基于 ElasticSearch 的 adapter，利用 ElasticSearch 的 HA 特性实现了 Prometheus 数据的持久化存储。
DBM	为了实现 DB Manager 服务的高可用，我们将 DB Manager 的前后端服务分离， DBM 是 DB Manager 的后台服务。目前 DBM 实现以下两个功能: 1. 转发依赖数据库环境本身的请求到集群端的 DB Manager 后台服务 2. 实现日志，告警，SQL 审计等不依赖于数据库环境的后台服务
DBM-web	为了实现 DB Manager 服务的高可用，我们将 DB Manager 的前后端服务分离，DBM-web 是 DB Manager 的 web 端服务。使用 nginx 做反向代理，配置了多副本，并用虚拟 IP 绑定保证前端服务高可用。

QianBase 分布式数据库启动后包括如下进程：

进程名	别名	说明
DTM		为 QianBase 提供分布式事务服务。
RMS		为 QianBase 提供运行时管理服务。
DcsMaster	DCS Master	进程负责监控集群中所有的服务器实例，它将 JDBC/ODBC/ADO.NET 客户端的连接请求分配给 Master Executor。
DcsServer	DCS Server	进程负责启动和执行 Master Executor 进程。
MXOSRVER	Master Executor	进程是 CMP 和 ESP 包括其中。
RestServer		发布 QianBase 关键组件、系统和查询指标的健康状况。
StrawscanDispatcher		
DBManager	Esgyn DBManager	为 OM 组件中的 DB Manager 提供数据，不再直接提供网页服务。

OM 启动后在 QianBase 节点上包括如下 OM client 端进程：

进程名	别名	说明
Filebeat		用于 log 日志采集。
Esgyn_exporter		用于 QianBase 相关指标收集。
Node_exporter		用于系统相关指标收集。

1.3 服务处理流程

QianBase 在接受到处理请求后，会进行如图下列操作流程：

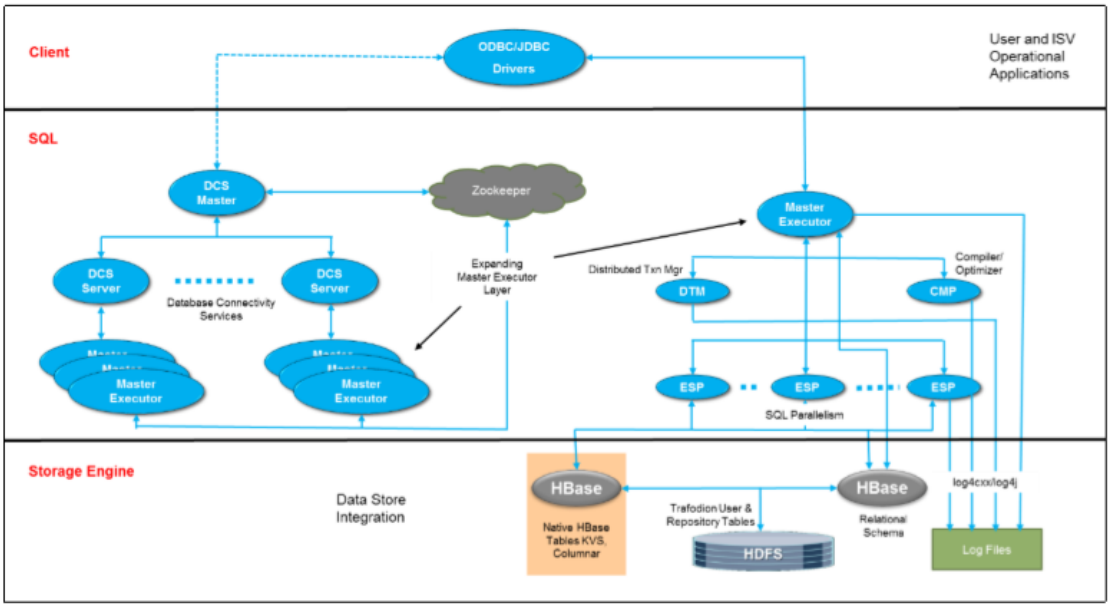

图 2-2 QianBase 服务处理流程

- 流程从客户业务应用程序或第三方 SQL 客户端工具开始。NT 或 Linux 客户端业务应用程序通过 ODBC/JDBC/ADO.NET 驱动程序访问 QianBase。
- 当客户端业务程序打开连接时，QianBase 的 DCS Master 进程会处理请求，再将连接分配至 Master Executor 进程。为了保证服务的负载均衡和高可用，DCS Server 进程失败或异常退出时支持无缝迁移，这部功能 QianBase 使用了 Apache Zookeeper。
- 客户端业务应用程序提交的 SQL 语句会依据 DCS Master 根据之前的分配信息，将 SQL 高效的发送到 Master Executor。
- Master Executor 接受到情况后，调用编译器和优化器解析和编译 SQL 语句，生成最优执行计划。
- 如果并行执行优化计划，则 Master Executor 将任务分配到多 ESP 并行计算，ESP 最终与 Apache HBase 数据交互操作。ESP 的执行结果被传回至 Master Executor 进行合并。
- 为了实现分布式 ACID 事务保护，QianBase 通过 DTM 进程实现。Master Executer 通过调用 DTM 服务完成分布式事务管理，DTM 内部是通过调用 QianBase 提供的 HBase 协处理器服务实现事务管理。

2. 数据库安装与启动

QianBase 的存储引擎是选用的 HBase，那么在安装 QianBase 核心组件前，先需要保证 HBase 集群已经就绪。QianBase 支持 Ambari、Cloudera 和 Apache 等标准版本 HBase 进行 QianBase 的核心组件安装。

2.1 数据库安装与卸载

具体操作可以查阅《QianBase 安装部署指南》。

在安装前，请确认 HBase 所选用的供应商和对应的版本，通过版本信息查找对应的安装章节说明。

2.2 数据库启停管理

管理 QianBase 需要使用 QianBase 用户身份登陆。

以下是管理各个服务的脚本：

组件	启动	停止	状态检查
QianBase Core	sqstart	sqstop	sqcheck
RMS Server	rmsstart	rmsstop	rmscheck
REST Server	reststart	reststop	restcheck
DCS Server	dcsstart	dcstop	dcscheck

3. 数据库状态监控

QianBase 提供 DBManager 监控并管理 QianBase 和工作负载状况。方便管理员时刻查看数据库的运行状况，包含如下功能：

- 仪表盘

  监控 QianBase 每个节点的服务运行状况，每个监控指标按时间序列进行显示。

- 工作负载

  监控系统中活跃的实时查询，以及每个查询语句在执行过程的资源使用情况。

- 数据库

  可以查询当前所有的元数据信息：Schema、表、视图、索引等。

- 日志

  QianBase 运行日志。

- 连接

  查看 QianBase 连接服务和 Master Executor 的状态，以及当前活跃的应用会话

- 查询工作台

  提供交互方式，执行 SQL，并支持显示执行计划和执行计划导出功能，协助远程分析。

- 报警

具体可以查阅《Esgyn DBManager 用户手册》。

以下为仪表盘截图，可以看出管理员可以实时看到数据库的运行状况：

4. 安全管理

QianBase 中存储大量的数据，保护数据不受内部和外部的侵害是数据库管理的重要组成部分。QianBase 系统提供了一整套保护数据安全的机制，包括角色、用户、用户组方法和手段。

具体操作可以查阅《Esgyn DBManager 用户指南》的第十一章。

Esgyn DBManager 提供通过窗体交互模式，完成所有操作。您无需记忆任何命令，通过窗体的提示，一步步完成工作即可。

5. 管理 Schema

注意

本章内容来源于《QianBase SQL 用户手册》，如果在阅读中发现本章内容与《QianBase SQL 用户手册》存在差异，请优先参考《QianBase SQL 用户手册》。

本章讲述以下内容:

5.1 创建 Schema

从数据库中创建 Schema。

5.1.1 Create Schema 语句

具体语法如下:

```
CREATE [schema-class] SCHEMA schema-clause
schema-class is: [ PRIVATE | SHARED ]
schema-clause is: { schema-name [AUTHORIZATION authid] |
AUTHORIZATION authid }
```

5.1.2 Create Schema 语法

- schema-class

  标明默认情况下是否只有授权 ID 才可访问 schema（PRIVATE)、以及是否任何数据库用户都可以将对象添加到 schema（SHARED)。默认类为 PRIVATE（私有类)。

- schema-name

  这既是新的 schema 名称，也是一个 SQL 标识符，它指定一个并非当前 schema 名称的唯一名称。此参数是可选的。然而，如果您未指定 schema 名称，则必须指定授权子句。如果没有提供 schema 名称，则使用授权 ID 作为 schema 名称。如果授权 ID 名称匹配现有的 schema，则 CREATE SCHEMA 命令将会运行失败。

- authid

  此为拥有和管理 schema 的数据库用户或角色的名称。如果该子句不存在，则当前用户将成为 schema 所有者。

5.1.3 Create Schema 注意事项

5.1.3.1 保留的 schema 名称

以前导下划线(_)开头的 schema 名称将被保留，以供将来使用。

5.1.3.2 AUTHORIZATION 子句

AUTHORIZATION 子句是可选的。如果您省略该子句，则当前用户将成为 schema 所有者。

> **注意**
>
> 在这种情况下，如果您在私有 schema 中创建函数，您必须是该 schema 的所有者。

schema 所有者可以执行 schema 的操作以及 schema 内部对象的操作。例如：

- 改变对象的 DDL。
- 删除 schema。
- 删除对象。
- 使用诸如 UPDATE STATISTICS 和 PURGEDATA 之类的实用工具命令管理对象。

5.1.3.3 谁可以创建 schema?

创建schema的权限受控于SQL_OPERATIONS组件的权限CREATE_SCHEMA。默认情况下，该权限将被授予 PUBLIC、但也可被 DB ROOT（数据库根用户）撤销。

当授权被初始化后，所有授权 ID 都将被授予 CREATE_SCHEMA 权限：

- PUBLIC
- DB__ROOT
- DB__ROOTROLE

DB__ROOT 或被授予 DB__ROOTROLE 角色的任何人都可以授予 CREATE SCHEMA 权限。

5.1.4 Create Schema 例子

该例子创建一个名为 MYSCHEMA、且归属当前用户的私有 schema：

```
CREATE SCHEMA myschema;
```

该例子创建了一个共享 schema，并指定 CliffG 作为所有者：

```
CREATE SHARED SCHEMA hockey_league AUTHORIZATION
"CliffG";
```

该例子创建了一个私有 schema，并指定 DBA 角色作为 schema 所有者：

```
CREATE PRIVATE SCHEMA contracts AUTHORIZATION DBA;
```

被授予 DBA 角色的用户可以授权其他用户和角色访问 CONTRACTSschema 里的对象。

该例子创建了一个名为 JSMITH 的 schema：

```
CREATE PRIVATE SCHEMA AUTHORIZATION JSmith;
```

5.2 删除 Schema

从数据库中删除 Schema。

5.2.1 Drop Schema 语句

具体语法如下:

```
DROP SCHEMA [IF EXISTS] schema-name [RESTRICT | CASCADE]
```

5.2.2 Drop Schema 语法

- schema-name

这既是新的 schema 名称，也是一个 SQL 标识符，它指定一个并非当前 schema 名称的唯一名称。此参数是可选的。然而，如果您未指定 schema 名称，则必须指定授权子句。如果没有提供 schema 名称，则使用授权 ID 作为 schema 名称。如果授权 ID 名称匹配现有的 schema,则 CREATE SCHEMA 命令将会运行失败。

- IF EXISTS

删除存在的 Schema，如果不存对应的 Schema，将抛出异常信息。

- RESTRICT

如果指定RESTRICT,则在指定的Schema 不为空时报告错误。默认值:RESTRICT。

- CASCADE

如果指定 CASCADE，则会删除指定 Schema 中的对象和 Schema 本身。其他 Schema 中依赖于此 Schema 中的对象的任何对象也将被删除。

5.2.3 Drop Schema 注意事项

5.2.3.1 AUTHORIZATION 子句

要删除 schema，必须满足以下条件之一:

- 您是 schema 的所有者。
- 您已被授予拥有 schema 的角色。
- 您已被授予 DROP_SCHEMA 权限。

5.2.4 Drop Schema 例子

该例子演示删除一个空的 Schema。

```
DROP SCHEMA sales;
```

6. 管理表

注意:

本章内容来源于《QianBase SQL 用户手册》，如果在阅读中发现本章内容与《QianBase SQL 用户手册》存在差异，请优先参考《QianBase SQL 用户手册》。

本章讲述以下内容:

6.1 创建表

6.1.1 Create Table 语句

具体语法如下：

```
CREATE [VOLATILE] TABLE [IF NOT EXISTS] table
{ table-spec | like-spec }
[STORE BY {PRIMARY KEY | (key-column-list)}]
[SALT USING num PARTITIONS [IN num REGIONS]
[ON (column[, column]...)]]
[RANGE SPLIT BY (column [ASC|DESC] [, column
[ASC|DESC] ...]) split-list]
[HBASE_OPTIONS (hbase-options-list)]
[LOAD IF EXISTS | NO LOAD]
[AS select-query]
[ATTRIBUTES (synchronous | no) replication]
[ATTRIBUTE NAMESPACE namespace-string]
table-spec is:
 (table-element [,table-element]...)
table-element is:
column-definition | [CONSTRAINT constraint-name] table-
constraint
column-definition is:
column data-type [DEFAULT default | NO DEFAULT]
[[CONSTRAINT constraint-name] column-constraint]...
data-type is:
CHAR[ACTER] [(length [CHARACTERS])]
          [CHARACTER SET char-set-name]
          [UPSHIFT] [[NOT]CASESPECIFIC]
```

```
| CHAR[ACTER] VARYING (length [CHARACTERS])
          [CHARACTER SET char-set-name]
          [UPSHIFT] [[NOT]CASESPECIFIC]
| VARCHAR (length) [CHARACTER SET char-set-name]
          [UPSHIFT] [[NOT]CASESPECIFIC]
| VARCHAR2 (length) [CHARACTER SET char-set-name]
          [UPSHIFT] [[NOT]CASESPECIFIC]
| NCHAR (length) [CHARACTERS]
          [UPSHIFT] [[NOT]CASESPECIFIC]
| NCHAR VARYING(length [CHARACTERS])
          [UPSHIFT] [NOT] CASESPECIFIC]
| NUMERIC [(precision [,scale])] [SIGNED|UNSIGNED]
| SMALLINT [SIGNED|UNSIGNED]
| INT[EGER] [SIGNED|UNSIGNED]
| LARGEINT
| DEC[IMAL] [(precision [,scale])] [SIGNED|UNSIGNED]
| FLOAT [(precision)]
| REAL
| DOUBLE PRECISION
| DATE
| TIME [(time-precision)]
| TIMESTAMP [(timestamp-precision)]
| INTERVAL { start-field TO end-field | single-field }
default is:
literal
| NULL
| CURRENT_DATE
| CURRENT_TIME
```

```
| CURRENT_TIMESTAMP
column-constraint is:
NOT NULL
| UNIQUE
| PRIMARY KEY [ASC[ENDING]
| DESC[ENDING]]
| CHECK (condition)
| REFERENCES ref-spec
table-constraint is:
UNIQUE (column-list)
| PRIMARY KEY (key-column-list)
| CHECK (condition)
| FOREIGN KEY (column-list) REFERENCES ref-spec
ref-spec is:
referenced-table [(column-list)]
column-list is:
column-name [,column-name]...
key-column-list is:
column-name [ASC[ENDING] | DESC[ENDING]] [,column-name
[ASC[ENDING] | DESC[ENDING]]]...
like-spec is:
LIKE source-table [include-option]
split-list is:
( split-list-element [, split-list-element] ... )
split-list-element is:
ADD FIRST KEY ( literal [, literal] ... )
hbase-options-list is:
hbase-option = 'value'[, hbase-option = 'value']...
```

6.1.2 Create Table 语法

- VOLATILE

  指定一个可变表，该表仅存在于创建表会话期间。会话结束时，该表将被自动删除。

- IF NOT EXISTS

  如果创建表时 HBase 表不存在，那么创建一个 HBase 表。该选项不适用于可变表。

- table

  指定一个表的 ANSI 逻辑名称。该名称在其所属的 schema 内部的表和视图名称里必须是唯一的。

- STORE BY { PRIMARY KEY | (key-column-list)}

  指定聚集键所基于的一组列。聚集键确定保存该表的物理文件内部的行序。存储顺序可以影响您的对象分区方案。

  - PRIMARY KEY：聚集键基于主键列而定。
  - key-column-list：聚集键基于 key-column-list 里的列而定。 key-column-list 里的键列必须被指定为 NOT NULL（非空），并且必须 与表上定义的主键列相同。 如果没有指定 STORE BY，则聚集键将成为主键。

- SALT USING num PARTITIONS [IN num REGIONS] [ON (column[, column]...)]

