UDBTX-TDS 用户手册

深圳九有数据库有限公司

UDBTX-TDS 用户手册

1 使用客户端工具连接到 UDBTX-TDS

可以使用 SQL Server 客户端在 TDS 端口上与 UDBTX-TDS 连接。从 Windows 系统连接时，SQL Server Management Studio （SSMS） 或客户端是不错的选择。如果要从 Linux 系统连接到 TDS 端口，则是一个不错的选择。

也可以使用其他使用 TDS 协议的客户端，但 UDBTX-TDS 不正式支持。在 Linux 上，这些包括 FreeTDS 命令行工具tsql

您可以使用任何 UDBTX 客户端在 UDBTX 端口上连接到 UDBTX-TDS。 和 pgAdmin 是在 Windows 和 Linux 平台上运行并使用 UDBTX 协议的开源客户端的示例。

1.1 用于连接到数据库实例sqlcmd

连接到 UDBTX-TDS 并与之交互的一种方法是使用 SQL Server 实用工具。UDBTX-TDS 连接字符串采用以下形式：

sqlcmd -S host.sample.com,1433 -U PUT_USER_HERE -P PUT_PASSWORD_HERE -d PUT_DBNAME_HERE

参数：

-S是数据库实例的服务器名称和 TDS 端口。

-U是用户的登录名。

-P是与用户关联的密码。

-d是要连接到的数据库。此值是可选的;如果省略，客户端将连接到数据库。master

连接后，可以使用熟悉的 T-SQL 语法创建和管理数据库对象。

1.2 使用 SSMS 进行连接

SQL Server Management Studio （SSMS） 是常用的 SQL Server 客户端。加载 SSMS 时，它可能会尝试与“对象资源管理器”对话框进行连接。如果此对话框默认打开，请点击 。UDBTX-TDS 仅支持从查询编辑器进行连接。

1.3 使用tsql连接

tsql是作为 FreeTDS 的一部分提供的命令行工具。它允许您从 Linux 终端连接 UDBTX-TDS（或任何其他启用 TDS 的数据源）并与之交互。

UDBTX-TDS 连接字符串采用以下形式：

tsql -S database.example.com -p 1433 -U unvdb -P secretpassword

有关使用 的信息，请参阅 FreeTDS 文档。

1.4 使用ud_sql连接

您可以使用 UDBTX 的客户端ud_sql在 UDBTX 端口上连接到 UDBTX-TDS（默认）。 支持 UDBTX 样式的 SQL 语法。

下面是一个如何连接的示例：

ud_sql -h database.example.com -p 6789 -U unvdb -d dbname

2 UDBTX-TDS互操作性指南

UDBTX-TDS 旨在通过 TDS 端口通过 T-SQL 语句在 UDBTX之上提供 T-SQL 语义。UDBTX-TDS 数据库也可以通过带有 UDBTX SQL语句的标准 UDBTX 端口访问。当 UDBTX 语句或 UDBTX 访问发生在 UDBTX-TDS 的上下文中时，UDBTX 和 UDBTX-TDS 之间可能会发生干扰，从而可能影响语法、语义以及与 UDBTX-TDS 未来版本的兼容性。如果发生此类情况，UDBTX-TDS 用户应注意与架构名称、标识符、权限、事务语义、多个结果集、默认排序规则和将来的 UDBTX-TDS 升级相关的潜在问题。

2.1 UDBTX-TDS/UDBTX 互操作性

UDBTX-TDS for UDBTX 提供了两个端点，通过这些端点可以访问 UDBTX-TDS 数据库：TDS 端口（默认值：1433）提供具有 T-SQL 语法和 T-SQL 语义的 T-SQL 接口，以及 UDBTX 端口（默认值：5432）提供具有 UDBTX SQL 语法和语义的 UDBTX 接口。

术语说明：在本文档中，TDS 和 T-SQL 以及 UDBTX 和 UDBTX SQL 用作 SQL 语句类型以及与数据库的相应连接的同义词。

在本文档的上下文中，UDBTX-TDS 数据库是一个 UDBTX 数据库，用于模拟 T-SQL 语句中的 SQL Server 多数据库环境，例如，使用 and 数据库和一个或多个用户数据库。默认情况下，此数据库名为 。TDS 连接将始终放置在此 UDBTX-TDS 数据库中，即使通过 T-SQL 不可见也是如此。每个 UDBTX 实例只能有一个 UDBTX-TDS 数据库。mastertempdbUDBTX-TDS_db

由于 UDBTX-TDS 数据库最终是一个 UDBTX 数据库，因此从技术上讲，可以从 UDBTX 端口访问它并针对它运行 UDBTX SQL 语句。通过 UDBTX 访问 UDBTX-TDS 数据库时应谨慎，因为需要考虑复杂性。

一个简单的方法是完全避免从 UDBTX 访问 UDBTX-TDS 数据库。想要使用 UDBTX 功能是有正当理由的。例如，当 UDBTX-TDS 不支持 T-SQL 的特定方面时，可以在 UDBTX 中本机实现该方面，以允许将应用程序迁移到 UDBTX-TDS UDBTX。

每当执行上下文不是纯 TDS/T-SQL 时，即在 UDBTX-TDS 上下文中执行 UDBTX SQL 语句时，互操作性注意事项就适用。当出现以下情况时，就是这种情况：

在 UDBTX 连接中，UDBTX 语句针对 UDBTX-TDS 创建的对象（例如表、视图、过程）执行。

在 TDS 连接中，T-SQL 语句针对 UDBTX 创建的对象（例如表、视图、过程）执行，或者在从 T-SQL 调用 UDBTX 内置函数时执行。根据客户反馈，这似乎是这些方案中更常见的。

在这两种情况下，这实际上意味着在访问/执行以不同的 SQL 语句创建的对象时，SQL 语句在会话中暂时（且透明地）切换（这不适用于访问表）。

当上述情况之一发生在基于 UDBTX-TDS 的应用程序中时，可能需要考虑以下一个或多个互操作性方面。UDBTX-TDS 用户负责确定具有此类混合 SQL 语句方案的应用程序是否按预期工作。此外，用户在升级到 UDBTX-TDS 的未来版本时应预见到潜在问题;此类问题可能需要由用户自己解决。

架构名称

UDBTX-TDS 通过将各种 SQL Server 数据库扁平化为 UDBTX 数据库（默认为 ）中的架构来模拟 SQL Server 多数据库结构。例如，这意味着创建为 的 T-SQL 表和创建为的过程必须分别从 UDBTX 中引用 作为 table 和 procedure ，请参阅将 UDBTX-TDS 与单个数据库或多个数据库一起使用。此外，SQL Server 使用与 UDBTX 不同的规则来解析没有架构名称的对象名称。

在 UDBTX-TDS 数据库中操作连接的 UDBTX 时，必须考虑这些差异。

标识符

UDBTX 的最大标识符长度为 63 个字符，而 SQL Server 最多支持 128 个字符。此外，UDBTX 对索引名称有更严格的唯一性要求。UDBTX-TDS 通过在内部附加或替换部分此类标识符来处理这些限制，该字符串包含一个 32 个字符的字符串，表示该标识符的哈希值。虽然这在 T-SQL 中是透明的，但从 UDBTX 中查看时，标识符 with-hash 是对象名称。

例如，在表上命名的索引将在 UDBTX-TDS 内部命名。在这里，UDBTX-TDS 将索引名称与表名称沿着使用 MD5 生成的字符串连接起来，作为小写索引名称。

权限

为了实现 SQL Server 权限模型，UDBTX-TDS 创建了 UDBTX 角色来反映 SQL Server 数据库主体等概念。这些 UDBTX-TDS 管理的 UDBTX 构造在 T-SQL 中是透明的，但在 UDBTX 连接中是可见的。

重要的是要认识到，不应从 UDBTX 修改 UDBTX-TDS 管理的 UDBTX 角色，并且不应从 UDBTX 授予/撤消对 UDBTX-TDS 创建的对象的权限。虽然目前在技术上可以这样做，但这可能会导致 UDBTX-TDS 实例与 UDBTX-TDS 的未来版本不兼容，并可能导致未来的升级失败。它还可能导致 UDBTX-TDS 的当前或未来版本出现意外的权限配置，因为与安全相关的 T-SQL 语句可能由于干扰在 UDBTX 中执行的操作而无法获得预期结果。

UDBTX-TDS 保证 T-SQL 在 UDBTX-TDS 的 TDS 连接中创建的表（或其他 SQL 对象）在通过 UDBTX 端口与连接到 TDS 时具有相同的权限进行访问，而无需运行任何其他 UDBTX 语句（尽管此处列出的其他一些注意事项， 与架构名称更改一样，可能仍然适用）。但是，不能保证相反的方式：在 UDBTX 中创建对象时，它可能无法从 TDS 连接中可见或访问，或者可以使用不同的权限访问它。因此，应在 T-SQL 中创建将从 UDBTX 访问的架构和对象。

事务语义

尽管 T-SQL 和 UDBTX SQL 的语义基本相同，但在事务回滚和错误处理方面存在一些显着差异。例如，在违反约束的情况下，T-SQL 默认回滚发生错误的语句，然后继续执行下一个语句，同时保持活动事务处于打开状态。相比之下，UDBTX SQL 将回滚整个事务并退出区块。

从 TDS 连接执行 T-SQL 时，将应用 T-SQL 事务语义;在 UDBTX 连接上执行 UDBTX 语句时，将应用 UDBTX 事务语义。但是，在连接中混合或组合两种语句时，原则上将忽略 T-SQL 事务语义，仅使用 UDBTX 语义。这可能意味着 T-SQL 过程在 UDBTX 上下文中执行时的行为可能会有所不同。

多个结果集

T-SQL 支持从批处理或过程向客户端返回多个结果集，但 UDBTX 不支持：对于过程，UDBTX 根本不支持返回结果集，对于批处理，UDBTX 仅发送第一个结果集。在这种情况下，如果在 UDBTX 上下文中执行 T-SQL 过程/批处理，客户可能无法获得预期的结果。

默认排序规则

T-SQL 和 UDBTX SQL 中的默认排序规则不同，这可能会导致一些语义差异。例如，在 UDBTX-TDS T-SQL 中默认为 true，因为默认排序规则不区分大小写;但在 UDBTX SQL 中，默认情况下这将是 false，其中排序规则区分大小写。'a' = 'A'

未来的 UDBTX-TDS 升级

UDBTX-TDS 用户可能希望在 UDBTX 中实现迁移的 T-SQL 应用程序的一部分，最可能的原因是解决 UDBTX-TDS 当前不支持的特定 T-SQL 功能，因此用户随后尝试在 UDBTX 中实现该部分。如果 UDBTX-TDS 在将来的版本中开始支持这些功能，则应用程序将与该未来版本不兼容。这可能意味着升级后的未来实例可能无法正常工作，或导致升级失败。

2.2 互操作性方案

首先，最简单的零互操作性方案是将应用程序垂直拆分为 T-SQL 部分和 UDBTX 部分，每个部分都有自己的用户和架构，并且它们之间没有交互。应用程序将打开与 UDBTX-TDS 数据库的 TDS 连接，以及与单独的 UDBTX 数据库的 UDBTX 连接。在这种情况下，实际上在数据库级别上根本不存在互操作性问题，因为 UDBTX 访问和 T-SQL 访问是完全分开的，不会干扰。（请注意，当两个连接使用相同的用户名进行连接时，可能存在依赖关系）。

一个更相关的变体是 UDBTX 连接访问 UDBTX-TDS 数据库，但仅访问 UDBTX 创建的对象，而 T-SQL 仅访问 UDBTX-TDS 创建的对象（例如表、视图、过程）。在这种情况下，没有干扰，但在实践中，通常需要在 T-SQL 上下文和 UDBTX 上下文之间交换信息，例如通过经常访问的表。这样的表应该在 T-SQL 中创建，而不是在 UDBTX 中创建，但除此之外，这种情况应该没有问题。互操作性方面：模式名称;标识符;权限。

另一种几乎没有预期并发症的情况是 UDBTX 连接仅从 UDBTX-TDS 创建的表中读取。互操作性方面：模式名称;标识符;权限。从视图读取数据时，可能还有其他注意事项与这些视图引用的架构名称和标识符相关。

当 UDBTX 连接还修改 UDBTX-TDS 创建的表或执行 DDL 来修改此类对象时，事务语义方面也将适用。

更复杂的情况是，当 UDBTX 连接执行 T-SQL 过程（直接或通过 UDBTX 过程）或修改具有 UDBTX-TDS 创建的触发器的 UDBTX-TDS 创建的表时。生成的语义应由 UDBTX-TDS 用户仔细验证。

2.3 互操作性示例

不支持的语句

考虑一个包含 100 条语句的 T-SQL 存储过程，其中 UDBTX-TDS 完全支持 99 条语句，但 1 条语句不支持。如果可以将不受支持的语句重写为 UDBTX 过程并从 T-SQL 语句调用，则这可能是迁移该过程的解决方案。但是，在实践中，只有当过程的其余部分也在 UDBTX 中重写时，才能在 UDBTX 中表示不受支持的语句。用户需要决定在多大程度上重写、重构或重新设计 T-SQL 或 UDBTX 中的过程。

表分区

UDBTX-TDS 目前不支持 SQL Server 的表分区。可以在 UDBTX-TDS 中创建一个表，并在 UDBTX 中修改该表以应用分区。由于分区是一种性能特性，应该在功能上是中立的，因此原则上它应该有效。但是，如果将来的 UDBTX-TDS 版本支持 T-

SQL 样式分区，它可能会与 UDBTX-TDS 不兼容。

全文搜索

通过重构应用程序，可以将执行全文搜索的应用程序迁移到 UDBTX-TDS，以便将全文搜索封装在 UDBTX 存储过程中，然后从 T-SQL 调用该存储过程。同样在这里，可能会出现与未来 UDBTX-TDS 版本的不兼容问题。

PostGIS 扩展

若要迁移具有地理空间 T-SQL 功能的应用程序，可以选择考虑使用 PostGIS 扩展。但是，这可能需要更广泛的重构，以及设计一种在 T-SQL 和 PostGIS 扩展之间传递数据的机制。将来的兼容性注意事项适用。

XML 处理

对于使用 XML 处理的 T-SQL 应用程序，如果 XML 功能可以拆分为 UDBTX 过程和函数，则可以使此功能正常工作。此类解决方法可能很复杂。将来的兼容性注意事项适用。

3 特性支持/不支持

3.1 SQL语句

功能	支持
@@variable	不支持
ADD SIGNATURE	不支持
ALTER AUTHORIZATION	不支持
ALTER DATABASE	不支持
ALTER DATABASE options	不支持
ALTER FUNCTION	不支持
ALTER INDEX	不支持
ALTER PROCEDURE	不支持
ALTER SCHEMA	不支持
ALTER SERVER CONFIGURATION	不支持
ALTER SERVER ROLE	不支持
ALTER TABLE	不支持
ALTER TRIGGER	不支持
ALTER VIEW	不支持

功能	支持
Aggregate functions	不支持
BULK INSERT	不支持
CHECKPOINT	不支持
CHECKSUM	支持
CLOSE KEY	不支持
CLUSTERED index	不支持
COLUMNPROPERTY	支持
CONNECTIONPROPERTY	支持
CREATE DATABASE options	不支持
CREATE SERVER ROLE	不支持
CURSOR parameters	不支持
Case-sensitive collation	不支持
Column attribute	不支持
Compound operator containing whitespace	不支持
Cross-database reference	不支持
Cursor options	不支持
DATABASEPROPERTYEX	支持
DATEADD	支持
DATEDIFF	支持
DATENAME	支持
DATEPART	支持
DB role options	不支持
DB roles	不支持
DBA statements	不支持
DBCC statements	不支持
DELETE	不支持
DENY	不支持

功能	支持
DESC constraint	不支持
DISABLE TRIGGER	不支持
DML Table Source	不支持
DROP INDEX	不支持
DROP multiple objects	不支持
Datatypes	不支持
Double-quoted string	不支持
Dynamic SQL	不支持
ENABLE TRIGGER	不支持
EXECUTE AS	不支持
EXECUTE SQL function	不支持
Execute procedure options	不支持
Execute string options	不支持
FETCH cursor	不支持
FOR REPLICATION	不支持
Features in computed columns	不支持
Function options	不支持
GLOBAL cursor	不支持
GRANT	支持
GROUP BY ROLLUP/CUBE (old syntax)	不支持
Geospatial features	不支持
Global Temporary Tables	不支持
HASHBYTES	支持
HIERARCHYID features	不支持
IGNORE_DUP_KEY index	不支持
INDEXKEY_PROPERTY	支持
INDEXPROPERTY	支持

功能	支持
INFORMATION_SCHEMA	不支持
INSERT	不支持
INSERT BULK	不支持
Index attribute	不支持
Index options	不支持
Indexed view	不支持
Inline index	不支持
Instead-Of Trigger	不支持
JSON features	不支持
Join hint	不支持
LIKE ‘[…]’	不支持
LOGINPROPERTY	支持
Line continuation character	不支持
Login options	不支持
MERGE	不支持
Materialized view	不支持
Maximum columns per index	支持
Maximum identifier length	支持
Maximum parameters per function	支持
Maximum parameters per procedure	支持
Maximum precision IDENTITY column	支持
Miscellaneous objects	不支持
NEXT VALUE FOR	不支持
NONCLUSTERED HASH index	不支持
NOT FOR REPLICATION	不支持
Non-PERSISTED computed columns	不支持
Nullable column	不支持

功能	支持
Numeric assignment to datetime variable/parameter/column	不支持
Numeric representation of datetime	不支持
OBJECTPROPERTY	支持
OBJECTPROPERTYEX	支持
ODBC Outer Join	不支持
ODBC literal	不支持
ODBC scalar function	不支持
OPEN KEY	不支持
Parameter value DEFAULT	不支持
Partitioning	不支持
Procedure options	不支持
Procedure versioning (declaration)	不支持
Procedure versioning (execution)	不支持
Query hint	不支持
READTEXT	不支持
REVOKE	支持
Regular variable named @@v	不支持
Remote object reference	不支持
SCHEMA options	不支持
SCHEMA_ID with N arguments	支持
SCHEMA_NAME with N arguments	支持
SECURITY DEFINER transaction mgmt	不支持
SELECT TOP PERCENT	不支持
SELECT TOP WITH TIES	不支持
SELECT TOP in Table-Valued Function	不支持
SELECT TOP without ORDER BY	不支持
SELECT..PIVOT	不支持

功能	支持
SELECT..UNPIVOT	不支持
SEQUENCE options	不支持
SERVERPROPERTY	支持
SET ANSI_NULL_DFLT_OFF	支持
SET ANSI_NULL_DFLT_ON	支持
SET ANSI_PADDING	支持
SET ANSI_WARNINGS	支持
SET ARITHABORT	支持
SET ARITHIGNORE	支持
SET CONTEXT_INFO	不支持
SET CURSOR_CLOSE_ON_COMMIT	支持
SET DATEFORMAT	不支持
SET DEADLOCK_PRIORITY	不支持
SET FMTONLY	不支持
SET LANGUAGE	支持
SET NOEXEC	支持
SET NUMERIC_ROUNDABORT	支持
SET OFFSETS	不支持
SET QUERY_GOVERNOR_COST_LIMIT	不支持
SET QUOTED_IDENTIFIER in batch	不支持
SET ROWCOUNT	不支持
SET SHOWPLAN_TEXT	不支持
SET SHOWPLAN_XML	不支持
SET TEXTSIZE	不支持
SET TRANSACTION ISOLATION LEVEL	支持
SETUSER	不支持
SQL graph	不支持

功能	支持
SQL_VARIANT_PROPERTY	支持
STRING_AGG() WITHIN GROUP	不支持
Scalar UDF in table DDL	不支持
Server role options	不支持
Service Broker	不支持
Special characters in identifier	不支持
Special characters in parameter	不支持
Special column names	不支持
Syntax Issues	不支持
T-SQL Outer Join operator	不支持
TRIGGER_NESTLEVEL with N arguments	支持
TYPEPROPERTY	支持
Table hint	不支持
Table value constructor	不支持
Temporal table	不支持
Temporary procedures	不支持
Traceflags	不支持
Transactions	不支持
Trigger options	不支持
UPDATE	不支持
UPDATE STATISTICS	不支持
UPDATETEXT	不支持
Unquoted string	不支持
User options	不支持
Variable aggregates across rows	不支持
Variable assignment dependency	不支持
Variable procedure name	不支持

功能	支持
View options	不支持
WAITFOR	不支持
WRITETEXT	不支持
XML features	不支持
expression AT TIME ZONE	不支持

3.2 内置函数

支持	不支持
ABS	ASSEMBLYPROPERTY
ACOS	ASYMKEYPROPERTY
ASCII	ASYMKEY_ID
ASIN	ATN2
ATAN	BINARY_CHECKSUM
APPLOCK_MODE	CAST
APPLOCK_TEST	CERTENCODED
CAST	CERTPROPERTY
CEILING	CERT_ID
CHAR	CHOOSE
CHARINDEX	COALESCE
CHECKSUM	COLLATIONPROPERTY
COALESCE	COLUMNPROPERTY
CONCAT	COLUMNS_UPDATED
CONVERT	COL_LENGTH
COS	COL_NAME
COT	COMPRESS
CURRENT_TIMESTAMP	CONTAINS
CURRENT_USER	CONTAINSTABLE

支持	不支持
USER	CONTEXT_INFO
DATALENGTH	CONVERT
DATEADD	CURRENT_REQUEST_ID
DATEDIFF	CURRENT_TIMEZONE
DATENAME	CURRENT_TIMEZONE_ID
DATEPART	CURRENT_TRANSACTION_ID
DAY	DATABASEPROPERTY
DB_ID	DATABASE_PRINCIPAL_ID
DB_NAME	DATEDIFF_BIG
SUSER_ID	DATETIME2FROMPARTS
SUSER_NAME	DATETIMEOFFSETFROMPARTS
USER_ID	DECOMPRESS
USER_NAME	DIFFERENCE
DEFAULT_DOMAIN	EOMONTH
DEGREES	EVENTDATA
EXP	FILEGROUPPROPERTY
ERROR_LINE	FILEGROUP_ID
ERROR_MESSAGE	FILEGROUP_NAME
ERROR_NUMBER	FILEPROPERTY
ERROR_PROCEDURE	FILEPROPERTYEX
ERROR_SEVERITY	FILE_ID
ERROR_STATE	FILE_IDEX
FLOOR	FILE_NAME
GETDATE	FN_GET_SQL
GETUTCDATE	FORMAT
IIF	FORMATMESSAGE
ISDATE	GETANSINULL

支持	不支持
ISNULL	GET_FILESTREAM_TRANSACTION_CONTEXT
ISNUMERIC	HASHBYTES
LEFT	HOST_ID
LEN	HOST_NAME
LOG	IDENTITY
LOG10	IIF
LOWER	INDEXKEY_PROPERTY
LTRIM	INDEXPROPERTY
MONTH	INDEX_COL
NCHAR	IS_OBJECTSIGNED
NEWID	JSON
NULLIF	JSON_MODIFY
OBJECT_ID	KEY_GUID
OBJECT_NAME	KEY_ID
PATINDEX	KEY_NAME
PI	LOGINPROPERTY
POWER	MIN_ACTIVE_ROWVERSION
RADIANS	NEWSEQUENTIALID
RAND	NULLIF
REPLACE	OBJECTPROPERTY
REPLICATE	OBJECTPROPERTYEX
REVERSE	OBJECT_DEFINITION
RIGHT	OBJECT_SCHEMA_NAME
ROUND	OPENDATASOURCE
RTRIM	OPENJSON
SCOPE_IDENTITY	OPENQUERY
IDENT_CURRENT	OPENROWSET

支持	不支持
IDENT_INCR	OPENXML
IDENT_SEED	ORIGINAL_DB_NAME
SCHEMA_ID	PARSE
SCHEMA_NAME	PARSENAME
SERVERPROPERTY	PARTITION
SIGN	PERMISSIONS
SIN	PUBLISHINGSERVERNAME
SPACE	REVERSE
SQRT	ROWCOUNT_BIG
STRING_AGG	SESSION_CONTEXT
STRING_SPLIT	SIGNBYASYMKEY
STUFF	SIGNBYCERT
SUBSTRING	SMALLDATETIMEFROMPARTS
SYSDATETIME	SOUNDEX
SYSDATETIMEOFFSET	STATS_DATE
SYSUTCDATETIME	STR
TAN	SID_BINARY
TRIM	SWITCHOFFSET
TRY_CAST	SYMKEYPROPERTY
TRY_PARSE	SYSTEM_USER
UPPER	TERTIARY_WEIGHTS
YEAR	TEXTPTR
QUOTENAME	TEXTVALID
XACT_STATE	TIMEFROMPARTS
STRING_ESCAPE	TODATETIMEOFFSET
DATABASEPROPERTYEX	TRANSLATE
CONNECTIONPROPERTY	TRY_CONVERT

支持	不支持
SQL_VARIANT_PROPERTY	TRY_PARSE
SESSIONPROPERTY	TYPEPROPERTY
COLLATIONPROPERTY	TYPE_ID
	TYPE_NAME
UNICODE	UPDATE
	VERIFYSIGNEDBYASMKEY
TRIGGER_NESTLEVEL() 支持，但仅不带参数。	VERIFYSIGNEDBYCERT
CONCAT_WS	VERIGYSIGNEDBYCERT
DATEFROMPARTS	CHANGETABLE
DATETIMEFROMPARTS	CHANGE_TRACKING_MIN_VALID_VERSION
ORIGINAL_LOGIN	CHANGE_TRACKING_CURRENT_VERSION
SESSION_USER	CHANGE_TRACKING_IS_COLUMN_IN_MASK
SQUARE	CHANGE_TRACKING_CONTEXT
CHOOSE	DEFAULT_DOMAIN
TRIGGER_NESTLEVEL	FILETABLEROOTPATH
CURSOR_STATUS	GETFILENAMESPACEPATH
COLUMNS_UPDATED()	GETPATHLOCATOR
UPDATE()	PATHNAME
FULLTEXTSERVICEPROPERTY()	CONTAINSTABLE
ISJSON()	FREETEXT
JSON_QUERY()	FREETEXTTABLE
JSON_VALUE()	SEMANTICKEYPHRASETABLE
HAS_DBACCESS()	SEMANTICSIMILARITYDETAILSTABLE
SUSER_SID()	SEMANTICSIMILARITYTABLE
SUSER_SNAME()	FULLTEXTCATALOGPROPERTY
IS_SRVROLEMEMBER()	FULLTEXTSERVICEPROPERTY
IS_MEMBER()	ENCRYPTBYASYMKEY

支持	不支持
IS_ROLEMEMBER()	ENCRYPTBYCERT
HAS_PERMS_BY_NAME()	ENCRYPTBYKEY
	ENCRYPTBYPASSPHRASE
	PWDCOMPARE
	PWDENCRYPT
	DECRYPTBYASYMKEY
	DECRYPTBYCERT
	DECRYPTBYKEY
	DECRYPTBYKEYAUTOASYMKEY
	DECRYPTBYKEYAUTOCERT
	DECRYPTBYPASSPHRASE
	DECRYPTBYKEYAUTOCERT
	PREDICT

3.3 支持的SQL Server错误码

错误码	含义
0	操作成功完成。
102	‘%.*ls’ 附近有语法错误。
132	标签 ‘%.*ls’ 已声明。标签名称在查询批次或存储过程内部必须唯一。
133	GOTO 语句引用了标签 ‘%.*ls’，但该标签尚未声明。
134	变量名 ‘%.*ls’ 已声明。变量名在查询批次或存储过程内部必须唯一。
135	不能在 WHILE 语句的作用域之外使用 BREAK 语句。
136	不能在 WHILE 语句的作用域之外使用 CONTINUE 语句。
141	向变量赋值的 SELECT 语句不能与数据检索操作结合使用。
142	约束 ‘%ls’ 的定义中有语法错误。
153	在 %ls 语句中选项 %.*ls 的用法无效。
155	“%.*ls”不是可以识别的 %ls 选项。

错误码	含义
180	此 %ls 语句中参数太多。最多允许 %d 个参数。
217	超出了存储过程、函数、触发器或视图的最大嵌套层数(最大层数为 %d)。
219	类型 '%.*ls' 已存在，或者您没有创建它的权限。
220	发生数据类型 %ls 的算术溢出错误，值 = %ld。
232	类型 %ls 发生算术溢出错误，值 = %f。
266	EXECUTE 后的事务计数指示 BEGIN 和 COMMIT 语句的数目不匹配。上一计数 = %ld，当前计数 = %ld。
289	无法构造数据类型 %ls，某些参数具有无效的值。
293	无法将 char 值转换为 smallmoney。该 char 值的语法有误。
306	不能比较或排序 text、ntext 和 image 数据类型，除非使用 IS NULL 或 LIKE 运算符。
346	因为参数 "%.*ls" 不是表值参数，所以不能将其声明为 READONLY。
352	必须使用 READONLY 选项声明表值参数 "%.*ls"。
477	TABLESAMPLE 子句中的 ROWS 值或 REPEATABLE 种子对表 "%.*ls" 无效。该值或种子必须为整数。
487	为语句 "%.*ls" 指定的选项无效。
506	在 %ls 谓词中指定的转义符 "%.*ls" 无效。
512	子查询返回的值不止一个。当子查询跟随在 =、!=、<、<=、>、>= 之后，或子查询用作表达式时，这种情况是不允许的。
515	不能将值 NULL 插入列 '%.*ls'，表 '%.*ls'；列不允许有 Null 值。%ls 失败。
517	将值添加到 '%ls' 列导致溢出。
545	当 IDENTITY_INSERT 设置为 ON 或某个复制用户向 NOT FOR REPLICATION 标识列中插入内容时，必须为表 '%.*ls' 中的标识列指定显式值。
547	%ls 语句与 %ls 约束 "%.*ls" 冲突。该冲突发生于数据库 "%.*ls"，表 "%.*ls"%ls%.*ls%ls。
550	试图进行的插入或更新已失败，原因是目标视图或者目标视图所跨越的某一视图指定了 WITH CHECK OPTION，而该操作的一个或多个结果行又不符合 CHECK OPTION 约束。
556	由于存储过程改变了目标表的架构，INSERT EXEC 失败。
574	在用户事务内不能使用 %ls 语句。
628	没有活动事务时，不能发出 SAVE TRANSACTION。
1034	语法错误: 在触发器声明中重复指定了操作 "%.*s"。
1049	不允许混合使用新旧语法来指定游标选项。