  在创建表时，将其预分割为多个区域。“加盐”技术会添加行键的哈希值作为键前缀，以防出现序列键“热点”。如果您未指定单独的分区和区域编号，则将为每个分区创建一个 HBase 区域。您指定的分区数量可以是目前 HBase 集群里的区域服务器数量的函数。如果您使用更少量的区域，则未来还有增长空间。您指定的分区数量从 2 到 1024 不等，区域数量可以从 2 到加盐分区数量不等。如果您未指定列，则默认使用所有主键列。

> **注意**:
> 如果您指定 SALT 选项，QianBase 会创建一个带有 ConstantSizeRegionSplitPolicy 的表。

您可以使用以下 HBASE_OPTIONS 覆盖此设置。

- SPLIT BY (column [, column]...]) split-list
  在创建一个表时将其预分隔为多个 HBase 区域。该选项不可与 SALT 同时使用。指定的列必须带有 PRIMARY KEY 或 STORE BY key 前缀。如果您使用 DIVISION BY 子句，则首列必须为“_DIVISION_1_”。
  如果您希望显式地指定 HBase 区域边界、而非由 HBase 自动分割区域，则请使用该选项。

> **注意**:
> 如果您指定 SALT 选项，QianBase 会创建一个带有 ConstantSizeRegionSplitPolicy 的表。

  您可以使用以下 HBASE_OPTIONS 覆盖此设置:
  - (split-list-element [, split-list-element]...) 这是 HBase 区域的分界列表，它按照逻辑顺序排列（ascending 代表升序键列，descending 代表降序键列）。如果指定 n 个 split-list-elements（拆分列表元素），则已创建的 HBase 表将具有（n+1）个区域。第一个区域开始于最低级别的键，此后的 n 个区域都具有 n 个拆分列表元素指定的起始键。
  - (literal [, literal]...)指定一个分割边界的列值。字面值对应于 SPLIT BY 里指定的列，并必须 具有相应的数据类型。
- HBASE_OPTIONS (hbase-option = 'value'[, hbase-option = 'value']...) 为该表设置

的 HBase 选项列表 hbase - option = 'value' 是此类 HBase 选项之一，其的分配数值如下（粗体字为默认值）：

BLOCKCACHE	**'true'** | 'false'
BLOCKSIZE	**'65536'** | 'positive-integer'
BLOOMFILTER	'NONE' | **'ROW'** | 'ROWCOL'
CACHE_BLOOMS_ON_WRITE	**'true'** | 'false'
CACHE_DATA_ON_WRITE	**'true'** | 'false'
CACHE_INDEXES_ON_WRITE	**'true'** | 'false'
COMPACT	**'true'** | 'false'
COMPACT_COMPRESSION	'GZ' | 'LZ4' | 'LZO' | **'NONE'** | 'SNAPPY'
COMPRESSION	'GZ' | 'LZ4' | 'LZO' | **'NONE'** | 'SNAPPY'
DATA_BLOCK_ENCODING	'DIFF' | 'FAST_DIFF' | **'NONE'** | 'PREFIX'
DURABILITY	'USE_DEFAULT' | **'SKIP_WAL'** | 'ASYNC_WAL' | 'SYNC_WAL' | 'FSYNC_WAL'
EVICT_BLOCKS_ON_CLOSE	'true' | **'false'**
IN_MEMORY	'true' | **'false'**
KEEP_DELETED_CELLS	'true' | **'false'**
MAX_FILESIZE	'positive-integer'
MAX_VERSIONS	**'1'** | 'positive-integer'
MEMSTORE_FLUSH_SIZE	'positive-integer'
MIN_VERSIONS	**'0'** | 'positive-integer'
PREFIX_LENGTH_KEY	'positive-integer'应小于表键的最大长度，只有当 SPLIT_POLICY 为 KeyPrefixRegionSplitPolicy 时才 适用。
REPLICATION_SCOPE	**'0'** | '1'
SPLIT_POLICY	**'org.apache.hadoop.hbase.regionserver.ConstantSizeRegionSplitPolicy'** | 'org.apache.hadoop.hbase.regionserver.IncreasingToUpperBoundRegionSplitPolicy' | 'org.apache.hadoop.hbase.regionserver.KeyPrefixRegionSplitPolicy'
TTL	**'-1'** (forever) | 'positive-integer'

- LOAD IF EXISTS

  将数据装载到一个现有的表里。必须与 AS select 查询同时使用。

- NO LOAD

  使用 CREATE TABLE AS 语句创建一个表，但不会将数据装载到该表里。

- AS select-query

  指定一条 select 查询语句，用于填充所创建的表格。一条 select 查询可以是任何 SQL select 语句。

- column data-type

  指定表中列的名称和数据类型。CREATE TABLE 语句里必须至少包含一个列的定义。

  - column

    是一种 SQL 标识符。该列在表的列名里必须是唯一的。如果列名是 Trafodion SQL 保留字，则您必须将其用双引号括起来，以进行分隔。此类分隔部分须区分大小写。例如："join".

  - data-type

    是列里可存储数值的数据类型。默认值必须与列具有相同的类型，其中包含字符列的字符集。更多信息，请参阅数据类型。 数据类型还包括特定大小写格式的信息，例如 UPSHIFT。

- [NOT] CASESPECIFIC

  指定该列包含非特定大小写格式的字符串。默认值为 CASESPECIFIC。只有当两个数值都不区分大小写时,才能以不区分大小写的方式对二者进行比较。这种做法适用于二元谓词、LIKE 谓词、以及 POSITION/REPLAC 字符串函数搜索的数值比较。

- DEFAULT default | NO DEFAULT

  指定一个列的默认值或指定该列不具有默认值。

- CONSTRAINT constraint-name

  指定一个列名或表约束名。constraint-name 必须与 table 具有相同的 schema，且其在 schema 中的约束名必须是唯一的。

  如果省略在 constraint-name 里指定名称的 schema 部分，则 Trafodion SQL 将

使用表的 schema 以扩展该约束名。

- NOT NULL

  这是一个列约束，它指定该列不能包含空值。

  如果您省略 NOT NULL，则该列中可以出现空值。如果您同时指定 NOT NULL 和 NO DEFAULT，则插入表里的每行都必须包含一个该列的值。

- UNIQUE 或 UNIQUE (column-list)

  分别为一种列约束或表约束，它指定该列或多列里不得多次出现同一个数值或同一组数值。如果省略 UNIQUE，则允许出现重复数值，除非该列是 PRIMARY KEY 的一部分。

  column-list 里不得多次出现相同的列。此外，在 UNIQUE 约束上指定的一组列不得匹配为表的主键约束或任何其他 UNIQUE 约束指定的一组列。所有被定义为 UNIQUE 的列都必须被指定为 NOT NULL。唯一索引强制实施 UNIQUE 约束。如果 column-list 里已有一个唯一索引，则 Trafodion SQL 将使用该索引。如果不存在唯一索引，则系统会创建一个此类索引。

- PRIMARY KEY [ASC[ENDING] | DESC[ENDING]], or, PRIMARY KEY (key-column-list)

  分别是一个列约束或表约束，它指定一列或多列作为表的主键。keycolumn-list 里不得多次出现同一列。

  ASCENDING 和 DESCENDING 指定键内一列的输入方向。默认值为 ASCENDING。

  表内各行的 PRIMARY KEY 值必须是唯一的。为一组列定义的 PRIMARY KEY 表明列值是唯一且不为空的。在任何 CREATE TABLE 语句上只能指定一次 PRIMARYKEY。Trafodion SQL 使用主键作为表的聚集键，以防创建一个独立唯一的索引以实施主键约束。

  PRIMARY KEY 在 Trafodion SQL 里是必需的。

- CHECK (condition)

这是一种约束，它指定了表里各行都必须满足的条件。

您既无法在 CHECK 约束里引用 CURRENT_DATE、CURRENT_TIME 或 CURRENT_TIMESTAMP，也无法在 CHECK 约束里使用子查询。

- REFERENCES ref-spec

  指定一种 REFERENCES 列约束，采用 REFERENCES 约束的各列的最大组合长度为 2048 字节。

  ref-spec 是:

  referenced-table [(column-list)]

  referenced-table 是引用约束里外键所引用的表。

  referenced-table 既不能是一个视图，也不能与表相同。

  referenced-table 对应于表的外键。

  column-list 指定 referenced-table 的一列或多列对应于该表的外键。与 REFERENCES 相关联的列表里的列、以及与 FOREIGN KEY 相关联的列表里的列必须具有相同顺序。

  如果省略 column-list，则被引用表的 PRIMARY KEY 列将是被引用的列。一个表可以有无数的引用约束，您也可以在多个引用约束里指定相同的外键， 但您必须单独定义每个引用约束。您无法创建自引用的外键约束。

- FOREIGN KEY (column-list) REFERENCES ref-spec

  这是一种表约束，它指定表的引用约束，并声明表里的一列或多列（外键）仅包含那些匹配 REFERENCES 子句指定表的一列或多列的数值。两列或一组列必须具备相同的特性（数据类型、长度、尺度、精度)。如果不存在 FOREIGN KEY 子句，则 table 的外键将是已定义的列；如果存在 FOREIGN KEY 子句，则外键将是 FOREIGN KEY 子句里指定的一列或多列。

- LIKE source-table [include-option]...

  指导 Trafodion SQL 创建一个与现有表和源表类似的表，并省略约束（NOT NULL 和 PRIMARY KEY 约束除外）和分区，除非已指定 include-option 子句。

- source-table

  这是现有表的 ANSI 逻辑名称，必须与其 schema 中的表名和视图名不同。

- include-option

  WITH CONSTRAINTS 指导 Trafodion SQL 使用来自 source-table 的约束。表的约束名是随机生成的唯一名称。

  执行 CREATE TABLE LIKE 语句时，无论是否包含 WITH CONSTRAINTS 子句，目标表都将具有源表上存在的所有 NOT NULL 列约束，但会采用不同的约束名。

- WITH PARTITIONS

  指导 Trafodion SQL 使用来自 source-table 的约束。每个新表分区和其原有 source-table 副本一样，占据相同的空间。新表分区不会沿用原有表的分区名称，Trafodion SQL 会根据物理文件位置生成新的名称。 如果指定 LIKE 从句和 SALT USING num PARITIONS 从句，则不能指定 WITH PARTITIONS 从句。

- ATTRIBUTES(synchronous | no) replication]

  启用表的同步复制或禁用表的复制。该属性可以与高级版 QianBase 的跨数据中心复制功能结合使用。如果该表包含索引，则这些索引也将被自动复制。默认情况下不会复制表。

6.1.3 Create Table 注意事项

6.1.3.1 必需的权限

发出一条 CREATE TABLE 语句，必须满足以下条件之一：

- 您是 DB__ROOT(数据库根用户)。
- 您正在一个共享 schema 中的创建表。
- 您是私有 schema 的所有者。

- 您拥有 SQL_OPERATIONS 组件的 CREATE 或 CREATE_TABLE 权限。

> **注意**:
>
> 在这种情况下，如果您在私有 schema 中创建函数，您必须是该 schema 的所有者。

6.1.3.2 创建引用完整性约束所需权限

创建一个引用完整性约束（一个表约束，它引用另一个表的某列），必须满足下列条件之一:

- 您是 DB__ROOT（数据库根用户）。
- 您是引用表和被引用表的所有者。
- 您对于引用表和被引用表拥有以下权限:
  - 对于引用表而言，您拥有 SQL_OPERATIONS 组件的 CREATE 或 CREATE_TABLE 权限。
  - 对于被引用表而言，您可以通过用户名或授予的角色拥有该表的 REFERENCES（或 ALL）权限。

如果该约束在查询表达式里引用另一个表，那么您必须拥有此表的 SELECT 权限。

6.1.3.3 CREATE VOLATILE TABLE 注意事项

- 临时可变表与会话紧密相联。它们的命名空间在多个并发会话中是唯一的，因此允许多个会话同时使用相同的临时可变表名，且不会造成任何冲突。
- 可变表支持创建索引。
- 可变表由系统进行分区。默认情况下分区数量限制为四个，这些分区将跨集群分布。不管系统配置如何，默认值都是四个分区。
- 可变表的统计信息不会自动更新。如果需要统计信息，您必须显式地运行 UPDATE STATISTICS 语句。

- 可以使用一部分、两部分或三部分名称创建和访问可变表。但是，必须为可变表上创建的任何 DDL 或 DML 语句使用相同的（一部分、两部分或三部分）名称。
- Trafodion SQL 允许用户在包含空值的列上显式地指定主键和 STORE BY 子句.
- Trafodion SQL 并不要求可变表的首列必须包含非空值、且必须为主键。相反，Trafodion SQL 会尝试对表进行分区；尽量使用合适的键列作为主键和分区键。

6.1.3.4 CREATE VOLATILE TABLE 使用限制

- 可变表不支持以下情况:
  - ALTER 语句
  - User 约束
  - 创建视图
  - 在一个可变表上创建非可变索引或在一个非可变表上创建可变索引
  - CREATE TABLE LIKE 语句操作

- Trafodion SQL 如何支持可变表的可空键?
  - 允许 PRIMARY KEY、STORE BY 和 UNIQUE 约束里出现可空键
  - 空值被视为该列的最高值
  - 一个空值等同于其他空值，该列仅允许出现一个值
  - Trafodion SQL 如何为可变表选择合适的键?
  - Trafodion SQL 在创建的表的列中搜索首个合适的列。一旦发现合适的列，将基于此列对该表进行分区。表里搜素到的列可以（在 CREATE TABLE 语句里）显式地指定或（在 CREATE TABLE AS SELECT 语句里）隐式地创建。
  - 只有在语句里未指定 PRIMARY KEY 或 STORE BY 子句时，才可以选择合适的键列。如果已经指定任何此类子句，则可以使用它们选择键列。
  - Trafodion SQL 遵循这些准则搜索和选择合适的键:

- 合适的列可以是一个可空列。
- Trafodion SQL 里的某些数据类型不可被用作一个分区键。目前，这包括任何浮点列（REAL、DOUBLE PRECISION 和 FLOAT）。
- Trafodion SQL 可以基于预定义的顺序搜索一个合适的列：
  - 应首先选择 Numeric 列、进而选择固定的 CHAR、DATETIME、INTERVAL 和 VARCHAR 数据类型。
  - Numeric 数据类型内部的选择顺序为二进制 NUMERIC（LARGEINT、INTEGER、SMALLINT）和 DECIMAL。
  - 无符号列的优先级高于有符号列。
  - 非空列的优先级高于可空列。
  - 如果所有数据类型都相同，则应选择首列。

- 如果未发现合适的列，则可变表将变为一个非分区表，系统定义的 SYSKEY 将作为其主键；如果已发现一个合适的列，则该列将成为分区键，其中主键为 suitable_column、SYSKEY。这将导致该表进行分区，同时能防止出现重复键和空到非空的错误。

当 Trafodion SQL 搜索一个合适的键时，Table 1 显示从低到高的数据类型优先顺序。后来出现的数据类型的优先级高于先前出现的数据类型的优先级。未在 Table 里出现的数据类型不能被选作键列。

在合适的键搜索期间内排列数据类型的优先级
VARCHAR
INTERVAL
DATETIME
CHAR(ACTER)
DECIMAL (signed, unsigned)
SMALLINT (signed, unsigned)
INTEGER (signed, unsigned)

LARGEINT (signed only)

6.1.3.5 在一个可变表里创建可空约束

以下例子为如何在一个可变表里创建可空约束（primary key、STORE BY 和 unique）:

```
create volatile table t (a int, primary key(a));
create volatile table t (a int, store by primary key);
create volatile table t (a int unique);
```

6.1.3.6 创建一个带有可空主键的可变表

以下例子创建了一个带有可空主键的可变表:

```
>>create volatile table t (a int, primary key(a));
--- SQL operation complete.仅允许出现唯一空值:
```

仅允许出现唯一空值:

```
>>insert into t values (null);
--- 1 row(s) inserted.

>>insert into t values (null);
*** ERROR[8102] The operation is prevented by a unique
constraint.
--- 0 row(s) inserted.
```

6.1.3.7 为可变表搜索合适的键的例子

以下例子为 Trafodion SQL 选择合适的键时所遵循的顺序，这些优先级规则在“Trafodion SQL 如何为可变表搜索合适的键”中进行了描述。

- 选择一列作为主键和分区键：

```
create volatile table t (a int);
```

- 选择 b 列，因为 int 的优先级高于 char：

```
create volatile table t (a char(10), b int);
```

- 选择 b 列，因为非空列的优先级高于可空列。

```
create volatile table t (a int, b int not null);
```

- 选择 b 列，因为 int 的优先级高于 decimal：

```
create volatile table t (a decimal(10), b int);
```

- 选择首列，因为两个列具有相同的数据类型：

```
create volatile table t (a int not null, b int not
null);
```

- 选择 b 列，因为 char 的优先级高于 date：

```
create volatile table t (a date, b char(10));
```

- 选择 b 列，因为 real 数据类型并非待检查的列的一部分：

```
create volatile table t (a real, b date);
```

- 不得选择任何列作为主键/分区键。SYSKEY 将被自动使用。

```
create volatile table t (a real, b double precision
not null);
```

类似的例子将用于 CREATE TABLE AS SELECT 查询。

6.1.4 Create Table…Like 注意事项

CREATE TABLE LIKE 语句不能基于源表创建新表的视图、所有者信息或权限。使用 LIKE 规范创建的新表的相关权限都已被定义，仿佛该新表是由当前用户显式地创建。

6.1.4.1 CREATE TABLE ...LIKE 和文件属性

CREATE TABLE ... LIKE 语句创建一个类似于其他表（文件属性除外）的表。文件属性包括 COMPRESSION 等。

如果您未将属性值列入 CREATE TABLE ... LIKE 命令的一部分，则 SQL 创建的表具有属性默认值、而非源对象的数值。例如：为使创建的表类似于其他指定压缩率的表，您必须在 CREATE TABLE ... LIKE 语句里指定压缩率属性值。

在以下例子中，原 CREATE TABLE 语句创建一个未经压缩的表。然而，CREATE TABLE ... LIKE 语句里指定了压缩率。

```
-- Original Table
create table NPTEST (
FIRST_NAME CHAR(12) CHARACTER SET ISO88591 COLLATE
DEFAULT NO DEFAULT NOT NULL ,
LAST_NAME CHAR(24) CHARACTER SET ISO88591 COLLATE
DEFAULT NO DEFAULT NOT NULL ,
ADDRESS CHAR(128) CHARACTER SET ISO88591 COLLATE DEFAULT
DEFAULT NULL ,
ZIP INT DEFAULT 0 ,
PHONE CHAR(10) CHARACTER SET ISO88591 COLLATE DEFAULT
DEFAULT NULL ,
SSN LARGEINT NO DEFAULT NOT NULL ,
INFO1 CHAR(128) CHARACTER SET ISO88591 COLLATE DEFAULT
DEFAULT NULL ,
INFO2 CHAR(128) CHARACTER SET ISO88591 COLLATE DEFAULT
DEFAULT NULL ,
primary key (SSN,first_name,last_name)
 )   max table size
```

```
-- CREATE TABLE LIKE
 create table LSCE002 like NPTEST ATTRIBUTE compression
type hardware;
```

6.1.5 Create Table as 语句注意事项

以下注意事项适用于 CREATE TABLE AS 语句:

- 访问一个由 CREATE TABLE AS 语句构建的表时，需要进行全表扫描，因为主键和聚集键无法轻易定义。
- 不能为 CREATE TABLE AS 表生成编译时间估值和运行时信息。
- 您无法使用 WMS 编译时和运行时规则以管理 CREATE TABLE AS 表。
- 如果没有显式地定义 CREATE TABLE 语句里的所有列，则无法指定 CREATE TABLE AS 表的主键。
- 您无法为一条 CREATE TABLE AS ... INSERT/SELECT 语句生成解释计划。但如果使用 NO LOAD 选项时，您可以为一条 CREATE TABLE AS ... INSERT/SELECT 语句使用解释计划。
- 您无法在一条 CREATE TABLE AS 语句里使用 ORDER BY 子句。编译器会对 所选的行进行透明排序，以此提高插入操作的效率。

6.1.5.1 CREATE TABLE AS 语句的 LOAD IF EXISTS 和 NO LOAD 选项的注意事项

CREATE TABLE AS 语句的 LOAD IF EXISTS 选项使数据装载到一个现有的表里。如果未指定 LOAD IF EXISTS 选项、并尝试将数据装载到一个现有的表里，则 CREATE TABLE AS 语句将执行失败。

以下情况可使用带有 AS 子句的 LOAD IF EXISTS 选项:

- 在没有重新建表的情况下运行 CREATE TABLE AS 语句。该表必须为空。否则，CREATE TABLE AS 语句将返回一个错误。发出 CREATE TABLE AS 语句之前，通过使用 DELETE 语句删除表里的数据。

- 使用 CREATE TABLE AS 语句将数据逐步添加到一个现有的表里。发出 CREATE TABLE AS 语句之前，您必须启动一个用户定义的事务。如果未启动用户定义的事务而尝试执行 CREATE TABLE AS 语句，则将返回一个错误，指出该数据已经存在于表里。借助于用户定义的事务，出现错误时，新添加的行都将被回滚。

运行带有 NO LOAD 选项的 CREATE TABLE AS 语句可以创建一个表，但不会将数据装载到该表里。

如果必须创建一个表以审查其结构、并使用 EXPLAIN 语句分析 CREATE TABLE AS 语句的 SELECT 部分，则该选项是有用的。您也可以使用 EXPLAIN 语句分析 CREATE TABLE AS ...NO LOAD 语句里所涉及的 INSERT/SELECT 部分。 例如如下：

```
CREATE TABLE ttgt NO LOAD AS (SELECT ...);
```

6.1.6 Create Table 语句的 QianBase SQL 扩展语句

该语句为 ANSI SQL:1999 入门级标准所支持。CREATE TABLE 语句的 Trafodion SQL 扩展语句为 ASCENDING、DESCENDING 和 PARTITION 子句。

CREATE TABLE LIKE 语句也是一条扩展语句。

6.1.7 Create Table 例子

- 该例子创建了一个表。该表的聚集键是其主键。

```
CREATE TABLE SALES.ODETAIL (
ordernum NUMERIC (6) UNSIGNED NO DEFAULT NOT NULL,
partnum NUMERIC (4) UNSIGNED NO DEFAULT NOT NULL,
unit_price NUMERIC (8,2) NO DEFAULT NOT NULL,
qty_ordered NUMERIC (5) UNSIGNED NO DEFAULT NOT NULL,
PRIMARY KEY (ordernum, partnum)
);
```

- 该例子创建了一个“加盐”表，它所拥有的尺寸相同的区域可以从 4 个增长到 8、12 或 24 个。

```
CREATE TABLE SALES.ODETAIL_SALT (
ordernum NUMERIC (6) UNSIGNED NO DEFAULT NOT NULL,
partnum NUMERIC (4) UNSIGNED NO DEFAULT NOT NULL,
unit_price NUMERIC (8,2) NO DEFAULT NOT NULL,
qty_ordered NUMERIC (5) UNSIGNED NO DEFAULT NOT NULL
PRIMARY KEY
 (ordernum, partnum)
 ) SALT USING 24 PARTITIONS IN 4 REGIONS ON
(ordernum);
```

- 该例子创建了一个带有 STORE BY 子句、而非主键的表。该表被预分割为 4 个 HBase 区域:

```
CREATE TABLE SALES.ODETAIL_SB (
ordernum NUMERIC (6) UNSIGNED NO DEFAULT NOT NULL,
partnum NUMERIC (4) UNSIGNED NO DEFAULT NOT NULL,
unit_price NUMERIC (8,2) NO DEFAULT NOT NULL,
qty_ordered NUMERIC (5) UNSIGNED NO DEFAULT NOT NULL
) STORE BY (ordernum, partnum DESC)
SPLIT BY (ordernum, partnum)
( ADD FIRST KEY (1000, 800), ADD FIRST KEY (1000,
400), ADD FIRST KEY (2000, 800) );
```

- 该例子创建了一个表，它类似于 JOB 表、并具有相同的约束:

```
CREATE TABLE PERSNL.JOB_CORPORATE LIKE PERSNL.JOB
WITH CONSTRAINTS;
```

- 这是一个 NOT CASESPECIFIC 的用例:

```
CREATE TABLE T (a char(10) NOT CASESPECIFIC, b
```

```
char(10)); INSERT INTO T values ('a', 'A');
```

- 该例子未返回这一行。常量‘A’区分大小写，而 ‘a’列不区分大小写。

```
SELECT * FROM T WHERE a = 'A';
```

- 该例子返回了这一行。双方都区分大小写。

```
SELECT * FROM T WHERE a = 'A' (not casespecific);
```

- 该例子里返回了这一行。因为‘b’列区分大小写，所以执行大小写敏感性比较。

```
SELECT * FROM T WHERE b = 'A';
```

- 该例子里返回了这一行。因为‘b’列区分大小写，所以执行大小写敏感性 比较。

```
SELECT * FROM T WHERE b = 'A' (not casespecific);
```

6.1.8 Create Table AS 语句例子

该节显示了列属性规则，用于生成和指定创建的表的列的名称和数据类型。

如果没有指定列属性，则 select 查询里的选择列表项将被用来生成已创建表的列名和数据属性。 如果选择列表项是一列，则它将被用作已创建的列名。例如:

```
create table t as select a,b from t1 t
```

表带有两个名为（a,b）的列、以及与 t1 表里的列相同的数据属性。

- 如果选择列表项是一个表达式，则必须使用 AS 子句将其重命名。如果表达式没有命名，则将返回一个错误。例如:

```
create table t as select a+1 as c from t1
```

- t 表带有一个名为 (c) 的列和(a+1)数据属性。

```
create table t as select a+1 from t1
```

返回一个错误后，表达式必须重命名。

- 如果已指定的列属性包含数据类型信息，则它们将覆盖选择查询里的选择项的属性。这些数据属性必须与 select 查询里的选择列表项相应的数据属性相兼容。

```
create table t(a int) as select b from t1
```

t 表带有一个名为“a”的列，其数据类型为“int”。

```
create table t(a char(10)) as select a+1 b from t1;
```

这将会返回一个错误，因为列“a”的数据属性-char（字符型）并不匹配选择列表项“b”的数据属性-numeric（数字型）。

- 如果已指定的列属性仅包含列名，指定的列名将覆盖 select 查询里派生的任何名称。

```
create table t(c,d) as select a,b from t1
```

表 t 带有两个名为 c 和 d 的列，并具有表 t1 的列 a 和 b 的数据属性。

- 已指定的列属性必须包含 select 查询里的所有选择列表项相应的属性。如果存在不匹配的情况，则将返回一个错误。

```
create table t(a int) as select b,c from t1
```

返回了一个错误。这两项需要被指定作为部分表属性。

列属性必须为所有列成对指定列名和数据类型信息，或仅指定列名。您不能仅使用名称和数据类型指定一些列。

```
create table t(a int, b) as select c,d from t1
```

返回了一个错误。

- 以下例子创建了 t1 表。使用 CREATE TABLE AS 语法创建的 t2 表不含表属性。

```
CREATE TABLE t1 (c1 int not null primary key, c2
char(50));
CREATE TABLE t2 (c1 int, c2 char (50) UPSHIFT NOT
NULL) AS SELECT * FROM t1;
```

6.2 删除表

6.2.1 Drop Table 语句

该语句删除一个 Trafodion SQL 表和其依赖性对象诸如索引和约束。

```
DROP [VOLATILE] TABLE [IF EXISTS] table [RESTRICT |
CASCADE]
```

6.2.2 Drop Table 语法

- VOLATILE

  指定将要删除的表是一个可变表。

- LOAD IF EXISTS

  删除已存在的 HBase 表。该选项不适用于可变表。

- TABLE

  这是将要删除的表名。

- RESTRICT

  如果您指定 RESTRICT、且该表为另一个对象所引用，则无法删除指定的表。

  默认值为 RESTRICT。

- CASCADE

  如果您指定 CASCADE、且该表和所有引用该表的对象（诸如视图）都将被删除。

6.2.3 Drop Table 注意事项

发出一条 DROP TABLE 语句，以下条件之一必须成立：

- 您是 DB__ROOT（数据库根用户）。
- 您是该表的所有者。
- 您拥有 SQL_OPERATIONS 组件的 DROP 或 DROP_TABLE 权限。

6.2.4 Drop Table 例子

- 删除一个表:

```
DROP TABLE mysch.mytable;
```

- 删除一个可变表:

```
DROP VOLATILE TABLE vtable;
```

7. 管理索引

注意:

本章内容来源于《QianBase SQL 用户手册》，如果在阅读中发现本章内容与《QianBase SQL 用户手册》存在差异，请优先参考《QianBase SQL 用户手册》。

本章讲述以下内容:

7.1 创建索引

7.1.1 Create Index 语句

该语句基于一个表或表状对象创建一个 SQL 索引。该语句创建一个 SQL 索引，其只能存在于创建该索引的 SQL 会话期间。会话结束时，可变索引将被自动删除。更多信息，请参阅索引。CREATE INDEX 是 Trafodion SQL 扩展语句。

```
CREATE [VOLATILE] INDEX index ON table (column-name
[ASC[ENDING] | DESC[ENDING]] [,columnname [ASC[ENDING] |
DESC[ENDING]]]...) [HBASE_OPTIONS (hbase-options-list)]
[SALT LIKE TABLE]
hbase-options-list is: hbase-option = 'value'[, hbase-
option = 'value']...
```

7.1.2 Create Index 语法

- INDEX

  这是一种 SQL 标识符，它指定新索引的简易名称。您不能用该索引所属 schema 的名称来限定它。一个 schema 内的索引都有各自的命名空间，因此索引名可能与表名或约束名相同。然而，一个 schema 内的索引名必须各不相同。

- TABLE

  这是用于创建索引的表名。

- column-name [ASC[ENDING] | DESC[ENDING]] [,column-name [ASC[ENDING] | DESC[ENDING]]]...