错误码	含义
1051	存储过程中的游标参数必须以 OUTPUT 和 VARYING 选项来声明，并且必须以 CURSOR VARYING OUTPUT 顺序指定。
1205	事务(进程 ID %d)与另一个进程被死锁在 %.*ls 资源上，并且已被选作死锁牺牲品。请重新运行该事务。
1505	因为发现对象名称 '%.*ls' 和索引名称 '%.*ls' 有重复的键，所以 CREATE UNIQUE INDEX 语句终止。重复的键值为 %ls。
1715	创建外键 '%.*ls' 失败。对于引用计算列 '%.*ls'，只允许使用 NO ACTION 引用更新操作。
1752	表 '%.*ls' 中的列 '%.*ls' 对于创建默认约束无效。
1765	外键 '%.*ls' 创建失败。对于引用计算列 '%.*ls'，只允许使用 NO ACTION 和 CASCADE 引用删除操作。
1768	外键 '%.*ls' 引用的对象 '%.*ls' 不是用户表。
1776	在被引用表 '%.*ls' 中没有与外键 '%.*ls' 中的引用列列表匹配的主键或候选键。
1778	列 '%.*ls.%.*ls' 的数据类型与外键 '%.*ls' 中的引用列 '%.*ls.%.*ls' 的数据类型不同。
1801	数据库 '%.*ls' 已存在。请选择其他数据库名称。
1946	操作失败。索引 '%.*ls' 的索引条目长度为 %d 字节，超出了允许的最大长度 %d 字节。
2627	违反了 %ls 约束 '%.*ls'。不能在对象 '%.*ls' 中插入重复键。
2714	数据库中已存在名为 '%.*ls' 的对象。
2732	错误号 %ld 无效。错误号必须介于 %ld 到 %ld 之间，而且不能是 50000。
2747	RAISERROR 的替代参数太多。替代参数不能超过 %d 个。
2787	指定的格式无效: '%.*ls'。
3609	事务在触发器中结束。批处理已中止。
3616	在触发器执行过程中引发了错误。批处理已中止，用户事务(如果有)已回滚。
3623	出现无效的浮点操作。
3701	无法对 %S_MSG '%.*ls' 执行 %S_MSG，因为它不存在，或者您没有所需的权限。
3723	不允许对索引 '%.*ls' 显式地使用 DROP INDEX。该索引正用于 %ls 约束的强制执行。
3726	无法删除对象 '%.*ls'，因为该对象正由一个 FOREIGN KEY 约束引用。
3728	'%.*ls' 不是约束。
3729	无法对 '%.*ls' 执行 %ls，因为对象 '%.*ls' 正引用它。
3732	无法删除类型 '%.*ls'，因为它正由对象 '%.*ls' 引用。可能还有其他对象在引用此类型。

错误码	含义
3902	COMMIT TRANSACTION 请求没有对应的 BEGIN TRANSACTION。
3903	ROLLBACK TRANSACTION 请求没有对应的 BEGIN TRANSACTION。
3914	数据类型 “%s” 对于事务名称或保存点名称无效。允许使用的数据类型为 char、varchar、nchar、varchar(max)、nvarchar 和 nvarchar(max)。
3930	当前事务无法提交，而且无法支持写入日志文件的操作。请回滚该事务。
4514	CREATE FUNCTION 失败，因为没有为列 %d 指定列名。
4708	无法截断对象 ‘%.*ls’，因为该对象不是表。
4712	无法截断表 ‘%.*ls’，因为该表正由 FOREIGN KEY 约束引用。
4901	ALTER TABLE 只允许添加满足下述条件的列: 列可以包含 Null 值；或者列具有指定的 DEFAULT 定义；或者要添加的列是标识列或时间戳列；或者，如果前几个条件均未满足，则表必须为空以允许添加此列。不能将列 ‘%.*ls’ 添加到非空表 ‘%.*ls’ 中，因为它不满足上述条件。
4920	由于表 ‘%.*ls’ 没有触发器 ‘%.*ls’，ALTER TABLE 失败。
6401	无法回滚 %.*ls。找不到该名称的事务或保存点。
8003	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。此 RPC 请求中提供了过多的参数。最多应为 %d。
8004	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。RPC 名无效。
8007	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。参数 %d (“%.*ls”): 类型为 0x%02X 的大型对象参数的块区格式不正确。
8009	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。参数 %d (“%.*ls”): 数据类型 0x%02X 未知。
8011	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。参数 %d (“%.*ls”): 对于类型特定的元数据，数据类型 0x%02X (sql_variant)的长度无效。
8016	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。参数 %d (“%.*ls”): 数据类型 0x%02X 的数据长度或元数据长度无效。
8018	参数 %d (‘%.*ls’)无效: 数据类型 0x%02X 为不推荐使用的大型对象或 LOB，但却标记为输出参数。不推荐使用的类型不能用作输出参数。请改用当前的大型对象类型。
8023	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。参数 %d (“%.*ls”): 提供的值不是数据类型 %.*ls 的有效实例。请检查源数据中的无效值。例如，小数位数大于精度的数值类型的数据即为无效值。
8028	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。参数 %d (“%.*ls”): 提供的长度对数据类型 %.*ls 无效。请检查源数据中的无效长度。例如，长度为奇数(以字节为单位)的 nchar 类型的数据即为无效长度。

错误码	含义
8029	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。表值参数 %d (“%.*ls”)，行 %I64d，列 %d: 处理表值参数时遇到了数据类型 0x%02X (用户定义表类型)的意外标记。
8031	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。表值参数 %d (“%.*ls”)，行 %I64d，列 %d: 数据类型 0x%02X 的大型对象参数的块区格式不正确。
8032	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。表值参数 %d (“%.*ls”)，行 %I64d，列 %d: 数据类型 0x%02X 未知。
8037	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。表值参数 %d (“%.*ls”)，行 %I64d，列 %d: 数据类型 0x%02X 的数据长度或元数据长度无效。
8043	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。表值参数 %d (“%.*ls”)，行 %I64d，列 %d: 提供的值不是数据类型 %.*ls 的有效实例。请检查源数据中是否存在无效值。例如， 小数位数大于精度的数值类型的数据即为无效值。
8047	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。表值参数 %d (“%.*ls”)，行 %I64d，列 %d: 为数据类型 0x%02X (用户定义的表类型)指定了非零长度的数据库名称。 数据库名称不允许带有表值参数，仅架构名称和类型名称有效。
8050	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。表值参数 %d (“%.*ls”)，行 %I64d，列 %d: 为数据类型 0x%02X (用户定义的表类型)指定了无效的列计数。
8057	表值参数 %d (“%.*ls”)，行 %I64d，列 %d: 数据类型 0x%02X (用户定义的表类型)。 指定的列是计算列或默认列，设置了排序或唯一性。 只能对具有客户端提供的数据的列设置排序和唯一性。
8058	传入的表格格式数据流(TDS)远程过程调用(RPC)协议流不正确。未对参数化字符串的表值参数 %d 定义表类型。
8106	表 ‘%.*ls’ 没有标识属性。无法执行 SET 操作。
8107	表 ‘%.*ls.%.*ls.%.*ls’ 的 IDENTITY_INSERT 已经为 ON。无法为表 ‘%.*ls’ 执行 SET 操作。
8115	将 %ls 转换为数据类型 %ls 时出现算术溢出错误。
8134	遇到以零作除数错误。
8143	多次提供了参数 ‘%.*ls’。
8152	将截断字符串或二进制数据。
8159	‘%.*ls’ 中的列少于列列表中指定的列。
8179	找不到句柄为 %d 的预定义语句。
9441	XML 分析: 行 %d，字符 %d，不正确的 xml 声明语法
9451	XML 分析: 行 %d，字符 %d，不正确的处理指令语法

错误码	含义
9809	从 %s 到 %s 的转换不支持样式 %d。
10610	无法基于表 '%.*ls' 创建筛选的 %S_MSG '%.*ls'，因为筛选表达式中的列 '%.*ls' 为计算列。 请重写该筛选表达式以便其不包括此列。
10727	用户定义的聚合不支持默认参数。
10733	在 UNION、INTERSECT 或 EXCEPT 运算符的任何一侧都不允许嵌套的 INSERT、UPDATE、DELETE 或 MERGE 语句。
10793	索引和 PRIMARY KEY 约束都已以内联方式定义，其中定义了列 '%.*ls'、表 '%.*ls'。 不允许使用列定义内联定义索引和 PRIMARY KEY 约束。
11555	参数 '%.*ls' 已声明为 NOT NULL。 非 NULL 参数仅支持本机编译的模块，但内联表值函数除外。
11700	序列对象 '%.*ls' 的增量不能为零。
11701	序列对象 '%.*ls' 增量的绝对值必须小于或等于序列对象的最小值和最大值之间的差异。
11702	序列对象 '%.*ls' 必须是数据类型 int、bigint、smallint、tinyint 或 decimal 或 numeric，其小数位数为 0，或者基于上述整数数据类型之一的任何用户定义的数据类型。
11703	序列对象 '%.*ls' 的起始值必须介于序列对象的最小值和最大值之间。
11705	序列对象 '%.*ls' 的最小值必须小于其最大值。
11706	序列对象 '%.*ls' 的缓存大小必须大于 0。
11708	为给定数据类型的参数 '%.*ls' 指定了无效值。
11709	CREATE 标准版QUENCE 语句中不能使用“RESTART WITH”参数。
11717	NEXT VALUE FOR 函数不支持默认约束、UPDATE 语句或 MERGE 语句中的 OVER 子句。
16901	%hs：此功能尚未实现。
16902	%ls: 参数 %ls 的值无效。
16903	调用 “%ls” 过程时使用的参数数目不正确。
16915	名为 '%.*ls' 的游标已存在。
16916	名为 '%.*ls' 的游标不存在。
16948	变量 '%.*ls' 不是游标变量，却出现在游标变量应出现的位置。
16950	目前没有为变量 '%.*ls' 分配游标。
18456	用户 '%.*ls'.%.*ls%.*ls%.*ls 登录失败
201	过程或函数 '%.*ls' 需要未提供的参数 '%.*ls'。

错误码	含义
206	操作数类型冲突: %ls 与 %ls 不兼容
2733	%ls 数据类型对返回值无效。
8144	为过程或函数 %.*ls 指定了过多的参数。
8145	%.*ls 不是过程 %.*ls 的参数。
8146	过程 %.*ls 没有参数，但却为该过程提供了参数。
213	列名称或所提供值的数目与表定义不匹配。

3.4 不支持的 SET 语句

SET ANSI_NULL_DFLT_OFF ON

SET ANSI_NULL_DFLT_ON OFF

SET ANSI_PADDING OFF

SET ANSI_WARNINGS OFF

SET ARITHABORT OFF

SET ARITHIGNORE ON

SET CONTEXT_INFO

SET CURSOR_CLOSE_ON_COMMIT ON

SET DATEFORMAT

SET DEADLOCK_PRIORITY

SET FORCEPLAN

SET NO BROWSETABLE

SET NUMERIC_ROUNDABORT ON

SET REMOTE_PROC_TRANSACTIONS

SET ROWCOUNT n WHERE n != 0

SET ROWCOUNT @variable

SET SHOWPLAN_ALL

SET SHOWPLAN_TEXT

SET SHOWPLAN_XML

SET STATISTICS

SET STATISTICS IO

SET STATISTICS PROFILE

SET STATISTICS TIME

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

SET OFFSETS

SETUSER

以下 SET 语句限制适用：

SET LANGUAGE- 除 或 以外的任何值都不支持此语法。englishus_english

以下 SET 语句可用于 UDBTX-TDS 1.2.0 及更高版本：

SET FMTONLY - SET FMTONLY ON/OFF仅适用于查询，所有其他查询均忽略。SELECT *

SET LOCK_TIMEOUT

3.5 系统存储过程

支持	不支持
sp_helpdb	sp_add%
sp_getapplock	sp_adjustpublisheridentityrange
sp_releaseapplock	sp_agent_%
sp_columns	sp_alter_nt_job_mem_configs
sp_columns_100	sp_altermessage
sp_columns_managed	sp_apply_job_to_targets
sp_cursor_list	sp_approlepassword
sp_datatype_info	sp_article%
sp_datatype_info_100	sp_assembl%
sp_describe_CURSOR	sp_attach%

支持	不支持
sp_DESCRIBE_FIRST_RESULT_SET	sp_audit_write
sp_DESCRIBE_UNDECLARED_PARAMETERS	sp_autoadmin_%
sp_executesql	sp_autoindex%
sp_oledb_ro_usrname	sp_autostats
sp_prepare	sp_availability_group_command_internal
sp_tablecollations_100	sp_backup_%
sp_tables	sp_bcp_dbcmptlevel
sp_unprepare	sp_begin_parallel_nested_tran
sp_statistics	sp_bindefault
sp_statistics_100	sp_bindrule
sp_updatestats	sp_bindsession
sp_databases	sp_browsemergesnapshotfolder
sp_cursor	sp_browsereplcmds
sp_cursoropen	sp_browsesnapshotfolder
sp_cursorprepare	sp_build_histogram
sp_cursorexecute	sp_can_tlog_be_applied
sp_cursorprepexec	sp_catalogs%
sp_cursorunprepare	sp_cdc_%
sp_cursorfetch	sp_certify_removable
sp_cursoroption	sp_change%
sp_cursorclose	sp_check%
sp_statistics	sp_clean%
sp_statistics_100	sp_clear_dbmaintplan_by_db
sp_updatestats	sp_cloud_update_blob_tier
sp_pkeys	sp_column%
sp_databases	sp_commit_parallel_nested_tran
sp_cursor	sp_configure%

支持	不支持
sp_cursoropen	sp_constr_col%
sp_cursorprepare	sp_control_dbmasterkey_password
sp_cursorexecute	sp_control_plan_guide
sp_cursorprepexec	sp_convert_jobid_to_char
sp_cursorunprepare	sp_copymergesnapshot
sp_cursorfetch	sp_copysnapshot
sp_cursoroption	sp_copysubscription
sp_cursorclose	sp_create_file_statistics
sp_table_privileges	sp_create_job
sp_column_privileges	sp_create_log_shipping_monitor_account
sp_special_columns	sp_create_plan_guide%
sp_fkeys	sp_create_removable
sp_stored_procedures	sp_createmergepalrole
xp_qv	sp_createorphan
sp_describe_undeclared_parameters	sp_createstats
sp_helpuser	sp_createtranpalrole
sp_sproc_columns	sp_cursor%
sp_sproc_columns_100	sp_cycle_agent_errorlog
sp_helprole	sp_cycle_errorlog
sp_helprolemember	sp_databases
	sp_datatype_info%
	sp_db_ebcdic277_2
	sp_db_increased_partitions
	sp_db_selective_xml_index
	sp_db_vardecimal_storage_format
	sp_dbcmptlevel
	sp_dbfixedrolepermission

支持	不支持
	sp_dbmmonitor%
	sp_dbremove
	sp_ddopen
	sp_defaultdb
	sp_defaultlanguage
	sp_delete%
	sp_denylogin
	sp_depends
	sp_describe_%
	sp_detach_db
	sp_detach_schedule
	sp_diagnostic_showplan_log_dbid
	sp_disableagentoffload
	sp_distcounters
	sp_do_backup
	sp_downloaded_row_limiter
	sp_drop%
	sp_dsninfo
	sp_enable_heterogeneous_subscription
	sp_enable_sql_debug
	sp_enableagentoffload
	sp_enclave_send_keys
	sp_enlist_tsx
	sp_enum%
	sp_enum_login_for_proxy
	sp_enum_proxy_for_subsystem
	sp_enum_sqlagent_subsystems

支持	不支持
	sp_enum_sqlagent_subsystems_internal
	sp_estimate_data_compression_savings
	sp_estimated_rowsize_reduction_for_vardecimal
	sp_execute
	sp_execute_external_script
	sp_execute_remote
	sp_expired_subscription_cleanup
	sp_externalmailqueuelistener
	sp_fido_%
	sp_filestream%
	sp_firstonly_bitmap
	sp_fkeys
	sp_flush%
	sp_force_slog_truncation
	sp_foreign%
	sp_fulltext_%
	sp_generate_agent_parameter
	sp_generate_server_description
	sp_generate_target_server_job_assignment_sql
	sp_generatefilters
	sp_get_backup_diagnostics
	sp_get_chunked_jobstep_params
	sp_get_composite_job_info
	sp_get_database_scoped_credential
	sp_get_distributor
	sp_get_encryption_option
	sp_get_file_statistics_hash

支持	不支持
	sp_get_job_alerts
	sp_get_job_status_mergesubscription_agent
	sp_get_jobstep_db_username
	sp_get_log_shipping_monitor_info
	sp_get_mergepublishedarticleproperties
	sp_get_message_description
	sp_get_oracle_publisher_metadata
	sp_get_proxy_properties
	sp_get_query_template
	sp_get_redirected_publisher
	sp_get_schedule_description
	sp_get_sqlagent_properties
	sp_get_striping_option
	sp_get_traceflag_status_internal
	sp_getagentparameterlist
	sp_getapplock
	sp_getattachmentdata
	sp_getbindtoken
	sp_getdefaultdatatypemapping
	sp_getmergedeletetype
	sp_getprocessorusage
	sp_getpublisherlink
	sp_getqueuedarticlesynctraninfo
	sp_getqueuedrows
	sp_getschemalock
	sp_getsqlqueueversion
	sp_getsubscription_status_hsnapshot

支持	不支持
	sp_getsubscriptiondtspackagename
	sp_gettopologyinfo
	sp_getvolumefreespace
	sp_grant_login_to_proxy
	sp_grant_proxy_to_subsystem
	sp_grant_publication_access
	sp_grantdbaccess
	sp_grantlogin
	sp_help%
	sp_http_%
	sp_identitycolumnforreplication
	sp_ih%
	sp_index%
	sp_internal_alter_nt_job_limits
	sp_invalidate_textptr
	sp_is_makegeneration_needed
	sp_is_sqlagent_starting
	sp_isprohibited
	sp_ivindexhasnullcols
	sp_jobhistory_row_limiter
	sp_kill_filestream_non_transacted_handles
	sp_kill_oldest_transaction_on_secondary
	sp_lightweightmergemetadataretentioncleanup
	sp_link_publication
	sp_linkedservers%
	sp_lock
	sp_log_shipping_%

支持	不支持
	sp_logshippinginstallmetadata
	sp_lookupcustomresolver
	sp_mailitemresultsets
	sp_maintplan_%
	sp_manage_jobs_by_login
	sp_mapdown_bitmap
	sp_markpendingschemachange
	sp_marksubscriptionvalidation
	sp_memory_optimized_cs_migration
	sp_merge%
	sp_migrate_user_to_contained
	sp_monitor
	sp_ms%
	sp_multi_server_job_summary
	sp_new_parallel_nested_tran_id
	sp_notify_operator
	sp_oa%
	sp_objectfilegroup
	sp_oledb%
	sp_orbitmap
	sp_password
	sp_peerconflictdetection_tableaug
	sp_polybase_%
	sp_post_msx_operation
	sp_postagentinfo
	sp_posttracertoken
	sp_prepare

支持	不支持
	sp_prepexec
	sp_prepexecrpc
	sp_primary%
	sp_procedure%
	sp_process_DialogTimer
	sp_processlogshipping%
	sp_processresponse
	sp_procoption
	sp_prop_oledb_provider
	sp_provider_types%
	sp_publication%
	sp_publish%
	sp_purge_jobhistory
	sp_query_store_%
	sp_rbpex_exec_cmd
	sp_rda_%
	sp_read_settings
	sp_readerrorlog
	sp_readrequest
	sp_reassign_proxy
	sp_recompile
	sp_redirect_publisher
	sp_refresh%
	sp_register%
	sp_reinit%
	sp_releaseapplock
	sp_releaseschemalock

支持	不支持
	sp_remote_data_archive_event
	sp_remoteoption
	sp_remove%
	sp_rename
	sp_renamedb
	sp_repl%
	sp_requestpeer%
	sp_reserve_http_namespace
	sp_reset_connection
	sp_reset_session_context
	sp_resetsnapshotdeliveryprogress
	sp_resetstatus
	sp_resign_database
	sp_resolve_logins
	sp_restoredbreplication
	sp_restoremergeidentityrange
	sp_resync%
	sp_revoke%
	sp_rollback_parallel_nested_tran
	sp_runmailquery
	sp_schemafilter
	sp_schemata_rowset
	sp_script%
	sp_sem_add_message
	sp_sem_drop_message
	sp_send_dbmail
	sp_sendmailmessage

支持	不支持
	sp_sendmailqueues
	sp_sequence_get_range
	sp_server_diagnostics
	sp_server_info
	sp_serveroption
	sp_set_db_backup
	sp_set_distributed_query_context
	sp_set_instance_backup
	sp_set_local_time
	sp_set_parameter
	sp_set_session_context
	sp_set_session_resource_group
	sp_set_sqlagent_properties
	sp_setapprole
	sp_setautosapasswordanddisable
	sp_setdefaultdatatypemapping
	sp_setnetname
	sp_setobdcertificate
	sp_setoraclepackageversion
	sp_setreplfailovermode
	sp_setsubscriptionxactseqno
	sp_settriggerorder
	sp_setuserbylogin
	sp_show_file_statistics
	sp_showcolv
	sp_showinitialmemo_xml
	sp_showlineage

支持	不支持
	sp_showmemo_xml
	sp_showpendingchanges
	sp_showrowreplicainfo
	sp_sm_detach
	sp_spaceused%
	sp_sparse_columns%
	sp_special_columns%
	sp_sproc_columns%
	sp_sqlagent%
	sp_sqlexec
	sp_srvrolepermission
	sp_ssis_%
	sp_start%
	sp_statistics%
	sp_stop%
	sp_stored_procedures
	sp_subscribe
	sp_subscription%
	sp_syscollector_%
	sp_sysdac_%
	sp_sysmail_activate
	sp_sysmanagement_%
	sp_syspolicy%
	sp_syspolicy_%
	sp_sysutility_%
	sp_table%
	sp_target_server_summary

支持	不支持
	sp_testlinkedserver
	sp_trace_%
	sp_try_set_session_context
	sp_unbindefault
	sp_unbindrule
	sp_uniquetaskname
	sp_unprepare
	sp_unregister%
	sp_unsetapprole
	sp_unsubscribe
	sp_update_agent_profile
	sp_update_alert
	sp_update_category
	sp_update_job
	sp_update_jobschedule
	sp_update_jobstep
	sp_update_log_shipping_monitor_info
	sp_update_notification
	sp_update_operator
	sp_update_proxy
	sp_update_replication_job_parameter
	sp_update_schedule
	sp_update_targetservergroup
	sp_update_user_instance
	sp_updateextendedproperty
	sp_updatestats
	sp_upgrade_log_shipping

支持	不支持
	sp_user_counter%
	sp_usertypes%
	sp_validate%
	sp_validate_user
	sp_verify_alert
	sp_verify_category
	sp_verify_category_identifiers
	sp_verify_credential_identifiers
	sp_verify_job
	sp_verify_job_date
	sp_verify_job_identifiers
	sp_verify_job_time
	sp_verify_jobproc_caller
	sp_verify_jobstep
	sp_verify_login_identifiers
	sp_verify_notification
	sp_verify_operator
	sp_verify_operator_identifiers
	sp_verify_performance_condition
	sp_verify_proxy
	sp_verify_proxy_identifiers
	sp_verify_proxy_permissions
	sp_verify_schedule
	sp_verify_schedule_identifiers
	sp_verify_subsystem
	sp_verify_subsystem_identifiers
	sp_verify_subsystems

支持	不支持
	sp_verifypublisher
	sp_views%
	sp_vupgrade%
	sp_who
	sp_who2
	sp_write_sysjobstep_log
	sp_xa_%
	sp_xml_preparedocument
	sp_xml_removedocument
	sp_xml_schema_rowset%
	sp_xp_cmdshell_proxy_account
	sp_xtp_%
	sp_sqljdbc_%
	xp_%

3.6 系统目录视图

支持	不支持
databases	sysallocunits
sysdatabases	sysaltfiles
server_principals	sysasymkeys
objects	sysaudacts
sysobjects	sysbinobjs
all_objects	sysbinsubobjs
schemas	sysbrickfiles
tables	syscacheobjects
indexes	syscerts
views	syscharsets

支持	不支持
all_views	syschildinsts
procedures	sysclones
extended_properties	sysclsobjs
columns	syscolpars
all_columns	syscolumns
syscharsets	syscomments
assemblies	syscompfragments
assembly_types	sysconfigures
sysforeignkeys	sysconstraints
foreign_key_columns	sysconvgroup
foreign_keys	syscscolsegments
key_constraints	syscscontainers
sysindexes	syscsdictionaries
syscolumns	syscsrowgroups
system_objects	syscurconfigs
identity_columns	syscursorcolumns
types	syscursorrefs
check_constraints	syscursors
computed_columns	syscursortables
default_constraints	sysdatabases
index_columns	sysdbfiles
sql_modules	sysdbfrag
sys.dm_os_host_info	sysdbreg
sys.dm_exec_sessions	sysdepends
sys.dm_exec_connections	sysdercv
sys.endpoints	sysdesend
sys.table_types	sysdevices

支持	不支持
sys.database_principals	sysendpts
sys.sysprocesses	sysextendedrecoveryforks
sys.sysconfigures- 目前提供单个只读配置设置。	sysextfileformats
sys.syscurconfigs- 目前提供单个只读配置设置。	sysextsources
sys.configurations- 目前提供单个只读配置设置。	sysexttables
sys.syslanguages	sysfgfrag
sys.indexes	sysfilegroups
sys.all_views	sysfiles
sys.database_files	sysfiles1
sys.sql_modules	sysfoqueues
sys.system_sql_modules	sysforeignkeys
sys.all_sql_modules	sysfos
sys.xml_schema_collections	sysftinds
sys.dm_hadr_database_replica_states	sysftproperties
sys.data_spaces	sysftsemanticsdb
sys.database_mirroring	sysftstops
sys.database_role_members	sysfulltextcatalogs
	sysguidrefs
	sysidxstats
	sysindexes
	sysindexkeys
	sysiscols
	syslanguages
	syslnklgns
	syslockinfo
	syslogins
	syslogshippers

支持	不支持
	sysmatrixageforget
	sysmatrixages
	sysmatrixbricks
	sysmatrixconfig
	sysmatrixmanagers
	sysmembers
	sysmessages
	sysmultiobjrefs
	sysmultiobjvalues
	sysnsobjs
	sysobjects
	sysobjkeycrypts
	sysobjvalues
	sysoledbusers
	sysopentapes
	sysowners
	sysperfinfo
	syspermissions
	sysphfg
	syspriorities
	sysprivs
	sysprocesses
	sysprotects
	syspru
	sysprufiles
	sysqnames
	sysreferences

支持	不支持
	sysremotelogins
	sysremsvcbinds
	sysrmtlgns
	sysrowsetrefs
	sysrowsets
	sysrscols
	sysrts
	sysscalartypes
	sysschobjs
	sysseobjvalues
	sysservers
	syssingleobjrefs
	syssoftobjrefs
	syssqlguides
	system_columns
	system_components_surface_area_configuration
	system_internals_allocation_units
	system_internals_partition_columns
	system_internals_partitions
	system_objects
	system_parameters
	system_sql_modules
	system_views
	systypedsubobjs
	systypes
	sysusermsgs
	sysusers

支持	不支持
	syswebmethods
	sysxlgns
	sysxmitbody
	sysxmitqueue
	sysxmlcomponent
	sysxmlfacet
	sysxmlplacement
	sysxprops
	sysxsrvs
	all_columns
	all_objects
	all_parameters
	all_sql_modules
	all_views
	allocation_units
	assemblies
	assembly_files
	assembly_modules
	assembly_references
	assembly_types
	asymmetric_keys
	availability_databases_cluster
	availability_group_listener_ip_addresses
	availability_group_listeners
	availability_groups
	availability_groups_cluster
	availability_read_only_routing_lists

支持	不支持
	availability_replicas
	backup_devices
	certificates
	change_tracking_databases
	change_tracking_tables
	check_constraints
	column_encryption_key_values
	column_encryption_keys
	column_master_keys
	column_store_dictionaries
	column_store_row_groups
	column_store_segments
	column_type_usages
	column_xml_schema_collection_usages
	columns
	computed_columns
	configurations
	conversation_endpoints
	conversation_groups
	conversation_priorities
	credentials
	crypt_properties
	cryptographic_providers
	data_spaces
	database_audit_specification_details
	database_audit_specifications
	database_automatic_tuning_mode

支持	不支持
	database_automatic_tuning_options
	database_credentials
	database_files
	database_filestream_options
	database_mirroring
	database_mirroring_endpoints
	database_mirroring_witnesses
	database_permissions
	database_principals
	database_query_store_options
	database_recovery_status
	database_role_members
	database_scoped_configurations
	database_scoped_credentials
	databases
	default_constraints
	destination_data_spaces
	dm_audit_actions
	dm_audit_class_type_map
	dm_broker_activated_tasks
	dm_broker_connections
	dm_broker_forwarded_messages
	dm_broker_queue_monitors
	dm_cache_hit_stats
	dm_cache_size
	dm_cache_stats
	dm_cdc_errors