  指定索引里包含的表列。索引列的序列与表列的序列没有对应关系。

  ASCENDING 或 DESCENDING 指定索引里的行的存储和检索顺序。默认值为 ASCENDING。

  这些索引行将根据指定的索引首列数值进行排序。如果多个索引行共享相同的首列数值，则第二列将被用来对这些索引行进行排序，依此类推。如果一个非唯一索引里出现了重复的索引行，则它们的顺序将基于底层表键列的指

定序列。为了进行排序（并非其他目的），空值应大于其他值。

- HBASE_OPTIONS (hbase-option= 'value'[,hbase-option = 'value']...)

  为该索引设置的 HBas 选项列表这些选项的应用与索引基表设置的任何 HBase 选项都无关。

- hbase-option = 'value'

  这是此类 HBase 选项之一，其分配数值如下：

BLOCKCACHE	'true' | 'false'
BLOCKSIZE	'65536' | 'positive-integer'
BLOOMFILTER	'NONE' | 'ROW' | 'ROWCOL'
CACHE_BLOOMS_ON_WRITE	'true' | 'false'
CACHE_DATA_ON_WRITE	'true' | 'false'
CACHE_INDEXES_ON_WRITE	'true' | 'false'
COMPACT	'true' | 'false'
COMPACT_COMPRESSION	'GZ' | 'LZ4' | 'LZO' | 'NONE' | 'SNAPPY'
COMPRESSION	'GZ' | 'LZ4' | 'LZO' | 'NONE' | 'SNAPPY'
DATA_BLOCK_ENCODING	'DIFF' | 'FAST_DIFF' | 'NONE' | 'PREFIX'
DURABILITY	'USE_DEFAULT' | 'SKIP_WAL' | 'ASYNC_WAL' | 'SYNC_WAL' | 'FSYNC_WAL'
EVICT_BLOCKS_ON_CLOSE	'true' | 'false'
IN_MEMORY	'true' | 'false'
KEEP_DELETED_CELLS	'true' | 'false'
MAX_FILESIZE	'positive-integer'
MAX_VERSIONS	'1' | 'positive-integer'
MEMSTORE_FLUSH_SIZE	'positive-integer'
MIN_VERSIONS	'0' | 'positive-integer'

PREFIX_LENGTH_KEY	'positive-integer'应小于表键的最大长度，只有当 SPLIT_POLICY 为 KeyPrefixRegionSplitPolicy 时才适用。
REPLICATION_SCOPE	'0' | '1'
SPLIT_POLICY	'org.apache.hadoop.hbase.regionserver.ConstantSizeR egionSplitPolicy' | 'org.apache.hadoop.hbase.regionserver.IncreasingToUp perBoundRegionSplitPolicy' | 'org.apache.hadoop.hbase.regionserver.KeyPrefixRegio nSplitPolicy'
TTL	'-1' (forever) | 'positive-integer'

- SALT LIKE TABLE
  这将导致索引使用相同的“加盐”方案（即 SALT USING num PARTITIONS [ON (column[, column]...)]）作为其基表。

7.1.3 Create Index 注意事项

7.1.3.1 创建索引步骤

在一个事务下创建索引，步骤如下:

1) 事务（用户启动的事务或系统启动的事务）开始执行。

2) 行被写入元数据中。

3) 创建物理标签保存索引（未经审计)。

4) 锁定基表，以便进行共享读访问，从而防止其他人对基表进行插入、更新和删除操作。

5) 使用侧插树读取基表，以实现未提交的读访问，由此可以装载索引。

注意:

由于这里的分区是未经审查的空分区,侧插树可以快速地装载执行专门优化的数据。

6) 索引装载完后，索引审查属性将被打开、并被连接至基表（使索引上线）。

7) 该事务既可以由系统提交、也可以由请求者稍后提交。

即使该事务由用户启动，如果执行基本语义检查后操作仍失败，则索引将不复存在、并且整个事务都将被回滚。

7.1.3.2 授权和可用性要求

一个索引和被其编入索引的表具有相同的安全性。

CREATE INDEX 防止对一个正在编制索引的表执行 INSERT、DELETE 和 UPDATE 操作。

如果其他进程在操作开始时已经锁定了表里的行，则 CREATE INDEX 语句将会继续等待、直到它的锁请求得到授权或出现超时为止。

您无法直接访问一个索引。

7.1.3.3 必需权限

发出一条 CREATE INDEX 语句，必须满足以下条件之一：

- 您是 DB_ROOT（数据库根用户）。
- 您正在一个共享 schema 里创建表。
- 您是私有 schema 的所有者。
- 您是该表的所有者。
- 您拥有 SQL_OPERATIONS 组件的 ALTER、ALTER_TABLE、CREATE 或 CREATE_INDEX 权限。

注意：

在这种情况下，如果您在私有 schema 中创建函数，您必须是 schema 的所有者。

7.1.3.4 索引的限制

非唯一索引里的列的长度总和加上底层表的聚集键的长度总和不得超过 2048 个字节。每个表的索引数量没有限制。

7.1.4 Create Index 例子

- 该例子在表的两列上创建一个索引：

```
CREATE INDEX xempname ON persnl.employee (last_name,
first_name);
```

7.2 删除索引

7.2.1 Drop Index 语句

该语句删除一个 Trafodion SQL 索引。

该语句是一条 Trafodion SQL 扩展语句。

```
DROP [VOLATILE] INDEX index
```

7.2.2 Drop Index 语法

- INDEX

待删除的索引。

7.2.3 Drop Index 注意事项

7.2.3.1 必需的权限

发出一条 DROP INDEX 语句，必须满足以下条件之一:

- 您是 DB__ROOT（数据库根用户）。
- 您是索引的所有者或与索引关联的表的所有者。
- 您拥有 SQL_OPERATIONS 组件的 DROP 或 DROP_INDEX 权限。

7.2.4 Drop Index 例子

- 该例子可以删除一个索引:

```
DROP INDEX myindex;
```

- 该例子可以删除一个可变索引:

```
DROP VOLATILE INDEX vindex;
```

8. 表和表的数据文件

QianBase 分布式关系型数据库的存储是 HBase，那所有 QianBase 表都是转化为 HBase 表的方式进行保存，并且 QianBase 每个表在 HBase 表中都有对应的表。

本章讲述以下内容:

8.1 QianBase 表

QianBase 的所有数据都保存在 HBase 中，对于 QianBase 来说 HBase 保存了两类表：数据库系统表和数据库用户表。

数据库系统表为：HBase 命令空间为 TRAF_RSRVD_* 的所有表,如下图（下图列出不代表全部数据库系统表）：

```
TRAF_RSRVD_1:TRAFODION._MD_.AUTHS
TRAF_RSRVD_1:TRAFODION._MD_.COLUMNS
TRAF_RSRVD_1:TRAFODION._MD_.DEFAULTS
TRAF_RSRVD_1:TRAFODION._MD_.INDEXES
TRAF_RSRVD_1:TRAFODION._MD_.KEYS
TRAF_RSRVD_1:TRAFODION._MD_.LIBRARIES
TRAF_RSRVD_1:TRAFODION._MD_.LIBRARIES_USAGE
TRAF_RSRVD_1:TRAFODION._MD_.LOBDescChunks__0317298208167
2746182_0002
TRAF_RSRVD_1:TRAFODION._MD_.LOBDescHandle__0317298208167
2746182_0002
TRAF_RSRVD_1:TRAFODION._MD_.LOBMD__03172982081672746182
TRAF_RSRVD_1:TRAFODION._MD_.OBJECTS
TRAF_RSRVD_1:TRAFODION._MD_.OBJECTS_UNIQ_IDX
TRAF_RSRVD_1:TRAFODION._MD_.REF_CONSTRAINTS
TRAF_RSRVD_1:TRAFODION._MD_.ROUTINES
TRAF_RSRVD_1:TRAFODION._MD_.SEQ_GEN
TRAF_RSRVD_1:TRAFODION._MD_.TABLES
TRAF_RSRVD_1:TRAFODION._MD_.TABLE_CONSTRAINTS
TRAF_RSRVD_1:TRAFODION._MD_.TABLE_CONSTRAINTS_IDX
TRAF_RSRVD_1:TRAFODION._MD_.TEXT
TRAF_RSRVD_1:TRAFODION._MD_.UNIQUE_REF_CONSTR_USAGE
TRAF_RSRVD_1:TRAFODION._MD_.VERSIONS
TRAF_RSRVD_1:TRAFODION._MD_.VIEWS
TRAF_RSRVD_1:TRAFODION._MD_.VIEWS_USAGE
TRAF_RSRVD_1:TRAFODION._PRIVMGR_MD_.COLUMN_PRIVILEGES
TRAF_RSRVD_1:TRAFODION._PRIVMGR_MD_.COMPONENTS
```

```
TRAF_RSRVD_1:TRAFODION._PRIVMGR_MD_.COMPONENT_OPERATIONS
TRAF_RSRVD_1:TRAFODION._PRIVMGR_MD_.COMPONENT_PRIVILEGES
TRAF_RSRVD_1:TRAFODION._PRIVMGR_MD_.OBJECT_PRIVILEGES
TRAF_RSRVD_1:TRAFODION._PRIVMGR_MD_.ROLE_USAGE
TRAF_RSRVD_1:TRAFODION._PRIVMGR_MD_.SCHEMA_PRIVILEGES
TRAF_RSRVD_1:TRAFODION._TENANT_MD_.RESOURCES
TRAF_RSRVD_1:TRAFODION._TENANT_MD_.RESOURCE_USAGE
TRAF_RSRVD_1:TRAFODION._TENANT_MD_.TENANTS
TRAF_RSRVD_1:TRAFODION._TENANT_MD_.TENANT_USAGE
TRAF_RSRVD_2:TRAFODION._REPOS_.METRIC_QUERY_AGGR_TABLE
TRAF_RSRVD_2:TRAFODION._REPOS_.METRIC_QUERY_TABLE
TRAF_RSRVD_2:TRAFODION._REPOS_.METRIC_SESSION_TABLE
TRAF_RSRVD_2:TRAFODION._REPOS_.METRIC_TEXT_TABLE
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_CONFIG
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_CUSTOMER
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_DISTRICT
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_HISTORY
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_ITEM
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_NEW_ORDER
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_OORDER
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_ORDER_LINE
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_STOCK
TRAF_RSRVD_3:TRAFODION.SEABASE.BMSQL_WAREHOUSE
TRAF_RSRVD_3:TRAFODION.SEABASE.CUSTOMER_I1
TRAF_RSRVD_3:TRAFODION.SEABASE.SB_HISTOGRAMS
TRAF_RSRVD_3:TRAFODION.SEABASE.SB_HISTOGRAM_INTERVALS
TRAF_RSRVD_3:TRAFODION.SEABASE.SB_PERSISTENT_SAMPLES
TRAF_RSRVD_3:TRAFODION.SEABASE.T
TRAF_RSRVD_5:TRAFODION._DTM_.LOB_META
TRAF_RSRVD_5:TRAFODION._DTM_.LOB_META_SHADOW
TRAF_RSRVD_5:TRAFODION._DTM_.MUTATION
TRAF_RSRVD_5:TRAFODION._DTM_.MUTATION_SHADOW
TRAF_RSRVD_5:TRAFODION._DTM_.SNAPSHOT
TRAF_RSRVD_5:TRAFODION._DTM_.TDDL
```

```
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG0
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG0_CONTROL_POINT
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG1
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG1_CONTROL_POINT
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG2
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG2_CONTROL_POINT
TRAF_RSRVD_1:TRAFODION._MD_.INDEXES
TRAF_RSRVD_1:TRAFODION._MD_.KEYS
TRAF_RSRVD_1:TRAFODION._MD_.LIBRARIES
TRAF_RSRVD_1:TRAFODION._MD_.LIBRARIES_USAGE
…
```

数据库系统表为：HBase 命令空间为 TRAF_1500000 的所有表,如下图：

```
TRAF_1500000:TRAFODION.BMS.A
TRAF_1500000:TRAFODION.BMS.SB_HISTOGRAMS
TRAF_1500000:TRAFODION.BMS.SB_HISTOGRAM_INTERVALS
TRAF_1500000:TRAFODION.BMS.SB_PERSISTENT_SAMPLES
```

除此外，还需要了解的信息有 HBase 表名和 QianBase 表名之间的关系。HBase 表名并不同直接等同于 QianBase 表名，存在如下公式：

HBase 表名=默认命名空间（TRAF_1500000）+ 默认实例名（TRAFODION）+ 用户自定义 Schema 名称 + QianBase 表名。

例如：HBase 的表 TRAF_1500000:TRAFODION.BMS.A，表示 QianBase 的 BMS Schema 下有一张 A 表。

8.2 表的数据

QianBase 表的持久化的数据有两类：日志和数据文件，它们最终持久化到 HDFS 文件系统中。

日志存储情况这里不进行赘述，请参考“数据库日志文件”章节的详细介绍。表的数据文件是以 HBase 的 HFile 的格式进行持久化，并按表和分区分类归类存储，如下图：

```
[trafodion@builder trafodion]$ swhadoop fs -lsr /hbase
lsr: DEPRECATED: Please use 'ls -R' instead.
19/04/25 20:24:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/.tmp
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/.tmp/data
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/.tmp/data/hbase
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/MasterProcWALs
-rw-r--r--   3 trafodion supergroup          0 2019-04-25 20:09 /hbase/MasterProcWALs/state-00000000000000000002.log
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/WALs
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:18 /hbase/WALs/builder,35303,1556248150817
-rw-r--r--   3 trafodion supergroup         83 2019-04-25 20:14 /hbase/WALs/builder,35303,1556248150817/builder%2C35303%2C1556248150817..meta.1556248482891.meta
-rw-r--r--   3 trafodion supergroup         83 2019-04-25 20:18 /hbase/WALs/builder,35303,1556248150817/builder%2C35303%2C1556248150817.default.1556248711137
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/default
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/meta
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/meta/.tabledesc
-rw-r--r--   3 trafodion supergroup        372 2019-04-25 20:09 /hbase/data/hbase/meta/.tabledesc/.tableinfo.0000000001
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/meta/.tmp
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:14 /hbase/data/hbase/meta/1588230740
-rw-r--r--   3 trafodion supergroup         32 2019-04-25 20:09 /hbase/data/hbase/meta/1588230740/.regioninfo
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:14 /hbase/data/hbase/meta/1588230740/.tmp
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:14 /hbase/data/hbase/meta/1588230740/info
-rw-r--r--   3 trafodion supergroup       1370 2019-04-25 20:14 /hbase/data/hbase/meta/1588230740/info/21f9b312852a424a9031daa8ce821a5e
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/meta/1588230740/recovered.edits
-rw-r--r--   3 trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/meta/1588230740/recovered.edits/3.seqid
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/namespace
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/namespace/.tabledesc
-rw-r--r--   3 trafodion supergroup        312 2019-04-25 20:09 /hbase/data/hbase/namespace/.tabledesc/.tableinfo.0000000001
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/namespace/.tmp
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:18 /hbase/data/hbase/namespace/d9fd28cbc3a26e1c5b47bc79c911bad1
-rw-r--r--   3 trafodion supergroup         42 2019-04-25 20:09 /hbase/data/hbase/namespace/d9fd28cbc3a26e1c5b47bc79c911bad1/.regioninfo
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:18 /hbase/data/hbase/namespace/d9fd28cbc3a26e1c5b47bc79c911bad1/.tmp
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:18 /hbase/data/hbase/namespace/d9fd28cbc3a26e1c5b47bc79c911bad1/info
-rw-r--r--   3 trafodion supergroup       1079 2019-04-25 20:18 /hbase/data/hbase/namespace/d9fd28cbc3a26e1c5b47bc79c911bad1/info/40c9ee88da504e60aa88d8f60a0a776e
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/namespace/d9fd28cbc3a26e1c5b47bc79c911bad1/recovered.edits
-rw-r--r--   3 trafodion supergroup          0 2019-04-25 20:09 /hbase/data/hbase/namespace/d9fd28cbc3a26e1c5b47bc79c911bad1/recovered.edits/2.seqid
-rw-r--r--   3 trafodion supergroup         42 2019-04-25 20:09 /hbase/hbase.id
-rw-r--r--   3 trafodion supergroup          7 2019-04-25 20:09 /hbase/hbase.version
drwxr-xr-x   - trafodion supergroup          0 2019-04-25 20:20 /hbase/oldWALs
```

HBase 所有数据默认保存在 HDFS 的/hbase/data 目的下，表的目录结构如下图：

```
--用户自定义的表空间
表空间/
        --自描述表隶属于那个命名空间
        表名/
          --用于保存 table 属性信息
          .tabledesc/
                .tableinfo.0000000001
          --用于保存 Region 信息
          RegionName（32 位编码）/
                .regioninfo
                列族名/
                      --表的数据文件
                      HFile(32 位编码)
                Recovered.edits/
```

9. 数据操作实用工具

本章节将介绍一些 QianBase SQL 中的一些实用工具：LOAD 加载数据、POPULATE 装载索引、PURGEDATA 清除数据、UNLOAD 卸载数据和更新统计信息。

本章具体讲述以下内容:

9.1 DUMP 数据导出

该语句的作用是导出指定对象的数据结构到指定文件。

```
DUMP object object_name TO FILE file_path

object is:
     SCHEMA
   | TABLE
   | VIEW
   | LIBRARY
   | FUNCTION
   | SEQUENCE
   | PROCEDURE
   | TRIGGER
   | PACKAGE
```

9.1.1 DUMP 语法

- object

  object 代表的是要 DUMP 出的对象类型。当导出对象为 SCHEMA 时，会导出该 SCHEMA 下除系统对象外的所有对象。

- file_path

  file_path 是你指定的导出的文件路径。可以使用绝对路径，也可以使用相对路径。使用相对路径时。如果使用的命令行工具是 sqlci，则相对路径以此时使用 sqlci 的路径为基准。如果使用的命令行工具是 trafci 时，则以 TRAF_HOME 目录为基准。当路径的最后一级目录不存在且有写权限创建时会自动创建，否则会报错找不到对应目录。

9.1.2 DUMP 注意事项

9.1.2.1 关于导出对象的权限

执行一个 DUMP 语句，以下条件之一必须成立:

- 您是 DB__ROOT（数据库根用户）。
- 您拥有 SQL_OPERATIONS 组件的 MANAGE_LOAD 组件权限。

9.1.2.2 关于导出路径的注意事项

执行一个 DUMP 语句，必须注意:

- 您对指定的导出路径有写权限。

  主要为写入权限。如果导出路径不具备相应的权限，将会导出失败。

- 导出路径必须使用单引号。

- 导出路径区分大小写。

- 导出路径可存在，可不存在。当导出路径存在时直接导出即可。当导出路径不存在时，只限于最后一级目录。如果只有最后一级目录不存在时，会直接创建目录且导出文件，如果连续多个目录不存在时，会提示失败。

9.1.2.3 关于导出文件的注意事项

执行一个 DUMP 语句
如果执行的导出对象是 SCHEMA 时，导出文件的命名为:
CATALOGNAME_SCHEMANAME
如果执行的导出对象不是 SCHEMA 时，导出文件的命名为:
CATALOGNAME_SCHEMANAME_OBJECTNAME

9.1.3 DUMP 例子

- 该例子导出指定的表到指定路径，此处使用绝对路径:

  ```
  DUMP TABLE trafodion.myschema.mytable TO FILE
  '/home/esgyn/dump';
  ```

  会在“/home/esgyn/dump”下生成表的导出文件“TRAFODION_MYSCHEMA_MYTABLE”。

- 该例子导出指定 schema 到指定路径，此处使用相对路径:

  ```
  DUMP SCHEMA myschema TO FILE './';
  ```

  会在当前目录下生成 SCHEMA 的导出文件“TRAFODION_MYSCHEMA”。

9.2 LOAD 数据加载

该语句使用 QianBase Bulk Loader 从单个源表（QianBase 表）装载数据到目标 QianBase 表。

QianBase Bulk Loader 直接在区域服务器里预处理和装载 HFile，由此绕过了写入路径和相关的成本。

写入路径开始于客户端，再将数据移至区域服务器，并在数据最终写入 HFile 的 HBase 数据文件后结束。

QianBase 批量装载进程将分为两阶段进行：

- 预处理阶段：在该阶段中，QianBase 读取 HDFS 源文件里的数据。基于目标表的分区方案对数据进行分区和排序，再生成键值对，最后将其填入 HFiles 中。QianBase 还对数据进行编码，以实现更快捷的存储和检索。
- 将文件装载到 HBase 阶段：该阶段使用 LoadIncrementalHFiles（completebulkload 工具）将生成的 HFiles 装载到区域服务器里。

LOAD 是一条 QianBase SQL 扩展语句。

```
LOAD  [WITH option[[,] option]...]
INTO target-table
SELECT ...FROM
source-table
option is:
TRUNCATE TABLE
| NO RECOVERY
| NO POPULATE INDEXES
| NO DUPLICATE CHECK
| NO OUTPUT
| INDEX TABLE ONLY
| UPSERT USING LOAD
```

9.2.1 LOAD 语法

- target-table

  这是数据将被载入的 QianBase 目标表的名称

- source-table

  这是带有源数据的 QianBase 表的名称。

  可以使用 HIVE.HIVEschema(例如 hive.hive.orders)在 QianBase 里访问 Hive 表。在 QianBase 访问之前，Hive 表需要存在于 Hive 中。如果您希望装载 HDFS 文件夹里现有的数据，则需要创建一个外部 Hive 表，它具有正确的字段、并指向包含数据的 HDFS 文件夹。您也可以在源数据上 指定一个 WHERE 子句作为过滤器。

- [WITH option[[,] option]...]

  可以为装载操作指定的一组选项。可以指定其中一个或多个选项:

  - TRUNCATE TABLE

    导致 Bulk Loader 在装载之前截断目标表。

    默认情况下，Bulk Loader 不会在装载数据之前截断目标表。

  - NO RECOVERY

    指定 Bulk Loader 不得使用 HBase 快照进行数据恢复。

    默认情况下，Bulk Loader 会使用 HBase 快照机制处理数据恢复。

  - NO POPULATE INDEXES

    指定 Bulk Loader 不得处理索引维护或填充索引。

    默认情况下，Bulk Loader 处理索引维护，在装载之前禁用索引，并在装载之后填充它们。

  - NO DUPLICATE CHECK

    导致 Bulk Loader 忽略源数据里的重复值。

    默认情况下，Bulk Loader 检查源数据里是否有重复值，并在检测到重复值时产生一个错误。

  - NO OUTPUT

    防止 LOAD 语句显示状态消息。默认情况下，LOAD 语句可以打印状态

消息，其中将列出 Bulk Loader 正在执行的步骤。

- INDEX TABLE ONLY

  指定目标表（索引）填充来自母表的数据。

- UPSERT USING LOAD

  指定使用“行集插入函数”,在没有事务的情况下插入数据目标表的数据。

9.2.2 LOAD 注意事项

9.2.2.1 必需的权限

发出一条 LOAD 语句，以下条件之一必须成立:

- 您是 DB__ROOT（数据库根用户）。
- 您是目标表的所有者。
- 您具有以下这些权限:
  - 目标表的 SELECT 和 INSERT 权限。
  - 目标表的 DELETE 权限（如果已指定 TRUNCATE TABLE）。
- 您拥有 SQL_OPERATIONS 组件的 MANAGE_LOAD 组件权限。

9.2.2.2 运行 LOAD 语句之前的配置

运行 LOAD 语句之前必须保证您已经根据这些准则配置了暂存文件夹、源表和 HBase。

9.2.2.3 HFiles 的暂存文件夹

Bulk Loader 使用 HDFS 文件夹作为 HFiles 的暂存区。 默认情况下, QianBase 使用/bulkload/作为暂存文件夹。该文件夹必须与运行 QianBase 的文件夹为同一个用户拥有。QianBase 也必须拥有该文件夹的完全权限。HBase 用户（运行 HBase 的用户）必须具有该文件夹的读写访问权限。

例如：

```
drwxr-xr-x - trafodion trafodion 0 2014-07-07 09:49
/bulkload.
```

9.2.2.4 Hive 源表

为了装载 HDFS 里存储的数据，您需要在启动装载之前创建一个 Hive 表，它具有正确的字段和类型，并且指向包含数据的 HDFS 文件夹。

9.2.2.5 HBase 快照

如果您没有在 LOAD 语句里指定 NO RECOVERY OPTION，则 Bulk Loader 将使用 HBase 快照作为数据恢复机制。

快照是一种轻量级操作，它可以复制一些元数据（数据不会被复制）。装载开始之前拍摄一个快照，并在装载成功后删除此快照。如果出现问题，可以进行数据恢复；此时快照可以将表恢复到装载之前的初始状态。

为了使用这种数据恢复机制，需要配置 HBase 为“允许快照”。

9.2.3 LOAD 例子

- 对于驻留在/hive/tpcds/customer_demographics 里的客户统计数据，应 使用以下 Hive SQL 创建一个外部 Hive 表：

```
create external table customer_demographics (
cd_demo_sk int,
cd_gender string,
cd_marital_status string,
cd_education_status string,
cd_purchase_estimate int,
cd_credit_rating string,
cd_dep_count int,
```

```
cd_dep_employed_count int,
cd_dep_college_count int
) row format delimited fields terminated by '|'
location '/hive/tpcds/customer_demographics'
```

- 使用这条 DDL 定义您希望将数据载入进的 Trafodion 表:

```
create table customer_demographics_salt (
cd_demo_sk int not null,
cd_gender char(1),
cd_marital_status char(1),
cd_education_status char(20),
cd_purchase_estimate int,
cd_credit_rating char(10),
cd_dep_count int,
cd_dep_employed_count int,
cd_dep_college_count int,
primary key (cd_demo_sk)
) salt using 4 partitions on (cd_demo_sk);
```

- 该例子显示了 LOAD 语句如何从 Hive (hive.hive.customer_demographics) 里装载 customer_demographics_salt 表:

```
>> load into customer_demographics_salt select * from
hive.hive.customer_demographics where cd_demo_sk <=
5000;
Task:LOAD Status:Started
Object:TRAFODION.HBASE.CUSTOMER_DEMOGRAPHICS_SALT
Task:DISABLE INDEX Status:Started
Object: TRAFODION.HBASE.CUSTOMER_DEMOGRAPHICS_SALT
Task:DISABLE INDEX Status:Ended
Object:TRAFODION.HBASE.CUSTOMER_DEMOGRAPHICS_SALT
Task:PREPARATION Status:Started
Object:TRAFODION.HBASE.CUSTOMER_DEMOGRAPHICS_SALT
Rows Processed:5000
```

```
Task:PREPARATION Status:Ended ET:00:00:03.199
Task:COMPLETION Status:Started
Object:TRAFODION.HBASE.CUSTOMER_DEMOGRAPHICS_SALT
Task:COMPLETION Status:Ended ET:00:00:00.331
Task:POPULATE INDEX Status:Started
Object: TRAFODION.HBASE.CUSTOMER_DEMOGRAPHICS_SALT
Task:POPULATE INDEX Status:Ended ET:00:00:05.262
```

9.3 POPULATE 装载索引

POPULATE INDEX 语句将母表数据快速插入到索引中。您可以在诸如 TrafCI 的基于客户端的工具里使用。

```
POPULATE INDEX utility。
POPULATE INDEX index ON table [index-option]
index-option is:
ONLINE | OFFLINE
```

9.3.1 POPULATE INDEX 语法

- index

这是一种 SQL 标识符，指定索引的简易名称。不能用该索引所属 schema 的名称来限定它。一个 schema 内的索引都有各自的命名空间，因此索引名可能与表名或约束名相同。然而，一个 schema 内的索引名必须各不相同。

- table

这是需要填充索引的表名。

- ONLINE

指定填充操作应该在线完成,即填充时,ONLINE 允许基表执行 DML 读写访问。此外，索引填充期间，ONLINE 还可以读取审计跟踪数据，以便重现基表的更新情况。

如果已经生成了大量的审计数据，并且您执行了很多 CREATE INDEX 操作，我们建议您不要使用 ONLINE 操作,因为它将为读取审计跟踪数据引入更多竞争。默认值为 ONLINE。

- OFFLINE

指定填充操作应该脱机完成。OFFLINE 允许对基表进行 DML 读取访问。 此时基表不可用于写入操作。OFFLINE 必须显式地指定。允许 SELECT 操作。

9.3.2 POPULATE INDEX 注意事项

POPULATE INDEX 的执行步骤如下:

POPULATE INDEX 操作可以运行在很多事务中。

- 实际装载操作运行于一个事务之外。如果操作失败，则将更快地进行回滚，因为它无需处理大量审计数据。另外，如果操作失败，索引将仍为空、未经审计、并且没有连接到基表（脱机)。
- 脱机执行 POPULATE INDEX 时，可以访问基表进行 DML 读操作。联机执行 POPULATE INDEX 期间（不含最后提交阶段)，可以访问基表进行 DML 读/写操作。
- 如果 POPULATE INDEX 操作意外失败，您可能需要再次删除索引、重新创建一个索引并填充。
- 通过读/写访问，Online POPULATE INDEX 可以读取审计跟踪数据，以便重现更新情况。如果您计划并行创建很多索引，或如果您有高等级的审计跟踪活动，则应考虑使用 OFFLINE 选项。 如果无法访问来源基表或目标索引、或如果数据装载操作因为某些资源问题或文件系统问题失败，则可能出现错误。