支持	不支持
	dm_cdc_log_scan_sessions
	dm_clr_appdomains
	dm_clr_loaded_assemblies
	dm_clr_properties
	dm_clr_tasks
	dm_cluster_endpoints
	dm_column_encryption_enclave
	dm_column_encryption_enclave_operation_stats
	dm_column_store_object_pool
	dm_cryptographic_provider_properties
	dm_database_encryption_keys
	dm_db_column_store_row_group_operational_stats
	dm_db_column_store_row_group_physical_stats
	dm_db_data_pool_nodes
	dm_db_data_pools
	dm_db_external_language_stats
	dm_db_external_script_execution_stats
	dm_db_file_space_usage
	dm_db_fts_index_physical_stats
	dm_db_index_usage_stats
	dm_db_log_space_usage
	dm_db_mirroring_auto_page_repair
	dm_db_mirroring_connections
	dm_db_mirroring_past_actions
	dm_db_missing_index_details
	dm_db_missing_index_group_stats
	dm_db_missing_index_group_stats_query

支持	不支持
	dm_db_missing_index_groups
	dm_db_partition_stats
	dm_db_persisted_sku_features
	dm_db_rda_migration_status
	dm_db_rda_schema_update_status
	dm_db_script_level
	dm_db_session_space_usage
	dm_db_storage_pool_nodes
	dm_db_storage_pools
	dm_db_task_space_usage
	dm_db_tuning_recommendations
	dm_db_uncontained_entities
	dm_db_xtp_checkpoint_files
	dm_db_xtp_checkpoint_internals
	dm_db_xtp_checkpoint_stats
	dm_db_xtp_gc_cycle_stats
	dm_db_xtp_hash_index_stats
	dm_db_xtp_index_stats
	dm_db_xtp_memory_consumers
	dm_db_xtp_nonclustered_index_stats
	dm_db_xtp_object_stats
	dm_db_xtp_table_memory_stats
	dm_db_xtp_transactions
	dm_distributed_exchange_stats
	dm_exec_background_job_queue
	dm_exec_background_job_queue_stats
	dm_exec_cached_plans

支持	不支持
	dm_exec_compute_node_errors
	dm_exec_compute_node_status
	dm_exec_compute_nodes
	dm_exec_compute_pools
	dm_exec_connections
	dm_exec_distributed_request_steps
	dm_exec_distributed_requests
	dm_exec_distributed_sql_requests
	dm_exec_dms_services
	dm_exec_dms_workers
	dm_exec_external_operations
	dm_exec_external_work
	dm_exec_function_stats
	dm_exec_procedure_stats
	dm_exec_query_memory_grants
	dm_exec_query_optimizer_info
	dm_exec_query_optimizer_memory_gateways
	dm_exec_query_parallel_workers
	dm_exec_query_profiles
	dm_exec_query_resource_semaphores
	dm_exec_query_stats
	dm_exec_query_transformation_stats
	dm_exec_requests
	dm_exec_session_wait_stats
	dm_exec_sessions
	dm_exec_trigger_stats
	dm_exec_valid_use_hints

支持	不支持
	dm_external_authentication
	dm_external_script_execution_stats
	dm_external_script_requests
	dm_external_script_resource_usage_stats
	dm_filestream_file_io_handles
	dm_filestream_file_io_requests
	dm_filestream_non_transacted_handles
	dm_fts_active_catalogs
	dm_fts_fdhosts
	dm_fts_index_population
	dm_fts_memory_buffers
	dm_fts_memory_pools
	dm_fts_outstanding_batches
	dm_fts_population_ranges
	dm_fts_semantic_similarity_population
	dm_hadr_ag_threads
	dm_hadr_auto_page_repair
	dm_hadr_automatic_seeding
	dm_hadr_availability_group_states
	dm_hadr_availability_replica_cluster_nodes
	dm_hadr_availability_replica_cluster_states
	dm_hadr_availability_replica_states
	dm_hadr_cluster
	dm_hadr_cluster_members
	dm_hadr_cluster_networks
	dm_hadr_database_replica_cluster_states
	dm_hadr_database_replica_states

支持	不支持
	dm_hadr_db_threads
	dm_hadr_instance_node_map
	dm_hadr_name_id_map
	dm_hadr_physical_seeding_stats
	dm_hpc_device_stats
	dm_hpc_thread_proxy_stats
	dm_io_backup_tapes
	dm_io_cluster_shared_drives
	dm_io_cluster_valid_path_names
	dm_io_pending_io_requests
	dm_logpool_hashentries
	dm_logpool_stats
	dm_os_buffer_descriptors
	dm_os_buffer_pool_extension_configuration
	dm_os_child_instances
	dm_os_cluster_nodes
	dm_os_cluster_properties
	dm_os_dispatcher_pools
	dm_os_dispatchers
	dm_os_enumerate_fixed_drives
	dm_os_host_info
	dm_os_hosts
	dm_os_job_object
	dm_os_latch_stats
	dm_os_loaded_modules
	dm_os_memory_allocations
	dm_os_memory_broker_clerks

支持	不支持
	dm_os_memory_brokers
	dm_os_memory_cache_clock_hands
	dm_os_memory_cache_counters
	dm_os_memory_cache_entries
	dm_os_memory_cache_hash_tables
	dm_os_memory_clerks
	dm_os_memory_node_access_stats
	dm_os_memory_nodes
	dm_os_memory_objects
	dm_os_memory_pools
	dm_os_nodes
	dm_os_performance_counters
	dm_os_process_memory
	dm_os_ring_buffers
	dm_os_schedulers
	dm_os_server_diagnostics_log_configurations
	dm_os_spinlock_stats
	dm_os_stacks
	dm_os_sublatches
	dm_os_sys_info
	dm_os_sys_memory
	dm_os_tasks
	dm_os_threads
	dm_os_virtual_address_dump
	dm_os_wait_stats
	dm_os_waiting_tasks
	dm_os_windows_info

支持	不支持
	dm_os_worker_local_storage
	dm_os_workers
	dm_pal_cpu_stats
	dm_pal_disk_stats
	dm_pal_net_stats
	dm_pal_processes
	dm_pal_spinlock_stats
	dm_pal_vm_stats
	dm_pal_wait_stats
	dm_qn_subscriptions
	dm_repl_articles
	dm_repl_schemas
	dm_repl_tranhash
	dm_repl_traninfo
	dm_resource_governor_configuration
	dm_resource_governor_external_resource_pool_affinity
	dm_resource_governor_external_resource_pools
	dm_resource_governor_resource_pool_affinity
	dm_resource_governor_resource_pool_volumes
	dm_resource_governor_resource_pools
	dm_resource_governor_workload_groups
	dm_server_audit_status
	dm_server_memory_dumps
	dm_server_registry
	dm_server_services
	dm_tcp_listener_states
	dm_tran_aborted_transactions

支持	不支持
	dm_tran_active_snapshot_database_transactions
	dm_tran_active_transactions
	dm_tran_commit_table
	dm_tran_current_snapshot
	dm_tran_current_transaction
	dm_tran_database_transactions
	dm_tran_global_recovery_transactions
	dm_tran_global_transactions
	dm_tran_global_transactions_enlistments
	dm_tran_global_transactions_log
	dm_tran_locks
	dm_tran_persistent_version_store
	dm_tran_persistent_version_store_stats
	dm_tran_session_transactions
	dm_tran_top_version_generators
	dm_tran_transactions_snapshot
	dm_tran_version_store
	dm_tran_version_store_space_usage
	dm_xe_map_values
	dm_xe_object_columns
	dm_xe_objects
	dm_xe_packages
	dm_xe_session_event_actions
	dm_xe_session_events
	dm_xe_session_object_columns
	dm_xe_session_targets
	dm_xe_sessions

支持	不支持
	dm_xtp_gc_queue_stats
	dm_xtp_gc_stats
	dm_xtp_system_memory_consumers
	dm_xtp_threads
	dm_xtp_transaction_recent_rows
	dm_xtp_transaction_stats
	edge_constraint_clauses
	edge_constraints
	endpoint_webmethods
	endpoints
	event_notification_event_types
	event_notifications
	events
	extended_procedures
	extended_properties
	external_data_sources
	external_file_formats
	external_language_files
	external_languages
	external_libraries
	external_libraries_installed
	external_library_files
	external_library_setup_errors
	external_table_columns
	external_tables
	filegroups
	filetable_system_defined_objects

支持	不支持
	filetables
	foreign_key_columns
	foreign_keys
	fulltext_catalogs
	fulltext_document_types
	fulltext_index_catalog_usages
	fulltext_index_columns
	fulltext_index_fragments
	fulltext_indexes
	fulltext_languages
	fulltext_semantic_language_statistics_database
	fulltext_semantic_languages
	fulltext_stoplists
	fulltext_stopwords
	fulltext_system_stopwords
	function_order_columns
	hash_indexes
	http_endpoints
	identity_columns
	index_columns
	index_resumable_operations
	indexes
	internal_partitions
	internal_tables
	key_constraints
	key_encryptions
	linked_logins

支持	不支持
	login_token
	masked_columns
	master_files
	master_key_passwords
	memory_optimized_tables_internal_attributes
	message_type_xml_schema_collection_usages
	messages
	module_assembly_usages
	numbered_procedure_parameters
	numbered_procedures
	objects
	openkeys
	parameter_type_usages
	parameter_xml_schema_collection_usages
	parameters
	partition_functions
	partition_parameters
	partition_range_values
	partition_schemes
	partitions
	periods
	plan_guides
	procedures
	query_context_settings
	query_store_plan
	query_store_query
	query_store_query_text

支持	不支持
	query_store_runtime_stats
	query_store_runtime_stats_interval
	query_store_wait_stats
	registered_search_properties
	registered_search_property_lists
	remote_data_archive_databases
	remote_data_archive_tables
	remote_logins
	remote_service_bindings
	resource_governor_configuration
	resource_governor_external_resource_pool_affinity
	resource_governor_external_resource_pools
	resource_governor_resource_pool_affinity
	resource_governor_resource_pools
	resource_governor_workload_groups
	routes
	schemas
	securable_classes
	security_policies
	security_predicates
	selective_xml_index_namespaces
	selective_xml_index_paths
	sensitivity_classifications
	sequences
	server_assembly_modules
	server_audit_specification_details
	server_audit_specifications

支持	不支持
	server_audits
	server_event_notifications
	server_event_session_actions
	server_event_session_events
	server_event_session_fields
	server_event_session_targets
	server_event_sessions
	server_events
	server_file_audits
	server_memory_optimized_hybrid_buffer_pool_configuration
	server_permissions
	server_principal_credentials
	server_principals
	server_role_members
	server_sql_modules
	server_trigger_events
	server_triggers
	servers
	service_broker_endpoints
	service_contract_message_usages
	service_contract_usages
	service_contracts
	service_message_types
	service_queue_usages
	service_queues
	services
	soap_endpoints

支持	不支持
	spatial_index_tessellations
	spatial_indexes
	spatial_reference_systems
	sql_dependencies
	sql_expression_dependencies
	sql_logins
	sql_modules
	stats
	stats_columns
	symmetric_keys
	synonyms
	table_types
	tables
	tcp_endpoints
	time_zone_info
	trace_categories
	trace_columns
	trace_event_bindings
	trace_events
	trace_subclass_values
	traces
	transmission_queue
	trigger_event_types
	trigger_events
	triggers
	trusted_assemblies
	type_assembly_usages

支持	不支持
	types
	user_token
	via_endpoints
	views
	xml_indexes
	xml_schema_attributes
	xml_schema_collections
	xml_schema_component_placements
	xml_schema_components
	xml_schema_elements
	xml_schema_facets
	xml_schema_model_groups
	xml_schema_namespaces
	xml_schema_types
	xml_schema_wildcard_namespaces
	xml_schema_wildcards
	syspolicy_configuration
	syspolicy_system_health_state

3.7 系统函数

支持	不支持
fn_helpcollations	fn_enumcurrentprincipals
fn_listextendedproperty	fn_getcurrentprincipal
	fn_getrowsetidfromrowdump
	fn_isbitsetinbitmask
	fn_msdayasnumber
	fn_msgeneration_downloadonly

支持	不支持
	fn_msget_dynamic_filter_login
	fn_msorbitmaps
	fn_msrepl_getsrvidfromdistdb
	fn_msrepl_map_resolver_clsid
	fn_mstestbit
	fn_msvector_downloadonly
	fn_msxe_read_event_stream
	fn_mapschematype
	fn_pagerescracker
	fn_physloccracker
	fn_physlocformatter
	fn_rowdumpcracker
	fn_builtin_permissions
	fn_ccolventries_80
	fn_cdc_check_parameters
	fn_cdc_decrement_lsn
	fn_cdc_get_column_ordinal
	fn_cdc_get_max_lsn
	fn_cdc_get_min_lsn
	fn_cdc_has_column_changed
	fn_cdc_hexstrtobin
	fn_cdc_increment_lsn
	fn_cdc_is_bit_set
	fn_cdc_map_lsn_to_time
	fn_cdc_map_time_to_lsn
	fn_check_object_signatures
	fn_column_store_row_groups

支持	不支持
	fn_db_backup_file_snapshots
	fn_dblog
	fn_dblog_xtp
	fn_dbslog
	fn_dump_dblog
	fn_dump_dblog_xtp
	fn_fiscoltracked
	fn_full_dblog
	fn_get_audit_file
	fn_get_sql
	fn_getproviderstring
	fn_hadr_backup_is_preferred_replica
	fn_hadr_distributed_ag_database_replica
	fn_hadr_distributed_ag_replica
	fn_hadr_is_primary_replica
	fn_hadr_is_same_replica
	fn_helpcollations
	fn_helpdatatypemap
	fn_isrolemember
	fn_listextendedproperty
	fn_my_permissions
	fn_numberof1inbinaryafterloc
	fn_numberof1invarbinary
	fn_repl_hash_binary
	fn_repladjustcolumnmap
	fn_repldecryptver4
	fn_replformatdatetime

支持	不支持
	fn_replgetcolidfrombitmap
	fn_replgetparsedddlcmd
	fn_replp2pversiontotranid
	fn_replreplacesinglequote
	fn_replreplacesinglequoteplusprotectstring
	fn_repluniquename
	fn_replvarbintoint
	fn_servershareddrives
	fn_sqlagent_job_history
	fn_sqlagent_jobs
	fn_sqlagent_jobsteps
	fn_sqlagent_jobsteps_logs
	fn_sqlagent_subsystems
	fn_sqlvarbasetostr
	fn_stmt_sql_handle_from_sql_stmt
	fn_trace_geteventinfo
	fn_trace_getfilterinfo
	fn_trace_getinfo
	fn_trace_gettable
	fn_translate_permissions
	fn_validate_plan_guide
	fn_varbintohexstr
	fn_varbintohexsubstring
	fn_virtualfilestats
	fn_virtualservernodes
	fn_xe_file_target_read_file
	fn_yukonsecuritymodelrequired

支持	不支持
	dm_cryptographic_provider_algorithms
	dm_cryptographic_provider_keys
	dm_cryptographic_provider_sessions
	dm_db_database_page_allocations
	dm_db_incremental_stats_properties
	dm_db_index_operational_stats
	dm_db_index_physical_stats
	dm_db_log_info
	dm_db_log_stats
	dm_db_missing_index_columns
	dm_db_objects_disabled_on_compatibility_level_change
	dm_db_page_info
	dm_db_stats_histogram
	dm_db_stats_properties- dm_db_stats_properties_internal
	dm_enumerate_blobdirectory
	dm_exec_cached_plan_dependent_objects
	dm_exec_cursors
	dm_exec_describe_first_result_set
	dm_exec_describe_first_result_set_for_object
	dm_exec_input_buffer
	dm_exec_plan_attributes
	dm_exec_query_plan
	dm_exec_query_plan_stats
	dm_exec_query_statistics_xml
	dm_exec_sql_text
	dm_exec_text_query_plan
	dm_exec_xml_handles

支持	不支持
	dm_fts_index_keywords
	dm_fts_index_keywords_by_document
	dm_fts_index_keywords_by_property
	dm_fts_index_keywords_position_by_document
	dm_fts_parser
	dm_io_virtual_file_stats
	dm_logconsumer_cachebufferrefs
	dm_logconsumer_privatecachebuffers
	dm_logpool_consumers
	dm_logpool_sharedcachebuffers
	dm_logpoolmgr_freepools
	dm_logpoolmgr_respoolsize
	dm_logpoolmgr_stats
	dm_os_enumerate_filesystem
	dm_os_file_exists
	dm_os_volume_stats
	dm_sql_referenced_entities
	dm_sql_referencing_entities
	geographycollectionaggregate
	geographyconvexhullaggregate
	geographyenvelopeaggregate
	geographyunionaggregate
	geometrycollectionaggregate
	geometryconvexhullaggregate
	geometryenvelopeaggregate
	geometryunionaggregate
	ormask

支持	不支持
	fn_syspolicy_is_automation_enabled

4 排序规则

在 T-SQL 语句中，可以包括 to 指定列或表应使用数据库的默认排序规则。在 UDBTX-TDS 中，您应该删除 COLLATE DATABASE_DEFAULT 子句，或指定归类名称。

DROP TABLE IF EXISTS t1

-- This CREATE TABLE statement works in T-SQL but not in UDBTX-TDS.

CREATE TABLE t1(col1 NVARCHAR(24) COLLATE DATABASE_DEFAULT NOT NULL)

DROP TABLE IF EXISTS t1

-- This CREATE TABLE statement works the same in T-SQL and UDBTX-TDS.

CREATE TABLE t1(col1 NVARCHAR(24) NOT NULL)

通常认为删除如上所示的 COLLATE DATABASE_DEFAULT 子句更可取，因为您不必在源代码中对任何排序规则名称进行硬编码。如果确实要指定确切的排序规则名称，则可以使用以下命令检索正在使用的数据库的排序规则：

SELECT CAST(DATABASEPROPERTYEX('my_database', 'Collation') AS varchar(64)) AS CollationId

CollationId

-————————————————————————

sql_latin1_general_cp1_ci_as

然后，指定数据库的排序规则，而不是在 UDBTX-TDS 中指定排序规则。

drop table if exists t1

CREATE TABLE t1(col1 NVARCHAR(24) COLLATE sql_latin1_general_cp1_ci_as NOT NULL)

5 动态游标

目前，UDBTX-TDS 无法识别在存储过程或函数中动态定义的游标。解决方法是使用临时表重写游标。

设置代码：

DROP TABLE IF EXISTS my_table

GO

```
CREATE TABLE my_table (c1 int, c2 varchar(64), c3 tinyint, c4 money, c5 datetime)

GO
```

在 SQL Server 中编译和工作的过程，但不是 UDBTX-TDS：

```
DROP PROC IF EXISTS sp_cur_test

GO

CREATE PROC sp_cur_test (@TableName varchar(100))

AS

DECLARE @exec_str VARCHAR(4000);

DECLARE @table_schema VARCHAR(100), @table_name VARCHAR(100), @column_name VARCHAR(100);

SELECT @exec_str = 'DECLARE fetch_cursor

CURSOR FOR SELECT table_schema, table_name, column_name

FROM INFORMATION_SCHEMA.COLUMNS

WHERE table_name = '' + @TableName + '';

EXEC(@exec_str);

OPEN fetch_cursor;

FETCH fetch_cursor INTO @table_schema, @table_name, @column_name;

WHILE @@FETCH_STATUS = 0

BEGIN

PRINT @table_schema;

PRINT @table_name;

PRINT @column_name;

FETCH fetch_cursor INTO @table_schema, @table_name, @column_name;

END;

CLOSE fetch_cursor;

DEALLOCATE fetch_cursor;

GO
```

```
execute sp_cur_test 'MY_TABLE';

GO
```

在 SQL Server 或 UDBTX-TDS 中使用临时表重写游标后的过程：

```
DROP PROC IF EXISTS sp_cur_test;

GO

CREATE PROC sp_cur_test (@TableName varchar(100))

AS

DECLARE @exec_str VARCHAR(4000);

DECLARE @table_schema VARCHAR(100), @table_name VARCHAR(100), @column_name VARCHAR(100);

CREATE TABLE #temp_cur (table_schema VARCHAR(100), table_name VARCHAR(100), column_name

VARCHAR(100));

SELECT @exec_str = 'INSERT INTO #temp_cur

SELECT table_schema, table_name, column_name

FROM INFORMATION_SCHEMA.COLUMNS

WHERE lower(table_name) = LOWER('' + @TableName + '')';

EXEC(@exec_str);

DECLARE fetch_cursor CURSOR FOR SELECT * FROM #temp_cur;

OPEN fetch_cursor;

FETCH fetch_cursor INTO @table_schema, @table_name, @column_name;

WHILE @@FETCH_STATUS = 0

BEGIN

PRINT @table_schema;

PRINT @table_name;

PRINT @column_name;

FETCH fetch_cursor INTO @table_schema, @table_name, @column_name;

END;
```

```
CLOSE fetch_cursor;

DEALLOCATE fetch_cursor;

GO

execute sp_cur_test 'MY_TABLE';

GO
```

6 分区表

6.1 单数据库模式

在 UDBTX-TDS TDS 端口上创建的分区表和在 UDBTX 端口上创建的分区表之间的主要区别在于表所有者。以下示例演示了如何在 UDBTX 端口上创建的表的所有者更改为 dbo，以便您可以访问 UDBTX-TDS 端口和 UDBTX 端口上的表。

范围分区示例

下面的示例创建并测试具有两个分区的分区表。首先，使用 pgAdmin（在 UDBTX 端口上）创建表：

```
DROP TABLE IF EXISTS dbo.PartitionTest;

DROP TABLE IF EXISTS dbo.PartitionTest_y2022m01;

DROP TABLE IF EXISTS dbo.PartitionTest_y2022m02;

DROP INDEX dbo.partitiontest_logdate_idx;`

CREATE TABLE IF NOT EXISTS dbo.PartitionTest

(

city_id integer NOT NULL,

logdate date NOT NULL,

peaktemp integer,

unitsales integer

) PARTITION BY RANGE (logdate);

CREATE INDEX PartitionTest_logdate_idx

ON dbo.PartitionTest(logdate ASC NULLS LAST);
```

Then, create the partitions:

```
CREATE TABLE IF NOT EXISTS dbo.PartitionTest_y2022m01 PARTITION OF dbo.PartitionTest
```

FOR VALUES FROM (‘2022-01-01’) TO (‘2022-02-01’);

CREATE TABLE IF NOT EXISTS dbo.PartitionTest_y2022m02 PARTITION OF dbo.PartitionTest

FOR VALUES FROM (‘2022-02-01’) TO (‘2022-03-01’);

然后，使用 pgAdmin 添加数据：

INSERT INTO dbo.partitiontest VALUES (1,’2022-01-01’,1,1);

INSERT INTO dbo.partitiontest VALUES (2,’2022-01-10’,1,2);

INSERT INTO dbo.partitiontest VALUES (3,’2022-01-15’,1,3);

INSERT INTO dbo.partitiontest VALUES (4,’2022-02-01’,2,1);

INSERT INTO dbo.partitiontest VALUES (5,’2022-02-03’,2,2);

INSERT INTO dbo.partitiontest VALUES (6,’2022-02-11’,2,3);

INSERT INTO dbo.partitiontest VALUES (7,’2022-02-15’,2,4);

INSERT INTO dbo.partitiontest VALUES (8,’2022-02-16’,2,5);

INSERT INTO dbo.partitiontest VALUES (8,’2022-02-17’,2,6);

INSERT INTO dbo.partitiontest VALUES (8,’2022-02-20’,2,7);

INSERT INTO dbo.partitiontest VALUES (8,’2022-02-21’,2,8);

当您从 pgAdmin 和 UDBTX-TDS 查询数据时：

SELECT * FROM dbo.partitiontest

• pgAdmin（在 UDBTX 端口上）将显示所有数据。• SSMS（在 UDBTX-TDS 端口上）将显示以下错误消息，并且对象浏览器不会显示表名称：

Msg 33557097, Level 16, State 1, Line 3

relation “master_dbo.partitiontest” does not exist

使用 pgAdmin（在 UDBTX 端口上）将表所有者更改为 dbo：

ALTER TABLE dbo.partitiontest OWNER to dbo;

ALTER TABLE dbo.partitiontest_y2022m01 OWNER to dbo;

ALTER TABLE dbo.partitiontest_y2022m02 OWNER to dbo;

Then, when you query the data from pgAdmin and SSMS:

SELECT * FROM dbo.partitiontest

pgAdmin (on the UDBTX port) will display all the data.

SSMS (on the UDBTX-TDS port) will display the data and you should see the tables in the object browser.

Analyzing the result set

First, query the database using SSMS on the TDS port:

SET UDBTX-TDS_SHOWPLAN_ALL ON

SELECT * FROM dbo.partitiontest WHERE logdate = '2022-02-21'

Query Text: Select * from dbo.partitiontest where logdate = '2022-02-21'

Bitmap Heap Scan on partitiontest_y2022m02 partitiontest (cost=4.22..14.76 rows=9 width=16)

Recheck Cond: (logdate = '2022-02-21'::date)

-> Bitmap Index Scan on partitiontest_y2022m02_logdate_idx (cost=0.00..4.22 rows=9 width=0)

Index Cond: (logdate = '2022-02-21'::date)

Then, using pgAdmin on the UDBTX port:

EXPLAIN ANALYZE

SELECT * FROM dbo.partitiontest WHERE logdate = '2022-02-21'

"Bitmap Heap Scan on partitiontest_y2022m02 partitiontest (cost=4.22..14.76 rows=9 width=16) (actual time=0.015..0.016 rows=1 loops=1)"

" Recheck Cond: (logdate = '2022-02-21'::date)"

" Heap Blocks: exact=1"

" -> Bitmap Index Scan on partitiontest_y2022m02_logdate_idx (cost=0.00..4.22 rows=9 width=0) (actual time=0.011..0.011 rows=1 loops=1)"

" Index Cond: (logdate = '2022-02-21'::date)"

"Planning Time: 0.124 ms"

"Execution Time: 0.043 ms"

Inheritance Partitioning Example

First, create the database objects that will be used in our example:

DROP TRIGGER IF EXISTS insert_measurement_trigger ON dbo.measurement_inheritance;

```
DROP FUNCTION IF EXISTS dbo.measurement_inheritance_insert_trigger();

DROP TABLE IF EXISTS dbo.measurement_inheritance_y2006m02;

DROP TABLE IF EXISTS dbo.measurement_inheritance_y2006m03;

DROP TABLE IF EXISTS dbo.measurement_inheritance;

CREATE TABLE dbo.measurement_inheritance (

city_id int not null,

logdate date not null,

peaktemp int,

unitsales int

);

CREATE TABLE dbo.measurement_inheritance_y2006m02 (

CHECK ( logdate >= DATE '2006-02-01' AND logdate < DATE '2006-03-01' )

) INHERITS (dbo.measurement_inheritance);

CREATE TABLE dbo.measurement_inheritance_y2006m03 (

CHECK ( logdate >= DATE '2006-03-01' AND logdate < DATE '2006-04-01' )

) INHERITS (dbo.measurement_inheritance);

DROP INDEX IF EXISTS dbo_log_measurement_inheritance_y2006m02;

DROP INDEX IF EXISTS dbo_log_measurement_inheritance_y2006m03;

CREATE INDEX dbo_log_measurement_inheritance_y2006m02 ON dbo.measurement_inheritance_y2006m02 (logdate);

CREATE INDEX dbo_log_measurement_inheritance_y2006m03 ON dbo.measurement_inheritance_y2006m03 (logdate);

CREATE OR REPLACE FUNCTION dbo.measurement_insert_trigger()

RETURNS TRIGGER AS $$

DECLARE

year1 int;

month1 smallint;

BEGIN
```

```
year1 = EXTRACT(YEAR FROM new.logdate );

month1 = EXTRACT(month FROM new.logdate );

raise info 'year1 : % ',year1;

if month1<10 then

EXECUTE 'INSERT INTO dbo.measurement_inheritance_y'|| year1::varchar(4)||'m0'|| month1::varchar||' VALUES ('||

NEW.city_id::varchar||','|| '''' ||NEW.logdate::date||''''

' ,'|| NEW.peaktemp::varchar||','||NEW.unitsales::varchar

|| ')';

ELSE

EXECUTE 'INSERT INTO dbo.measurement_inheritance_y'|| year1::varchar(4)||'m'|| month1::varchar||' VALUES ('||

NEW.city_id::varchar||','|| '''' ||NEW.logdate::date||''''

' ,'|| NEW.peaktemp::varchar||','||NEW.unitsales::varchar

|| ')';

END IF;

-- 2008m01 VALUES (NEW.*);

RETURN NULL;

END;

$$

LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS insert_measurement_trigger ON dbo.measurement_inheritance;

CREATE TRIGGER insert_measurement_trigger

BEFORE INSERT ON dbo. measurement_inheritance

FOR EACH ROW EXECUTE FUNCTION dbo.measurement_insert_trigger();
```

Then, add half of the test data to the table using pgAdmin (on the UDBTX port):

```
INSERT INTO dbo.measurement_inheritance VALUES (1,'2006-02-01',1,1);

INSERT INTO dbo.measurement_inheritance VALUES (2,'2006-02-10',1,2);
```

INSERT INTO dbo.measurement_inheritance VALUES (3,'2006-02-15',1,3);

INSERT INTO dbo.measurement_inheritance VALUES (4,'2006-03-01',2,1);

INSERT INTO dbo.measurement_inheritance VALUES (5,'2006-03-03',2,2);

INSERT INTO dbo.measurement_inheritance VALUES (6,'2006-03-11',2,3);

INSERT INTO dbo.measurement_inheritance VALUES (7,'2006-03-15',2,4);

INSERT INTO dbo.measurement_inheritance VALUES (8,'2006-03-16',2,5);

INSERT INTO dbo.measurement_inheritance VALUES (8,'2006-03-17',2,6);

INSERT INTO dbo.measurement_inheritance VALUES (8,'2006-03-20',2,7);