9.3.2.1 必需的权限

执行一个 DROP INDEX 语句，以下条件之一必须成立:

- 您是 DB__ROOT（数据库根用户)。
- 您是该表的所有者。
- 您拥有关联表的 SELECT 和 INSERT（或 ALL）权限。

9.3.2.2 POPULATE INDEX 例子

- 该例子从指定的表里装载指定的索引:

```
POPULATE INDEX myindex ON myschema.mytable;
```

- 该例子从使用默认 schema 的指定表里装载指定的索引:

```
POPULATE INDEX index2 ON table2;
```

9.4 PURGEDATA 清除数据

PURGEDATA 语句快速删除一个表和其相关索引的数据。您可以在诸如 TrafCI 之类的基于客户端的工具里使用 PURGEDATA 语句。

PURGEDATA 语句在行为上跟 TRUNCATE 语句是一样的。

```
PURGEDATA object
```

9.4.1 PURGEDATA 语法

- object

这是需要清除数据的表名。

9.4.2 PURGEDATA 注意事项

- 该对象可以是一个表名。
- 如果该表无法访问、或如果资源问题或文件系统问题导致删除操作失败，则将返回错误。
- 可变表不支持运行 PURGEDATA。

9.4.2.1 必需的权限

执行一个 PURGEDATA 操作，以下条件之一必须成立：

- 您是 DB__ROOT（数据库根用户）。
- 您是该表的所有者。
- 您拥有关联表的 SELECT 和 DELETE（或 ALL）权限。

9.4.2.2 可用性

PURGEDATA 将一个表标记为 OFFLINE，并在处理期间设置损坏位。如果 PURGEDATA 操作在完成之前失败，则该表和它的依赖性索引都将不可用，并且您必须再次运行 PURGEDATA，以便完成操作、并删除数据。在这种情况下将返

回错误 8551、以及附带的文件系统错误 59 或错误 1071。

9.4.3 PURGEDATA 例子

该例子清除指定表里的数据。如果一个表拥有多个索引，则它们的数据也将被清除。

```
PURGEDATA myschema.mytable;
```

9.5 TRUNCATE 清除数据

TRUNCATE 语句快速删除一个表和其相关索引的数据。您可以在诸如 TrafCI 之类的基于客户端的工具里使用 TRUNCATE 语句。
TRUNCATE 语句在行为上跟 PURGEDATA 语句是一样的。

```
TRUNCATE [TABLE] table-name;
```

9.5.1 TRUNCATE 语法

table-name
这是需要清除数据的表名。更多信息，请参阅数据库对象名称。

9.5.2 TRUNCATE 注意事项

- 该对象可以是一个表名。
- 如果该表无法访问、或如果资源问题或文件系统问题导致删除操作失败，则将返回错误。
- 临时表不支持运行 TRUNCATE。

9.5.2.1 必需的权限

执行一个 TRUNCATE 操作，以下条件之一必须成立：

- 您是 DB__ROOT（数据库根用户）。
- 您是该表的所有者。
- 您拥有关联表的 SELECT 和 DELETE（或 ALL）权限。

9.5.2.2 可用性

TRUNCATE 将一个表标记为 OFFLINE，并在处理期间设置损坏位。
如果 TRUNCATE 操作在完成之前失败，则该表和它的从属索引都将不可用，

并且您必须再次运行 TRUNCATE，以便完成操作、并删除数据。
在这种情况下将返回错误 8551、以及附带的文件系统错误 59 或错误 1071。

9.5.3 TRUNCATE 例子

- 该例子清除指定表里的数据。如果一个表拥有多个索引，则它们的数据也将被清除。

  TRUNCATE myschema.mytable;

9.6 UNLOAD 卸载数据

UNLOAD 语句将 Trafodion 表里的数据卸载到您指定的 HDFS 位置。抽取的数据既可以是压缩数据，也可以是未压缩数据，您可以自行选择。UNLOAD 是一条 Trafodion SQL 扩展语句。

```
UNLOAD [WITH option[ option]...]INTO ' target-location'
SELECT ...FROM source-table ...
option is:
   DELIMITER { ' delimiter-string' | delimiter-
asciivalue }
   | RECORD_SEPARATOR {' separator-literal' |
separatorascii-value }
| NULL_STRING ' string-literal'
| PURGEDATA FROM TARGET
| COMPRESSION GZIP
| MERGE FILE merged_file-path [OVERWRITE]
| NO OUTPUT
| { NEW | EXISTING } SNAPSHOT HAVING SUFFIX ' string'
```

9.6.1 UNLOAD 语法

- 'target-location'

  这是抽取数据将被写入的 HDFS 目标文件夹的完整路径名称。将该文件夹的名称用单引号括起来。

  指定文件夹名称为完整路径名称、而非相对路径。您必须对 HDFS 目标文件夹拥有写入权限。

  如果您并行运行 UNLOAD 语句，则 target-location（目标位置）中将产生多个文件。已创建文件的数量将等于 ESP 的数量。

- SELECT ...FROM source-table ...

  这是一条简单查询或是一条包含 GROUP BY、JOIN 或 UNION 子句的复杂查询。source-table 是带有源数据的 Trafodion 表的名称。

- [WITH option[option]...]

  这是为卸载操作指定的一组选项。如果您多次指定一个选项，则 Trafodion 将返回一个带有 SQLCODE:-4489 的错误。您可以指定其中的一个或多个选项：

  - DELIMITER { 'delimiter-string' | delimiter-ascii-value }

    指定分隔符是一个分隔字符串或一个 ASCII 值。 如果您没有指定该选项，Trafodion 将使用字符“|”作为分隔符。delimiter-string（分隔字符串）可以是任何 ASCII 或 Unicode 字符串。您也可以指定分隔符是一个 ASCII 值。有效值的范围从 1 到 255。指定一个十进制数值；目前暂不支持十六进制或八进制表示法。如果您正在使用一个 ASCII 值，则分隔符宽度仅限一个字符。如果指定一个 ASCII 值作为分隔符，则不得使用引号。

  - RECORD_SEPARATOR { 'separator-literal' | separator-ascii-value }

    该字符将被用于分隔输出文件里的连续记录或行。您可以指定一个字面值或 ASCII 值作为分隔符。默认值为换行符。 separator-literal 可以为任何 ASCII 或 Unicode 字符。您也可以指定分 隔符是一个 ASCII 值。有效值的范围从 1 到 255。指定一个十进制数值；目前不支持十六进制或八进制表示法。如果您正在使用一个 ASCII 值，则分隔符宽度仅限一个字符。如果指定一个 ASCII 值作为分隔符，则不得使用引号。

- NULL_STRING 'string-literal'

  指定一个用于表示空值的字符串。默认值为空字符串。

- PURGEDATA FROM TARGET

  卸载前导致目标 HDFS 文件夹里的文件被删除。

- COMPRESSION GZIP

  在抽取节点里使用 gzip 压缩，将数据以这种压缩格式写入磁盘。 GZIP 是目前唯一支持的压缩类型。如果您未指定该选项，则抽取的数据将被解压。

- MERGE FILE merged_file-path [OVERWRITE]

  将卸载文件合并到指定的 merged-file-path（合并文件路径）的单一文件里。如果您指定了压缩率，则卸载数据和合并文件都将具有压缩格式。如果您指定了可选的 OVERWRITE 关键字，则现有文件将被覆盖。否则，Trafodion 会

抛出一个错误。

- NO OUTPUT

  防止 UNLOAD 语句显示状态消息。默认情况下，UNLOAD 语句可以打印状态消息，将列出 Bulk Loader 正在执行的步骤。

- { NEW | EXISTING } SNAPSHOT HAVING SUFFIX 'string'

  在卸载期间发起 HBase 快照扫描。快照扫描期间，Bulk Unloader 将从查询解释计划中获得一份 Trafodion 表的清单，然后将为这些表创建和验证快照。指定一个后缀字符串'string'，它将被追加到每个表名后面。

9.6.2 UNLOAD 注意事项

- 您必须对 HDFS 目标文件夹拥有写入权限。
- 如果您多次指定一个 WITH 选项，则 Trafodion 将返回一个带有 SQLCODE:-4489 的错误。

9.6.2.1 必需的权限

发出一条 UNLOAD 语句，以下条件之一必须成立：

- 您是 DB__ROOT（数据库根用户）。
- 您是目标表的所有者。
- 您拥有目标表的 SELECT 权限。
- 您拥有 SQL_OPERATIONS 组件的 MANAGE_LOAD 组件权限

9.6.3 UNLOAD 例子

该例子显示 UNLOAD 语句如何将一个 Trafodion 表 TRAFODION.HBASE.CUSTOMER_DEMOGRAPHICS 里的数据抽取到一个 HDFS 文件夹 /bulkload/ customer_demographics:中

```
>>UNLOAD
+>WITH PURGEDATA FROM TARGET
+>MERGE FILE 'merged_customer_demogs.gz' OVERWRITE
+>COMPRESSION GZIP
+>INTO '/bulkload/customer_demographics'
+>select * from trafodion.hbase.customer_demographics
+><<+ cardinality 10e10 >>;
Task:UNLOAD Status:Started
Task:EMPTY TARGET Status:Started
Task:EMPTY TARGET Status:Ended ET:00:00:00.014
Task:EXTRACT Status:Started Rows Processed:200000
Task:EXTRACT Status:Ended ET:00:00:04.743
Task:MERGE FILES Status: Started
Task:MERGE FILES Status:Ended ET:00:00:00.063
--- 200000 row(s) unloaded.
```

10. 更新统计信息

该语句更新表里的一组或多组列的直方图统计信息。这些统计信息可被用来设计优化的访问计划。UPDATE STATISTICS 是一条 Trafodion SQL 扩展语句。

```
UPDATE STATISTICS FOR TABLE table [CLEAR | on-clause]
on-clause is:
ON column-group-list CLEAR
| ON column-group-list [histogram-option]...
column-group-list is:
column-list [,column-list]...
| EVERY COLUMN [,column-list]...
| EVERY KEY [,column-list]...
| EXISTING COLUMN[S] [,column-list]...
| NECESSARY COLUMN[S] [,column-list]...
column-list for a single-column group is:
column-name
| (column-name)
| column-name TO column-name
| (column-name) TO (column-name)
| column-name TO (column-name)
| (column-name) TO column-name
column-list for a multicolumn group is:
(column-name, column-name [,column-name]...)
histogram-option is:
GENERATE n INTERVALS
| SAMPLE [sample-option]
sample-option is:
[r ROWS]
| RANDOM percent PERCENT
| PERIODIC size ROWS EVERY period ROWS
```

本章讲述以下内容:

10.1UPDATE STATISTICS 语法

- table

命名一个将要更新统计信息的表。为了引用一个表，请使用 ANSI 逻辑名称。

- CLEAR

删除该表某些或所有直方图。当新的应用程序不再使用某些直方图统计信息 时，可以使用该选项。如果您没有指定 column-group-list，则应删除表中所有直方图；如果您指定了 column- group-list, 则仅删除组列表里的列。

- ON column-group-list

指定为一个或多个列组生成直方图统计信息，并可以清除该信息。您必须使用 ON 子句生成直方图表里存储的统计信息。

  - column-list

    指定如何定义一个 column-group-list。 该“列列表”可以同时代表单列组和多列组。

    - 单列组:

      column-name | (column-name) | column-name TO columnname | (column-name) TO (column-name)

      说明如何指定单独的列或一组单独的列。为了给单列生成统计信息，请将每列都列出。您可以列举每个单列的名称，有无括号均可。

    - 多列组:

      (column-name, column-name [,column-name]...)

      指定一个多列组。为了生成多列统计信息，请对括号中的列集进行分组。

      您不得在同一个列组里多次指定一个列名。

      可以为每个唯一列组生成一个直方图。重复组意味着同一个列组的任何排列都将被忽略不计，数据处理将继续进行。当再次为同一个用户表运行 UPDATE STATISTICS 语句时，该表的新数据将替换先前生成、并存储在该表直方图表里的数据。ON 子句中没有指定的列组直方图将在直方图表里保持不变。

  - EVERY COLUMN

EVERY COLUMN 关键字表明将为一个表的每个单独的列、以及任何构 成主键和索引的多列生成直方图统计信息。 例如，一个表已经定义了列 A、B、C 和 D，其中 A 列、B 列和 C 列将构成主键。 在这种情况下， ON EVERY COLUMN 选项可以为 A 列、B 列、C 列和 D 列生成单列直方图，此外它还可生成两个多列直方图(A, B, C)和(A, B)。EVERY COLUMN 选项与 EVERY KEY 选项功能一致，并带有单列的附加统计信息。

- EVERY KEY

  EVERY KEY 关键字表明将为构成主键和索引的列生成直方图统计信息。例如，一个表已经定义了 A 列、B 列、C 列和 D 列，如果主键由 A 列和 B 列构成，则将为(A, B)、A 和 B 生成统计信息。如果主键由 A 列、B 列和 C 列构成，则将为 (A,B,C)、(A,B)、A、B、C 生 成统计信息。如果主键由 A 列、B 列、C 列和 D 列构成，则将为(A, B, C, D)、(A, B, C) 、(A, B)和 A、B、C、D 生成统计信息。

- EXISTING COLUMN[S]

  EXISTING COLUMN 关键字表明该表目前所有的直方图都要更新。必须事先捕获统计信息，以便建立现有的列。

- NECESSARY COLUMN[S]

  NECESSARY COLUMN[S]关键字为直方图生成统计信息，该直方图是优化器请求的并不存在的。必须针对 NECESSARY COLUMN[S]启用更新统计自动化，以便生成统计信息。

- histogram-option
  - GENERATE n INTERVALS

    UPDATE STATISTICS 语句的 GENERATE n INTERVALS 选项接受 1 和 10,000 之间的数值。提高每个直方图的时间间隔数量可能对编译时间造成负面影响。

    可以为带有小型数值集和较大的数值频率方差的列增加时间间隔的数

量。例如，SALES 表里的列‘CITY’，存储了已售项目所在的城市代码，销售数据里的城市数量为 1538。

将时间间隔数量设置为一个大于或等于城市数量的数字（即将时间间隔数量设置为 1600）可以保证生成的直方图能够捕获每个城市的行数)。如果指定的数值 n 超过了列中唯一值的数量，则系统将只能生成与唯一值数量相等的时间间隔。

- SAMPLE [sample-option]

  该子句里指定的抽样用于收集表数据的子集。

  UPDATE STATISTICS 语句存储样本结果并生成直方图。

  如果您指定了不含附加选项的 SAMPLE 子句，则结果将取决于表里的行数。

  如果该表的行数不超过 1 万，则整个表都将被读取（无抽样)。

  如果该表的行数介于 1 万和 100 万之间，则将从该表里随机抽样 1 万行。

  如果该表的行数超过了 100 万，则可使用随机行样本读取该表行数的 1%，最多抽样 100 万行。

  > **提示**：默认抽样该表行数的 1%（最多抽样 100 万行）可以为优化器提供良好 的统计信息，从而生成良好的（执行）计划。

  如果没有指定 SAMPLE 子句，则该表的行数将低于指定值；另外，如果 样本大小超过了系统限制，则 Trafodion SQL 将从表里读取所有行。

- sample-option
  - r rows

    行样本被用于读取表中的 r 行。数值 r 必须是一个大于零的整数（r >0)。

  - RANDOM percent PERCENT

    指导 Trafodion SQL 随机地选择表里的行。数值百分比必须是一个

介于 0 和 100 之间的数值（0 < percent <= 100）。

此外，只有小数点右侧的前四位数才是有效位。例如：数值 0.00001 将被视为 0.0000，数值 1.23456 将被视为 1.2345。

- PERIODIC size ROWS EVERY period ROW

  指导 Trafodion SQL 每个周期的行里选择 size 第一数量的行。 Size 值必须是一个大于零且小于或等于 period 值的整数（0 < size <= period）。通过该 period 的指定行数可以定义其大小。Period 值必须是一个大于零的整数。