INSERT INTO dbo.measurement_inheritance VALUES (8,'2006-03-21',2,8);

Use pgAdmin to change the table owner to dbo:

ALTER FUNCTION dbo.measurement_insert_trigger() OWNER TO dbo;

ALTER TABLE dbo.measurement_inheritance OWNER to dbo;

ALTER TABLE dbo.measurement_inheritance_y2006m02 OWNER TO dbo;

ALTER TABLE dbo.measurement_inheritance_y2006m03 OWNER TO dbo;

Add Data from SSMS:

INSERT INTO dbo.measurement_inheritance values (1,'2006-02-05',1,4);

INSERT INTO dbo.measurement_inheritance values (2,'2006-02-15',1,5);

INSERT INTO dbo.measurement_inheritance values (3,'2006-02-20',1,6);

INSERT INTO dbo.measurement_inheritance values (4,'2006-03-02',2,9);

INSERT INTO dbo.measurement_inheritance values (5,'2006-03-05',2,10);

INSERT INTO dbo.measurement_inheritance values (6,'2006-03-12',2,11);

INSERT INTO dbo.measurement_inheritance values (7,'2006-03-16',2,12);

INSERT INTO dbo.measurement_inheritance values (8,'2006-03-18',2,13);

INSERT INTO dbo.measurement_inheritance values (8,'2006-03-19',2,14);

INSERT INTO dbo.measurement_inheritance values (8,'2006-03-23',2,15);

INSERT INTO dbo.measurement_inheritance values (8,'2006-03-25',2,16);

打开 EXECUTE ANALYZE 功能，并查询数据：

SET UDBTX-TDS_SHOWPLAN_ALL on

SELECT * FROM dbo.measurement_inheritance WHERE logdate =’2006-03-25’;

冲突分区示例

首先，创建将在示例中使用的数据库对象：

DROP TABLE IF EXISTS dbo.customers;

CREATE TABLE dbo.customers (

customer_id serial PRIMARY KEY,

name VARCHAR UNIQUE,

email VARCHAR NOT NULL,

active bool NOT NULL DEFAULT TRUE

);

Then, add data:

INSERT INTO

dbo.customers (name, email)

VALUES

(‘ABC’, ‘contact@abc.com’),

(‘MBA’, ‘contact@mba.com’),

(‘XYZ’, ‘contact@xyz.com’);

CREATE OR REPLACE FUNCTION dbo.onConflictTestFromSSM()

RETURNS void AS $$

DECLARE

year1 int;

month1 smallint;

BEGIN

INSERT INTO dbo.customers (NAME, email)

VALUES('MBA','hotline@mbacom')

ON CONFLICT (name)

DO

UPDATE SET email = EXCLUDED.email || ';' || customers.email;

RAISE INFO 'function executes';

END;

$$

LANGUAGE plpgsql;

ALTER FUNCTION dbo.onConflictTestFromSSM() OWNER TO dbo;

ALTER TABLE dbo.customers OWNER to dbo;

使用 SSMS 查询 dbo.onConflictTestFromSSM（）：

SELECT * FROM dbo.onConflictTestFromSSM()

EXEC dbo.onConflictTestFromSSM

SELECT * FROM dbo.customers

6.2 多数据库模式

在UDBTX 端用PG语法创建分区表：

```
DROP TABLE IF EXISTS nidc_dbo.PartitionTest;
DROP TABLE IF EXISTS nidc_dbo.PartitionTest_y2022m01;
DROP TABLE IF EXISTS nidc_dbo.PartitionTest_y2022m02;
DROP INDEX nidc_dbo.partitiontest_logdate_idx;

CREATE TABLE IF NOT EXISTS nidc_dbo.PartitionTest
(
    city_id integer NOT NULL,
    logdate date NOT NULL,
    peaktemp integer,
    unitsales integer
) PARTITION BY RANGE (logdate);

CREATE INDEX PartitionTest_logdate_idx
    ON nidc_dbo.PartitionTest(logdate ASC NULLS LAST);

```

```
CREATE TABLE IF NOT EXISTS nidc_dbo.PartitionTest_y2022m01 PARTITION OF nidc_dbo.PartitionTest
    FOR VALUES FROM ('2022-01-01') TO ('2022-02-01');

CREATE TABLE IF NOT EXISTS nidc_dbo.PartitionTest_y2022m02 PARTITION OF nidc_dbo.PartitionTest
    FOR VALUES FROM ('2022-02-01') TO ('2022-03-01');
```

授权给nidc_dbo， nidc是数据库名称。不授权，在SSMS端不可见。

```
ALTER TABLE nidc_dbo.partitiontest OWNER to  nidc_dbo;
ALTER TABLE nidc_dbo.partitiontest_y2022m01 OWNER to  nidc_dbo;
ALTER TABLE nidc_dbo.partitiontest_y2022m02 OWNER to  nidc_dbo;
```

在UDBTX端用下面的语句插入数据

```
INSERT INTO nidc_dbo.partitiontest VALUES (1,'2022-01-01',1,1);
INSERT INTO nidc_dbo.partitiontest VALUES (2,'2022-01-10',1,2);
INSERT INTO nidc_dbo.partitiontest VALUES (3,'2022-01-15',1,3);
INSERT INTO nidc_dbo.partitiontest VALUES (4,'2022-02-01',2,1);
INSERT INTO nidc_dbo.partitiontest VALUES (5,'2022-02-03',2,2);
INSERT INTO nidc_dbo.partitiontest VALUES (6,'2022-02-11',2,3);
INSERT INTO nidc_dbo.partitiontest VALUES (7,'2022-02-15',2,4);
INSERT INTO nidc_dbo.partitiontest VALUES (8,'2022-02-16',2,5);
INSERT INTO nidc_dbo.partitiontest VALUES (8,'2022-02-17',2,6);
INSERT INTO nidc_dbo.partitiontest VALUES (8,'2022-02-20',2,7);
INSERT INTO nidc_dbo.partitiontest VALUES (8,'2022-02-21',2,8);
```

在SSMS端用如下语句插入数据

```
INSERT INTO partitiontest VALUES (1,'2022-01-01',1,1);
INSERT INTO partitiontest VALUES (2,'2022-01-10',1,2);
INSERT INTO partitiontest VALUES (3,'2022-01-15',1,3);
INSERT INTO partitiontest VALUES (4,'2022-02-01',2,1);
INSERT INTO partitiontest VALUES (5,'2022-02-03',2,2);
INSERT INTO partitiontest VALUES (6,'2022-02-11',2,3);
INSERT INTO partitiontest VALUES (7,'2022-02-15',2,4);
INSERT INTO partitiontest VALUES (8,'2022-02-16',2,5);
INSERT INTO partitiontest VALUES (8,'2022-02-17',2,6);
INSERT INTO partitiontest VALUES (8,'2022-02-20',2,7);
INSERT INTO partitiontest VALUES (8,'2022-02-21',2,8);
```

7 重建表索引

在 T-SQL 中，使用 DBCC DBREINDEX 语句或 ALTER INDEX 语句重新索引表中的所有索引。目前，UDBTX-TDS 不支持这些说法。相反，您可以从 Postgres 连接运行 UDBTX 语句。有关 UDBTX 语句的信息，请参阅 UDBTX 文档中的 REINDEX。REINDEX TABLE

如果 T-SQL 维护脚本具有如下语句之一：

DBCC DBREINDEX (‘my_database.dbo.my_table’);

或

ALTER INDEX ALL ON my_database.dbo.my_table REBUILD;

然后运行以下等效语句，同时连接到启用了 UDBTX-TDS 的服务器的 UDBTX 端口：

REINDEX TABLE my_database_dbo.my_table; – For servers using multi-database mode

或

REINDEX TABLE dbo.my_table; – For servers using single-database mode

为方便起见，您可以定义一个 UDBTX 存储过程，该存储过程运行指定语句的语句，并使用 T-SQL 语句调用该存储过程。如果这样做，请确保调用过程的权限与对表执行 DDL 操作的权限一致，并且任何动态创建的语句都引用所有架构和表名称以避免格式错误的语句。

8 创建用户

8.1 创建UDBTX-TDS管理账号

依次执行下面的命令，创建sc账号，创建成功后即可通过TDS端口登录。

ud_sql -p 5678 -U unvdb -d unvdb -c “CREATE USER sc WITH SUPERUSER CREATEDB CREATEROLE PASSWORD ‘12345678’ INHERIT;”

ud_sql -p 5678 -U unvdb -d unvdbtds_db -c “GRANT ALL ON SCHEMA sys to sc;”

ud_sql -p 5678 -U unvdb -d unvdbtds_db -c “ALTER USER sc CREATEDB;”

ud_sql -p 5678 -U unvdb -d unvdbtds_db -c “call sys.babel_initialize_logins(‘sc’);”

执行结果如下：

[root@localhost unvdb-data]# ud_sql -p 5678 -U unvdb -d unvdb -c “CREATE USER sc WITH SUPERUSER CREATEDB CREATEROLE PASSWORD ‘12345678’ INHERIT;”

CREATE ROLE

[root@localhost unvdb-data]# ud_sql -p 5678 -U unvdb -d unvdbtds_db -c “GRANT ALL ON SCHEMA sys to sc;”

GRANT

[root@localhost unvdb-data]# ud_sql -p 5678 -U unvdb -d unvdbtds_db -c “ALTER USER sc CREATEDB;”

ALTER ROLE

[root@localhost unvdb-data]# ud_sql -p 5678 -U unvdb -d unvdbtds_db -c “call sys.babel_initialize_logins(‘sc’);”

CALL

[root@localhost unvdb-data]#

8.2 创建UDBTX-TDS普通账号

用UDBTX-TDS管理员账户通过TDS端口登录，然后执行如下命令：

-- Creates the login test_user with password ‘test123456’.

CREATE LOGIN test_user

WITH PASSWORD = ‘Test123456’;

GO

-- Creates a database user for the login created above.

CREATE USER test_user FOR LOGIN test_user;

GO

执行完成后test_user就可以通过TDS端口登录。

8.3 查询系统中的TDS账号

执行如下命令查询：

SELECT name

FROM sys.server_principals;

GO

9 常见操作SQL语句及兼容性

9.1 通过TDS端口的常见操作

操作类别	SQL示例	
系统查询	查询版本	SELECT @@version*;*
	查询数据库信息	SELECT * FROM sys.databases;

操作类别	SQL示例	
数据库操作	创建数据库	CREATE DATABASE testdb; 说明 迁移模式为single-db时， 只支持创建一个数据库，如果您已创建了一个数据库，则无法再次创建。
	查询数据库	SELECT * FROM sys.databases WHERE name = ‘testdb’;
	切换数据库	USE testdb GO SELECT db_name();
	删除数据库	DROP DATABASE testdb;
Schema操作	创建Schema	CREATE SCHEMA sch_demo;
	查看Schema	SELECT * FROM sys.schemas AS sch WHERE sch.name = ‘sch_demo’;
	创建Schema下表	CREATE TABLE sch_demo.tb_demo(id int); SELECT sch.name AS schema_name, tb.name AS table_name FROM sys.tables AS tb INNER JOIN sys.schemas AS sch ON tb.schema_id = sch.schema_id WHERE tb.name = ‘tb_demo’;
	删除Schema	说明 如果Schema下存在表，需要先删除表后，再删除Schema。 DROP TABLE sch_demo.tb_demo; GO DROP SCHEMA sch_demo; GO
表操作	新建表	USE testdb GO CREATE TABLE dbo.tb_test(id int not null IDENTITY(1,1) PRIMARY KEY, name varchar(50)) GO
	查询表	SELECT sche.name AS schema_name, tb.name AS table_name FROM sys.tables AS tb INNER JOIN sys.schemas AS sche ON tb.schema_id = sche.schema_id WHERE tb.name = ‘tb_test’; GO
	新增字段	ALTER TABLE dbo.tb_test ADD col_added bigint null; GO
	修改表字段	ALTER TABLE dbo.tb_test ALTER column col_added varchar(50); GO
	删除表字段	ALTER TABLE dbo.tb_test DROP column col_added; GO
	创建索引	CREATE INDEX ix_tb_test_name ON tb_test(name); GO
	删除索引	DROP INDEX ix_tb_test_name ON tb_test; GO
数据库操作	INSERT	INSERT INTO dbo.tb_test SELECT ‘A’ UNION ALL SELECT ‘B’; GO
	SELECT	SELECT * FROM dbo.tb_test;
	UPDATE	UPDATE TOP(1) dbo.tb_test SET name = ‘A_updated’; GO
	DELETE	DELETE TOP(1) FROM dbo.tb_test; GO SELECT * FROM dbo.tb_test;
存储过程	创建存储过程	USE testdb GO CREATE PROC dbo.UP_getDemoData(@id int) AS BEGIN SET NOCOUNT ON SELECT * FROM dbo.tb_test WHERE id = @id END; GO
	查看存储过程	SELECT * FROM sys.procedures WHERE name = ‘up_getdemodata’;

操作类别	SQL示例	
	执行存储过程	EXEC dbo.UP_getDemoData @id = 7; GO
	删除存储过程	USE testdb GO DROP PROC dbo.UP_getDemoData GO

9.2 兼容性说明

说明

不支持的SQL操作如下：

查看表结构。

EXEC sp_help ‘dbo.tb_test’

不支持在修改表字段时设置默认值NULL。

ALTER TABLE dbo.tb_test ALTER column col_added varchar(50) null;

GO

不支持重建索引，建议删除后，重新创建。

ALTER INDEX ix_tb_test_name ON tb_test REBUILD;

GO

不支持修改存储过程，建议删除后，重新创建。

USE testdb

GO

ALTER PROC dbo.UP_getDemoData(

@id int

)

AS

BEGIN

SET NOCOUNT ON

SELECT *

FROM dbo.tb_test

WHERE id >= @id

END;

GO

不支持执行计划（showplan_xml）。

SET showplan_xml ON

SELECT * from tb_test;

10 客户端编程

10.1 C语言

在本节中，我们将演示如何编写一个简单的 C/C++ 程序来访问 UDBTX-TDS。该示例不会创建完整且安全的程序，但突出显示了您的程序与 UDBTX-TDS 一起使用所需的一些功能。

本节中的示例是使用 FreeTDS 库为 Linux 编写的，该库适用于大多数 Linux 发行版。

要开始使用，请安装并配置 FreeTDS 驱动程序：

[root@fedora ~]# dnf install freetds*

Last metadata expiration check: 0:02:52 ago on Fr 24 Sep 2021 11:11:43 CEST.

Dependencies resolved.

==========

Package Architecture Version Repository Size

==========

Installing:

freetds x86_64 1.1.20-4.fc34 fedora 373 k

freetds-devel x86_64 1.1.20-4.fc34 fedora 39 k

freetds-doc noarch 1.1.20-4.fc34 fedora 1.0 M

Installing dependencies:

freetds-libs x86_64 1.1.20-4.fc34 fedora 423 k

如果您使用 C 语言编码，请确保您还安装了包（头文件）。在系统上编译代码时会使用头文件。

连接到数据库

示例的第一部分管理头文件、变量定义和错误处理。该示例使用标准 C 库（stdio.h、stdlib.h、unistd.h、sys/param.h）和 FreeTDS 库（sybfront.h、sybdb.h 和 syberror.h）中的头文件：

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/param.h>

#include <sybfront.h>

#include <sybdb.h>

#include <syberror.h>

#define UID “postgres”

#define PWD “verysecret!”