10.2UPDATE STATISTICS 注意事项

10.2.1 使用统计信息

使用 UPDATE STATISTICS 语句收集和保存列的统计信息。SQL 编译器使用直方图统计信息确定谓词、索引和表的选择率。由于选择率直接影响访问计划的成本，常规的统计信息收集可以提高 Trafodion SQL 选择有效访问计划的可能性。在一个表上运行 UPDATE STATISTICS 语句时，该表处于活动状态，并可用于查询访问。通过显著更改一个用户表的数据或定义而改变这个表时，应对其重新执行 UPDATE STATISTICS 语句。

10.2.2 直方图统计信息

编译器使用直方图统计为一条给定的 SQL 查询产生最佳计划。如果这些直方图不可用，编译器将做出默认假设、由此产生的计划可能也表现不佳。对于这个 问题，反映表中最新数据的直方图是最佳解决方案。

编译器不需要一个表所有列的直方图统计信息。例如，如果一列仅出现在选择 列表里，则其直方图统计信息将无关紧要。

如果列出现在以下情况中，那么直方图统计信息将非常有价值：

- 一个谓词中
- 一个 GROUP BY 列中
- 一个 ORDER BY 子句中
- 一个 HAVING 子句中
- 或类似子句中

除了单列直方图统计信息以外，编译器还需要多列直方图统计信息，例如，当 一个查询里出现 group by column-5, column-3, column-19 时，编译器就需要（column-5, column-3, column-19）组合的直方图统计信息。

10.2.3 必需的权限

执行一个 UPDATE STATISTICS 操作，以下条件之一必须成立：

- 您是 DB__ROOT（数据库根用户）。
- 您是目标表的所有者。
- 您拥有 SQL_OPERATIONS 组件的 MANAGE_STATISTICS 组件权限。

10.2.4 锁定

UPDATE STATISTICS 语句在操作期间暂时锁定用户表的定义（而非用户表自身）。UPDATE STATISTICS 句为用户表使用 READ UNCOMMITTED（未提交读）隔离级别。

10.2.5 事务

执行 UPDATE STATISTICS 语句前不得启动一个事务。必要时 UPDATE STATISTICS 语句可以同时运行多个自身的事务。

处理期间启动一个正在运行 UPDATE STATISTICS 语句的自身事务可能导致事务自动中止时间超标。

10.2.6 为列生成和清除统计信息

为了给特定列生成统计信息，应对表里的每个列或一连串列里的首列和末列进行命名。例如，假设一个表有连续列 CITY、STATE、ZIP，该列表为您可以指定的选项的例子：

单列组	括号里的单列组	多列组
ON CITY, STATE, ZIP	ON (CITY),(STATE),(ZIP)	ON (CITY, STATE) 或 ON (CITY,STATE,ZIP)
ON CITY TO ZIP	ON (CITY) TO (ZIP)	
ON CITY, STATE TO ZIP	ON (CITY), (STATE) TO (ZIP)	
ON CITY TO STATE,	ON (CITY) TO	

ZIP	(STATE), (ZIP)	

当表中有很多列，并且您希望得到一个列的子集的直方图时，那么 TO 规范将有用。

请勿混淆(CITY) TO (ZIP)和(CITY, STATE, ZIP)，后者引用了一个多列直方图。

您可以按照任何指定的列组合清除统计信息，而不必通过创建统计信息时使用的 column-group-list 执行该操作。

然而，这些统计信息将被一直保留，直到清除它们为止。

10.2.7 “列的列表”和访问计划

针对表数据访问计划里最常用的列生成统计信息，即定义在该表上的主键和索引、任何常在 WHERE 或 GROUP BY 子句(对表进行查询)中的谓词引用的列。使用 EVERY COLUMN 选项为构成主键和索引的单列或多列生成直方图。EVERY KEY 选项生成的直方图以构成主键和索引。

如果您经常在一个表的特定列上执行 GROUP BY 子句，则请使用 UPDATESTATISTICS 语句（由 GROUP BY 子句里的列组成）里的 multicolumn lists 生成直方图统计信息，以使优化器能够选择一个更好的计划。同样，当一个查询通过两列或更多列联接两个表时，multi-column lists 能帮助优化器选择一个更好的计划。

10.2.8 更新统计自动化

为了启用更新统计自动化，请在运行更新统计操作的会话里设置控制查询默认（CQD）属性 USTAT_AUTOMATION_INTERVAL。例如：

```
control query default USTAT_AUTOMATION_INTERVAL '1440';
```

USTAT_AUTOMATION_INTERVAL 的数值拟定为自动化时间间隔(以分钟为 单位)；但在 Trafodion Release 1.0 里，该值并不作为定时间隔。相反，任何大于零

的值都能启用更新统计自动化。

启用更新统计自动化后，对您希望优化的每条查询进行预处理。例如：

```
prepare s from select...;
```

PREPARE 语句使 Trafodion SQL 编译器在没有执行的情况下能编译和优化一个查询。缺少优化器所需的任何直方图都将导致这些列被标记为“需要直方图”。使用 ON NECESSARY COLUMN[S]针对每个表运行该 UPDATE STATISTICS 语句，以便生成所需的直方图：

```
update statistics for table table-name on necessary
columns sample;
```

10.3UPDATE STATISTICS 例子

- 该例子可以为 EMPLOYEE 表的列 jobcode、empnum、deptnum 和 (empnum, deptnum) 生成 4 个直方图。根据该表的大小和数据分布情况，每个直方图应该包含 10 个时间间隔。

```
UPDATE STATISTICS FOR TABLE employee ON
(jobcode),(empnum, deptnum) GENERATE 10 INTERVALS;
--- SQL operation complete.
```

- 该例子使用 DEPT 表的 ON EVERY COLUMN 选项生成直方图统计信息。This 语句将执行一次全表扫描，Trafodion SQL 确定默认的时间间隔数量。

```
UPDATE STATISTICS FOR TABLE dept ON EVERY COLUMN;
--- SQL operation complete.
```

- 假设一家建筑公司有潜在场地的 ADDRESS 表和一个包含 ADDRESS 表某些列的 DEMOLITION_SITES 表。主键为 ZIP。通过两个公共列联接这两个表:、

```
SELECT COUNT(AD.number), AD.street, AD.city, AD.zip,
AD.state
FROM address AD, demolition_sites DS
WHERE AD.zip = DS.zip AND AD.type = DS.type GROUP BY
AD.street, AD.city, AD.zip, AD.state;
```

为了生成该查询特定的统计信息，输入下列语句:

```
UPDATE STATISTICS FOR TABLE address ON (street),
(city), (state), (zip, type);
 UPDATE STATISTICS FOR TABLE demolition_sites ON
(zip, type);
```

- 该例子删除 DEMOLITION_SITES 表的所有直方图:

```
UPDATE STATISTICS FOR TABLE demolition_sites CLEAR;
```

- 该例子有选择性地删除 ADDRESS 表的 STREET 列的直方图：

```
UPDATE STATISTICS FOR TABLE address ON street CLEAR;
```

11. 共享缓存管理

共享缓存的引入解决了核心银行业务的一个关键性能瓶颈问题: 对于一个由成百上千需要高并发去访问的表组成的数据库，首次编译所需的时间是毫秒级的。通常情况下，在查询的首次编译过程中，编译器需要访问多个系统表才能获取所涉及表的系统数据，如列定义、分区方案、关联索引、唯一性和引用约束以及权限等等。这些系统数据查询必须进行编译和执行，每个查询可能需要几百毫秒甚至 1 秒才能完成。

通过引入共享缓存，数以万计的编译器可以对表描述信息[1]（关于表的编码系统数据，以二进制数据的形式存储在系统数据中）进行规模化访问。每个节点都有一个共享缓存。在数据库启动期间，表的描述信息会自动从系统表加载到共享缓存。

11.1共享缓存的配置

在 schema "_MD_"下的 defaults 系统表中配置以下 CQD 以启用共享缓存功能。

表 2-1 共享缓存 CQD 配置信息表

CQD	**值**	**说明**
TRAF_ENABLE_METADATA_LOAD_IN_CACHE	ON	启用以允许将系统表的描述信息加载到编译器缓存中。
TRAF_ENABLE_METADATA_LOAD_IN_SHARED_CACHE	ON	启用以允许将描述信息加载到共享缓存中。默认值为 OFF。
TRAF_STORE_OBJECT_DESC	ON	允许在创建表时自动生成表描述信息并将其存储在系统表中。

在 ms.env 中为以下环境变量设置适当的值。

[1] 描述信息指的是表结构的描述信息，比如表的列数，每一列的数据类型，索引信息，约束，统计信息等所有与表有关的描述信息。

表 2-2 环境变量设置参考表

环境变量	值	说明
SHARED_CACHE_SEG_SIZE	<大小(MB)>	默认设置为 256MB， 500 张表的描述信息需要 128MB。1400 个描述信息则设置为 500MB。

通过 trafci 执行此命令加载共享缓存:

```
trafci> load trafodion metadata into shared cache
```

11.2 实用工具

11.2.1 Load 共享缓存

load 语句将具有 stored desc 属性的表的描述信息加载到所有相关节点上的共享缓存中。

编译器需要读取多张系统表来获得完整的用户表结构，这个过程涉及到和 HBase 的多次交互，所以耗时长。为了加快编译时间，可以将用户表的描述信息预先加载到共享缓存中，来减少编译期间对系统数据表的访问。

11.2.1.1 语法:

```
load trafodion metadata into shared cache [for nodes (<node_list>)];

<node_list>::=<node>, … <node>
<node> ::= 带引号的字符串，用来命名集群中节点
```

从节点故障中恢复后，其中只有一部分节点需要加载其共享缓存，您将需要使用“for nodes”子句。

在加载操作期间，共享缓存被锁定。

11.2.1.2 举例:

加载整个集群的共享缓存:

```
load trafodion metadata into shared cache
```

加载节点'node2','node5'的共享缓存:

```
load trafodion metadata into shared cache for nodes
('node2','node5');
```

11.2.2 Enable 共享缓存

该语句在所有相关节点的共享缓存中启用按名称列出的表描述信息，或按 schema 名称列出 schema 中的所有表描述信息。一旦启用，所有编译器进程都可以查找表描述信息。

11.2.2.1 语法

```
alter trafodion metadata shared cache for
          { table <table_name>  |
            schema <schema_name> }
           enable;
```

<table_name>::= 有效的 SQL 标识符

<schema_name>::= 有效的 SQL 标识符

11.2.3 Disable 共享缓存

该语句在所有相关节点的共享缓存中按名称禁用表描述信息，或按 schema 名称禁用 schema 中的所有表描述信息。一旦禁用，任何编译器进程都无法从共享缓存中查找表描述信息。编译器必须读取系统表获取相同的信息。

11.2.3.1 语法

```
alter trafodion metadata shared cache for
```

```
          { table <table_name>  |
            schema <schema_name> }
           disable;
```

<table_name>::= 有效的 SQL 标识符

<schema_name>::= 有效的 SQL 标识符

11.2.4 Update 共享缓存

Update 语句使用存储在“_MD_”.TEXT 中的表描述信息更新共享缓存中的表描述信息。该操作在所有涉及的节点上并行执行。

11.2.4.1 语法

```
alter trafodion metadata shared cache for
          { table <table_name>  |
            schema <schema_name> }
           update;
```

<table_name>::= 有效的 SQL 标识符

<schema_name>::= 有效的 SQL 标识符

11.2.5 Delete 共享缓存

语句从共享缓存中删除表描述信息。该操作在所有涉及的节点上并行执行。

11.2.5.1 语法

```
alter trafodion metadata shared cache for
          { table <table_name>  |
            schema <schema_name> }
           delete;
```

<table_name>::= 有效的 SQL 标识符

<schema_name>::= 有效的 SQL 标识符

11.2.6 Clear 共享缓存

该语句将清除共享缓存并重新创建一个新缓存。该操作在所有涉及的节点上并行执行。

11.2.6.1 语法

```
alter trafodion metadata shared cache clear;
```

注意：

如果发生异常，或者现场需要定位问题，可以使用以下命令清除共享缓存，系统正常运行时请谨慎使用。

```
edb_pdsh -a tdm_arkcmp -testSharedCacheClean
```

11.2.7 Check 共享缓存

语句检查整个共享缓存或表描述信息的正常状态，并将摘要显示为语句的输出。

11.2.7.1 语法

```
check trafodion metadata shared cache [for table
<table_name>];
```

<table_name>::= 有效的 SQL 标识符

11.2.7.2 举例

```
1. check trafodion metadata shared cache for table
   t035_cube1;
   edev06.esgyn.local: a descriptor of 7030 bytes is found
   for the table. heap_sizes(bytes): total=52425904,
```

```
alloc=148840, free=52277064;
heap_starting_address=0x139000070
--- SQL operation complete.
```

edev06.esgyn.local ： 节点名称。

7030 bytes ： 表的描述信息的大小。

Total,alloc, free : 共享缓存用堆来管理，Total 指堆的总的大小，alloc 指已分配的堆的大小，free 指的是空余大小。

heap_starting_address： 堆的起始地址。一旦数据库启动完成，值就保持不变。

2. check trafodion metadata shared cache;

```
edev06.esgyn.local: entries: total=2, enabled=2;
sum_of_length(bytes): total=13612, enabled=13612;
heap_sizes(bytes): total=52425904, alloc=148840,
free=52277064; heap_starting_address=0x139000070
--- SQL operation complete.
```

entries: total=2, enabled=2; ： **total** 是指数据总共缓存表的描述信息的个数，**enabled** 指处于 enable 状态的表的个数。正常使用，两者应该是相等的。

sum_of_length(bytes): total=13612, enabled=13612;: **total** 指的是所有表的描述信息占用的字节数，**enable** 指的是处于 enable 状态的表的描述信息的字节数。

11.2.8 缓存一致性

验证实例中的所有节点上存储在共享缓存中的描述信息都一致。

11.2.8.1 语法

```
tdm_arkcmp -testSharedCacheFindAll
```

11.2.8.2 举例

```
[trafodion@conn40 trafodion]$ edb_pdsh -a tdm_arkcmp -
testSharedCacheFindAll| grep FindAll
conn40: FindAll(): finding all pairs of (key, value) from
hash table takes 800us. Total enabled found=62 Total
disabled found=0, size in bytes: total=977004,
avg=15758.1
conn41: FindAll(): finding all pairs of (key, value) from
hash table takes 787us. Total enabled found=62 Total
disabled found=0, size in bytes: total=977004,
avg=15758.1
conn42: FindAll(): finding all pairs of (key, value) from
hash table takes 1495us. Total enabled found=62 Total
disabled found=0, size in bytes: total=977004,
avg=15758.1
```

11.3共享缓存操作实例

cqd TRAF_STORE_OBJECT_DESC 设置之后,当前连接中创建的表和 schema 都会带有 stored desc 属性，可以用 showddl 查看。

11.3.1 加载共享缓存

11.3.1.1 Schema/表级别加载共享缓存:

通过运行以下命令可以加载 schema 或表级别的 shard cache:

1. ALTER TABLE table_name GENERATE STORED DESC;
2. ALTER SCHEMA schema_name GENERATE STORED DESC;
3. ALTER TRAFODION METADATA SHARED CACHE FOR TABLE schema_name.table_name ENABLE;
4. ALTER TRAFODION METADATA SHARED CACHE FOR TABLE schema_name.table_name UPDATE;
5. ALTER TRAFODION METADATA SHARED CACHE FOR SCHEMA schema_name ENABLE;

11.3.1.2 集群/节点级别加载共享缓存

1. 重新加载整个集群的 shared cache，linux 下执行命令 load_shared_cache verbose
2. 重新加载整个集群的 shared cache，执行 query，load trafodion metadata into shared cache
3. 重新加载单个节点的 shared cache，linux 下执行命令 load trafodion metadata into shared cache for nodes('your_node_short_name')

11.3.2 清除共享缓存

11.3.2.1 Schema/表级别清除共享缓存:

1. ALTER TABLE table_name DELETE STORED DESC;

2. ALTER SCHEMA schema_name DELETE STORED DESC;
3. ALTER TRAFODION METADATA SHARED CACHE FOR TABLE schema_name.table_name DISABLE;
4. ALTER TRAFODION METADATA SHARED CACHE FOR TABLE schema_name.table_name DELETE;
5. ALTER TRAFODION METADATA SHARED CACHE FOR SCHEMA schema_name DISABLE;
6. ALTER TRAFODION METADATA SHARED CACHE FOR SCHEMA schema_name DELETE;

11.3.2.2 集群/节点级别清除共享缓存:

1. 清除整个集群的 shared cache，linux 下运行 edb_pdsh -a tdm_arkcmp -testSharedCacheClean
2. 清除整个集群的 shared cache，执行 query， alter trafodion metadata shared cache clear
3. 清除当前节点的 shared cache，在当前点运行 linux 命令 tdm_arkcmp -testSharedCacheClean

11.3.3 查看共享缓存

1. Linux 下运行 edb_pdsh -a tdm_arkcmp -testSharedCacheFindAll，加上 grep，则可以过滤单张表。
2. 执行查询，check trafodion metadata shared cache for table T
3. 以下 SQL 查询的是一张表的描述信息的大小

```
select
o.catalog_name,
o.schema_name,
o.object_name,
sum(char_length(t.text))
```

```
from TRAFODION."_MD_".OBJECTS O,
TRAFODION."_MD_".TEXT T
where O.object_uid = T.text_uid
and (O.object_type = 'BT' or O.object_type = 'IX')
and T.text_type = 7
and o.schema_name='SHARED_CACHE_TEST_SCHEMA'
and o.object_name='CORE_BANK_TABLE'
group by 1,2,3
order by 4 desc;
```

11.4 日志

默认情况下，跟共享缓存相关的日志默认是打开的，其配置可以在以下文件中修改：

```
log4cxx.trafodion.sql.config
log4cxx.trafodion.masterexe.config
```

启用日志记录（从 QianBase-1.5.4 起默认）：

```
log4j.logger.SQL.SharedCache=DEBUG
```

您可以通过将日志记录级别从 DEBUG 更改为 INFO 来关闭日志记录。

11.4.1 举例

插入记录的示例：

```
2020-01-23 18:50:06,412, DEBUG, SQL.SharedCache, Node
Number: 0, CPU: 0, PIN: 7973, Process Name:
$Z00000007S5,,,SharedDescriptorCache::insert():
key="2007821175", value length=4736, inserted=1,
return=0x203a0fa0, heap sizes(bytes): total=104854480,
allocated=3811440, sanity=1
10946 2020-01-23 18:50:06,412, DEBUG, SQL.SharedCache,
```

```
Node Number: 0, CPU: 0, PIN: 7973, Process Name:
$Z00000007S5,,,SharedDescriptorCache::insert():
key="878789501", value length=2744, inserted=1, return
=0x203a30d0, heap sizes(bytes): total=104854480,
allocated=3814560, sanity=1
```

12. 数据库迁移

不同数据库之间的数据迁移是一个复杂的系统工程，包括元数据的迁移和数据迁移，元数据包含数据对象的 DDL 定义，数据迁移包含从其他数据源迁移到 QianBase 中，以及从 QianBase 迁移到其他数据目标。数据从源到目的的过程中往往还需要进行数据的清洗和必要的转换。

QianBase 支持 ANSI99 SQL 标准，可以很方便的进行元数据的 DDL 迁移；

QianBase 支持从各种主流数据源进行数据导入，也提供高效的数据导出工具。

QianBase 可以进行简单的数据清洗，更加复杂的数据清洗工作则需要利用 QianBase 的 JDBC/ODBC 接口开发专门的程序或使用专业的 ETL 软件。

支持数据库有：MySQL、Orcale。

具体操作可以查阅《QianBase 数据库迁移指南》。

13. 数据库备份与恢复

为了确保在发生灾难性故障后快速恢复数据库服务，QianBase 支持多种备份和恢复方式，例如：

- 全量数据库在线备份
- 在线备份和恢复 Schema 与表
- 基于时间点恢复

从 1.1.0 开始，QianBase 支持增量备份数据库，并支持指定 Schema 和表。

具体操作可以查阅《QianBase 备份恢复指南》。

14. 数据库日志文件

在数据库系统中，既有存放数据的文件，也有存放日志的文件。日志在内存中也是有缓存 Log buffer，也有磁盘文件 log file。在关系数据库系统中，我们需要数据库可靠，所谓的可靠就是当系统崩溃/故障等情况下，保证数据库的一致性和完整性，这也就是数据库日志文件的主要目的。本文主要描述存放日志的文件以及相关概念。

数据库中有一种特殊的“日志文件”叫 Redo（重做）Undo（撤销），传统意义上的日志文件是记录系统运行状态的。而 Redo/Undo 文件是数据库的一部分，主要用于数据恢复，保证数据的一致性和完整性，QianBase 同样实现了数据库日志系统，即：分布式日志系统。

分布式日志系统：首先在数据库接受客户的 Commit 操作后，会将操作信息到保存在 Hadoop 的分布式文件系统（HDFS）中，这样日志文件是安全的。这样日志文件信息对集群所有节点都可见，也就为节点平滑迁移做好了基础，平滑迁移不是本节重点，这里不展开说明。其次，为了实现日志的分布式，与传统的单机关系型数据库有所以不同，日志文件包含两类文件：TLOG 和 HLOG。HLOG 是节点保存，记录每个节点的操作日志信息。TLOG 记录每个 HLOG 的操作关系，以及执行情况。

本章讲述以下内容:

14.1TLOG 管理

TLOG 日志信息是保存在 Hbase 表中：*._DTM_.TLOG*_CONTROL_POINT，可以通过 HBase 相关查询工具进行查阅。

在集群安装启动完毕后，通过下面的方式可以查看到相关的表。

```
[root@test01 ~]# hbase shell
Java HotSpot(TM) 64-Bit Server VM warning: Using
incremental CMS is deprecated and will likely be removed
in a future release
19/06/28 10:33:11 INFO Configuration.deprecation:
hadoop.native.lib is deprecated. Instead, use
io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported
commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.3, rUnknown, Sat Mar 17 04:43:46
PDT 2018
hbase(main):001:0> list
…
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG0
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG0_CONTROL_POINT
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG1
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG1_CONTROL_POINT
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG2
TRAF_RSRVD_5:TRAFODION._DTM_.TLOG2_CONTROL_POINT
…
```

14.2HLOG 管理

HLOG 即为 HBase 原生的 WAL Log，文件将会按 HRegionService 划分，分别进行记录。默认情况下 HLOG 保存在 HDFS 上的/hbase/WAL/目录下。

```
[trafodion@localhost trafodion]$ swhdfs dfs -lsr
/hbase/WALs/
lsr: DEPRECATED: Please use 'ls -R' instead.
drwxr-xr-x   - hbase hbase          0 2019-06-26 10:30
/hbase/WALs/test02,60020,1561516077118
drwxr-xr-x   - hbase hbase          0 2019-06-28 10:22
```

```
/hbase/WALs/test02,60020,1561516327086
-rw-r--r--   1 hbase hbase      5103 2019-06-28 10:22
/hbase/WALs/test02,60020,1561516327086/test02%2C60020%2C
1561516327086.null0.1561684928478
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:22
/hbase/WALs/test02,60020,1561516327086/test02%2C60020%2C
1561516327086.null0.1561688530331
-rw-r--r--   1 hbase hbase      5103 2019-06-28 10:22
/hbase/WALs/test02,60020,1561516327086/test02%2C60020%2C
1561516327086.null1.1561684928486
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:22
/hbase/WALs/test02,60020,1561516327086/test02%2C60020%2C
1561516327086.null1.1561688530340
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:22
/hbase/WALs/test02,60020,1561516327086/test02%2C60020%2C
1561516327086.null2.1561688530195
drwxr-xr-x   - hbase hbase         0 2019-06-28 10:38
/hbase/WALs/test03,60020,1561516318367
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:38
/hbase/WALs/test03,60020,1561516318367/test03%2C60020%2C
1561516318367.meta.1561689499826.meta
-rw-r--r--   1 hbase hbase      5103 2019-06-28 10:23
/hbase/WALs/test03,60020,1561516318367/test03%2C60020%2C
1561516318367.null0.1561684967093
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:23
/hbase/WALs/test03,60020,1561516318367/test03%2C60020%2C
1561516318367.null0.1561688568542
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:23
/hbase/WALs/test03,60020,1561516318367/test03%2C60020%2C
1561516318367.null1.1561688568554
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:23
/hbase/WALs/test03,60020,1561516318367/test03%2C60020%2C
1561516318367.null2.1561688568563
drwxr-xr-x   - hbase hbase         0 2019-06-26 10:30
```

```
/hbase/WALs/test04,60020,1561516061849
drwxr-xr-x   - hbase hbase         0 2019-06-28 10:33
/hbase/WALs/test04,60020,1561516312119
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:33
/hbase/WALs/test04,60020,1561516312119/test04%2C60020%2C
1561516312119.meta.1561689201843.meta
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:22
/hbase/WALs/test04,60020,1561516312119/test04%2C60020%2C
1561516312119.null0.1561688522249
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:22
/hbase/WALs/test04,60020,1561516312119/test04%2C60020%2C
1561516312119.null1.1561688521836
-rw-r--r--   1 hbase hbase   13571983 2019-06-28 10:22
/hbase/WALs/test04,60020,1561516312119/test04%2C60020%2C
1561516312119.null2.1561684920180
-rw-r--r--   1 hbase hbase        83 2019-06-28 10:22
/hbase/WALs/test04,60020,1561516312119/test04%2C60020%2C
1561516312119.null2.1561688522659
```

15. 数据库初始化命令

QianBase 提供一系列初始化命令对数据库元数据及权限进行初始化或其他操作。详情请见下表：

命令	功能	说明
INITIALIZE TRAFODION;	此命令执行后，数据库元数据表将被创建	
INITIALIZE TRAFODION,MINIMAL;	最小限度的初始化数据库	
INITIALIZE TRAFODION,NO RETURN STATUS;	此命令执行后，数据库元数据表将被创建，但不打印具体的创建信息	
INITIALIZE TRAFODION,MINIMAL,NO RETURN STATUS;	最小限度的初始化数据库，不打印具体的创建信息	
INITIALIZE TRAFODION,DROP;	此命令执行后，数据库元数据表将全部删除	
INITIALIZE TRAFODION,DROP FORCE;	强制删除全部数据库元数据表	需要打开 CQD: INITIALIZE_DROP_FORCE
INITIALIZE TRAFODION,CREATE METADATA VIEWS;	创建_MD_下的元数据视图	
INITIALIZE TRAFODION,DROP METADATA VIEWS;	删除_MD_下的元数据视图	
INITIALIZE TRAFODION,UPGRADE;	升级 Metadata	
INITIALIZE TRAFODION,CREATE SCHEMA OBJECTS;	创建元数据 schema 对象	
INITIALIZE TRAFODION,CREATE LIBRARY MANAGEMENT;	创建_LIBMGR_这个 schema 下的 library DB__LIBMGRNAME	

INITIALIZE TRAFODION,DROP LIBRARY MANAGEMENT;	删除_LIBMGR_这个 schema 下的 library DB__LIBMGRNAME 和 DB__LIBMGR_LIB_CPP	
INITIALIZE TRAFODION,UPGRADE LIBRARY MANAGEMENT;	升级_LIBMGR_这个 schema 下的 library DB__LIBMGRNAME 和 DB__LIBMGR_LIB_CPP	
INITIALIZE TRAFODION,CREATE REPOSITORY;	创建_REPOS_这个 schema 和 schema 下的表	
INITIALIZE TRAFODION,DROP REPOSITORY;	删除_REPOS_这个 schema 和 schema 下的表	
INITIALIZE TRAFODION,UPGRADE REPOSITORY;	升级_REPOS_这个 schema 和 schema 下的表	
INITIALIZE TRAFODION,CREATE BACKUP REPOSITORY;	创建_REPOS_这个 schema 和 schema 下的表 BACKUP_OBJECTS_METRICS 和 BACKUP_OPERATION_METRICS，与 INITIALIZE TRAFODION,CREATE EPOSITORY 有区别。目前此功能不完善。	需要打开 CQD: TRAF_BACKUP_REPOSITORY_SUPPORT，否则运行无效果
INITIALIZE TRAFODION,DROP BACKUP REPOSITORY;	删除_REPOS_这个 schema 下的表 BACKUP_OBJECTS_METRICS 和 BACKUP_OPERATION_METRICS，与 INITIALIZE TRAFODION,DROP EPOSITORY 有区别。目前此功能不完善。	需要打开 CQD: TRAF_BACKUP_REPOSITORY_SUPPORT，否则运行无效果
INITIALIZE TRAFODION,UPGRADE BACKUP REPOSITORY;	目前不支持	

INITIALIZE TRAFODION,UPDATE METADATA INDEXES;	升级_MD_下的元数据索引	
INITIALIZE TRAFODION,UPGRADE NAMESPACE;	升级 NAMESPACE	
INITIALIZE AUTHORIZATION;	此命令执行后权限管理的 schema（_PRIVMGR_MD_）和_PRIVMGR_MD_这个 schema 下的表将会被创建，如果权限管理的表存在则不执行什么操作	
INITIALIZE AUTHORIZATION,CLEANUP;	此命令执行后权限管理的 schema（_PRIVMGR_MD_）和_PRIVMGR_MD_这个 schema 下的表将被删除	
INITIALIZE AUTHORIZATION,DROP;	此命令执行后权限管理的 schema（_PRIVMGR_MD_）和_PRIVMGR_MD_这个 schema 下的表将被删除	
INITIALIZE AUTHORIZATION,UPDATE SOFTWARE VERSION;	目前不支持	
INITIALIZE AUTHORIZATION,CREATE LOB METADATA;	目前不支持	
INITIALIZE AUTHORIZATION,DROP LOB METADATA;	目前不支持	