#define PROGNAME “demo_prog”

#define DBSERVER “sample-host.example.com”

#define DBNAME “postgres”

/* handler for messages from the server */

static int

msg_handler(DBPROCESS* dbproc, DBINT msgno, int msgstate, int severity,

char *msgtext, char *srvname, char *procname, int line)

{

/* regular errors are handled by the error handler */

if (severity < 11)

fprintf(stderr, “Server message (severity %d): %s\n”, severity, msgtext);

return 0;

}

/* error handler */

static int err_handler(DBPROCESS* dbproc, int severity, int dberr, int oserr, char *dberrstr, char *oserrstr)
```

```
{
fprintf(stderr, “Server error %d: %s\n”, dberr, dberrstr);
if (oserr != 0)
fprintf(stderr, “Caused by system error %d: %s\n”, oserr, oserrstr);
return INT_CANCEL;
}
```

该示例还在初始化 FreeTDS 库之前声明局部变量：

```
int main(void)
{
LOGINREC *login;
DBPROCESS *dbconn;
char hostname[MAXHOSTNAMELEN];
int max_len = MAXHOSTNAMELEN;
DBCHAR accession[10];
DBCHAR examdesc[10];
DBCHAR examcode[255];
if (dbinit() == FAIL)
{
fprintf(stderr, “Could not init db.\n”);
return 1;
}
```

接下来，分配一个登录结构，并将登录 ID、密码和主机名设置到登录句柄中：

```
/* Allocate a login params structure */
if ((login = dblogin()) == FAIL)
{
fprintf(stderr, “Could not initialize dblogin() structure.\n”);
```

return 2;

}

/* Initialize the login params in the structure */

DBSETLUSER(login, UID);

DBSETLPWD(login, PWD);

if (gethostname(hostname, max_len) == 0)

{

DBSETLHOST(login, hostname);

fprintf(stderr, “setting login hostname: %s\n”, hostname);

}

设置 TDS 端口环境变量（如果在默认端口 1433 上进行连接，则不需要）：

/* the port can only be set via environment variable */

if (putenv(“TDSPORT=1433”) != 0)

{

fprintf(stderr, “error setting TDSPORT environment variable\n”);

return 0;

}

安装错误处理程序和消息处理程序 - 如果发生错误/消息，TDS 库将调用给定的函数。我们的示例将消息发送到：stderr

/* install error handler */

dberrhandle(err_handler);

dbmsghandle(msg_handler);

调用以连接到服务器：dbopen

/* Now connect to the DB Server */

if ((dbconn = dbopen(login, DBSERVER)) == NULL)

{

fprintf(stderr, “Could not connect to DB Server: %s\n”, DBSERVER);

return 3;

}

printf(“success\n”);

return 0;

}

使用 GCC 编译代码：

gcc main_01.c -lsybdb -I/usr/include/ -o main_01

编译代码时，有两件事至关重要：

您应该包括 -I 选项，以告诉编译器在哪里查找头文件。

包括 -lsydbd 选项以确保 FreeTDS 库已正确链接。

此示例创建名为 的二进制文件。main_01

这是一个简单的例子，但它突出显示了在开始开发与 C 语言 UDBTX-TDS 服务器联系的程序时需要捕获的行为。

FreeTDS 配置问题

以下消息是由 FreeTDS 配置错误引起的：

[hs@fedora tds_test]$ ./main_01

setting login hostname: fedora

Server error 20018: General SQL Server error: Check messages from the SQL Server

Caused by system error -1: (null)

Server error 20002: Adaptive Server connection failed (sample-host.example.com)

Server error 20002: Adaptive Server connection failed (sample-host.example.com)

Could not connect to DB Server: sample-host.example.com)

如果您收到此错误消息，则 FreeTDS 配置文件可能包含错误或不存在。

10.2 C#语言

Microsoft SQL Server是Microsoft Windows平台上的主要数据库系统之一，C#是该生态系统中使用的较流行的语言之一。因此，了解如何建立 C# 连接以及如何获取数据非常重要。

C#语言示例代码

让我们看一下示例代码：

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

namespace sample

{

class Program

{

static void Main(string[] args)

{

// Setting up MSSQL Credentials

SqlConnection con;

string conString = “Server=” + @”PUT_HOSTNAME_HERE” + “;” +

“User id=” + “PUT_USERNAME_HERE” + “;” +

“Password=” + “PUT_PASSWORD_HERE” + “;” +

“Database=” + “PUT_DATABASE_HERE” + “;” +

“MultipleActiveResultSets=true;”;

con = new SqlConnection(conString);

SqlCommand cmd = new SqlCommand();

// Creating MSSQL Connection

try

{

con.Open();
```

```
Console.WriteLine(“Connection established\n”) ;
}
catch
{
Console.WriteLine(“Can not connect to database!\nPlease check credentials!”);
Environment.Exit(1);
}
string sqlQuery = “”;
// Select values example
select_all(con);
// Transaction example
// Insert values into sample table
cmd = con.CreateCommand();
SqlTransaction transaction = con.BeginTransaction(“SampleTransaction”);
try
{
sqlQuery = “INSERT INTO sample VALUES(@vorname, @nachname, @persoid)”;
cmd.Parameters.AddWithValue(“@vorname”, “Max”);
cmd.Parameters.AddWithValue(“@nachname”, “Mustermann”);
cmd.Parameters.AddWithValue(“@persoid”, “1020”);
cmd.CommandType = System.Data.CommandType.Text;
cmd.CommandText = sqlQuery;
cmd.Transaction = transaction;
cmd.ExecuteNonQuery();
cmd.Parameters.Clear();
cmd.Parameters.AddWithValue(“@vorname”, “Erika”);
```

```
cmd.Parameters.AddWithValue(“@nachname”, “Musterfrau”);

cmd.Parameters.AddWithValue(“@persoid”, “1021”);

cmd.ExecuteNonQuery();

transaction.Commit();

Console.WriteLine(“\nInsert successful!\n”);

}

catch

{

transaction.Rollback();

Console.WriteLine(“\nInsert failed!\n”);

}

select_all(con);

// Removing inserted values

sqlQuery = “DELETE FROM sample WHERE vorname = ‘Max’ or vorname = ‘Erika’”;

cmd = con.CreateCommand();

cmd.CommandText = sqlQuery;

int row_count = cmd.ExecuteNonQuery();

// Select metadata

// Select row count from delete

Console.WriteLine(“\nDeleted rows: “ + row_count + “\n”);

// Select column names from table

sqlQuery = “SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = ‘sample’”;

cmd = con.CreateCommand();

cmd.CommandText = sqlQuery;

SqlDataReader reader = cmd.ExecuteReader();

string value = “”;
```

```
while (reader.Read())
{
value += reader.GetValue(0) + " ";
}
Console.WriteLine(value);
reader.Close();
// Closing connection
con.Close();
Console.WriteLine("\nConnection closed!");
}
private static void select_all(SqlConnection con)
{
string sqlQuery = "SELECT * FROM sample";
SqlCommand cmd = con.CreateCommand();
cmd.CommandText = sqlQuery;
SqlDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
string value = "";
for (int i = 0; i != reader.FieldCount; i++)
{
value += reader.GetValue(i) + " ";
}
Console.WriteLine(value);
}
reader.Close();
```

}

}

}

代码相对简单。首先，分配并打开连接。在 C# 中，可以使用 try / catch 轻松完成错误处理。在我们的例子中，如果无法打开连接，程序就会终止。

然后我们调用 select_all 函数。它的作用是创建一个 SQL 命令，该命令只需读取“示例”表中的所有数据，并循环遍历所有行和所有列以在屏幕上显示它们。最后，SqlDataReader 关闭。这是一个简单的例子，展示了如何轻松地从UDBTX-TDS 中提取数据。

接下来是一个示例，演示如何运行事务。执行此操作的方法是运行 BeginTransaction 方法。运行 SQL 后，我们可以提交事务。错误处理由 try / catch 块完成。

可以使用 Microsoft Studio 或您选择的任何其他工具编译代码。需要注意的重要一点是，您可以像通常实现代码一样实现 UDBTX-TDS 的代码，直接与 Microsoft SQL Server 通信。

10.3 Python语言

将 Python 与 UDBTX-TDS 结合使用

在本节中，您将学习如何使用 Python 连接到 UDBTX-TDS 并从服务器中提取数据。将显示三种变体：

使用 Python 和 ODBC

使用 Python 和本机 TDS

将 Python 与 FreeTDS 配合使用（在 arm64 体系结构上不存在）msobdcsql

所有这三个变体都包含一些示例，这些示例将帮助您在 UDBTX-TDS 中处理事务操作。

使用 Python 和 ODBC

在运行 Python 代码之前，您需要确保所有库都存在。如果您使用的是 Ubuntu 20.04，则可以使用以下命令来更新任何丢失的包：

#!/bin/bash

sudo su

curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -

curl https://packages.microsoft.com/config/ubuntu/20.04/prod.list > /etc/apt/sources.list.d/mssql-release.list

exit

sudo apt-get update

sudo ACCEPT_EULA=Y apt-get install -y msodbcsql17

根据您的 Linux 发行版，安装过程会略有不同，但基本要求是相同的 - 确保已安装库。

在调用示例代码之前，还需要创建一个名为 的 UDBTX-TDS 表。您可以在 UDBTX-TDS 数据库的 TDS 端口上使用您选择的客户端进行连接，并使用以下命令：contacts

CREATE TABLE contacts (

contact_id INT PRIMARY KEY,

first_name VARCHAR (10) NOT NULL,

last_name VARCHAR (20) NOT NULL,

visited DATE

);

INSERT INTO contacts VALUES (‘1’, ‘Alex’, ‘Arthur’, ‘20210831’);

INSERT INTO contacts VALUES (‘2’, ‘Bonnie’, ‘Bret’, ‘20210831’);

INSERT INTO contacts VALUES (‘3’, ‘Colin’, ‘Cristobal’, ‘20210901’);

INSERT INTO contacts VALUES (‘4’, ‘Danielle’, ‘Dexter’, ‘20210907’);

INSERT INTO contacts VALUES (‘5’, ‘Earl’, ‘Edwards’, ‘20210908’);

INSERT INTO contacts VALUES (‘6’, ‘Fiona’, ‘Ferdinand’, ‘20210917’);

我们的示例侧重于使用 Python、ODBC 和 UDBTX-TDS 创建连接、读取数据并将其显示在屏幕上。首先，该示例包括它将使用的库：

import sys

import os

import pyodbc

然后，该示例创建与 SQL Server 主机的连接。连接字符串的语法必须包含正确格式的所有相关信息：

Provide your SQL Server credentials

server = ‘host.example.com’

database = ‘postgres’

username = ‘postgres’

password = ‘1safepassword’

```
# Establish a connection

try:

connection = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL
Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)

cursor = connection.cursor()

print("Connection established for select examples\n")

except pyodbc.ProgrammingError:

print("Cannot connect to the database\nPlease check credentials")

exit(1)
```

Then, the example creates a cursor that iterates through the rows of your contacts table and displays them:

```
# Select values

cursor.execute("SELECT * FROM contacts")

for row in cursor.fetchall():

print(row)
```

The example will display:

```
Connection established for select examples

(1, 'Alex', 'Arthur', 20210831)

(2, 'Bonnie', 'Bret', 20210831)

(3, 'Colin', 'Cristobal', 20210901)

(4, 'Danielle', 'Dexter', 20210907)

(5, 'Earl', 'Edwards', 20210908)

(6, 'Fiona', 'Ferdinand', 20210917)
```

Transaction handling

Our next example demonstrates using cursors and native TDS to perform some basic transaction handling. First, the example establishes a connection with the server:

```
# Transaction example

# Establish a UDBTX-TDS connection
```

```
try:

transact_connection = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL
Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password, autocommit=False)

t_cursor = transact_connection.cursor()

print("\nConnection established\n")

except pyodbc.ProgrammingError:

print("\nConnection attempt failed\nPlease check credentials")

exit(1)
```

Then, the example uses a cursor to add new entries to the database:

```
# Insert values into your contacts table

try:

t_cursor.execute("INSERT INTO contacts VALUES('Gaston', 'Gordon', '20210918')")

t_cursor.execute("INSERT INTO contacts VALUES('Hermione', 'Henry', '20210925')")

except pyodbc.Error as e:

t_cursor.rollback()

print("Insert failed: {}".format(e))

else:

t_cursor.commit()

print("Insert successful\n")
```

Then, the example iterates through the table and display the contents:

```
t_cursor.execute("SELECT * FROM contacts")

for row in t_cursor.fetchall():

print(row)

t_cursor.close()

transact_connection.close()
```

The example then removes two rows from the table:

```
# Remove inserted values

cursor.execute("DELETE FROM contacts WHERE last_name = 'Bret' OR last_name = 'Edwards'")

cursor.commit()

# Select metadata

print(f"\nDeleted Rows: {cursor.rowcount}\n")
```

Then, the example closes the connection and alerts the user:

```
# Close connection

cursor.close()

connection.close()

print("Success")
```

The example should display the following messages:

```
Connection established

Insert successful

(1, 'Alex', 'Arthur', 20210831)

(3, 'Colin', 'Cristobal', 20210901)

(4, 'Danielle', 'Dexter', 20210907)

(6, 'Fiona', 'Ferdinand', 20210917)

(7, 'Gaston', 'Gordon', 20210918)

(8, 'Hermione', 'Henry', 20210925)

Deleted Rows: 2

Success
```

The table now reflects the changes made in the example.

Using Python with FreeTDS

If you are using an arm64 architecture, you can use the FreeTDS drivers to connect through the TDS protocol.

Before invoking the sample code, use the following commands to install the prerequisite software:

```
sudo apt install python3-pip python3-venv freetds-dev
```

```
python3 -m venv .venv

source .venv/bin/activate

pip install pymssql psycopg2
```

The example writes data to UDBTX-TDS with the pymssql API, and reads it from UDBTX-TDS with the psycopg2 library. The example includes the libraries it needs to use those libraries before establishing a connection with the UDBTX-TDS host:

```
import sys

import os

import pymssql

import psycopg2

server = 'host.example.com'

port = 1433

database = 'master'

username = 'UDBTX-TDS_user'

password = '1safepassword'

pg_database = "UDBTX-TDS_db"

pg_schema = "master_dbo"

def main():

with pymssql.connect(server, username, password, database) as conn:

with conn.cursor() as cursor:
```

Then, the example drops and recreates a simple table (named) and a procedure (named):contactsfind_contact

```
try:

cursor.execute("DROP TABLE contacts")

cursor.execute("DROP PROCEDURE find_contact")

conn.commit()

except Exception as e:

# For the sake of this example, we
```

```
# ignore exceptions if the objects do not exist.
pass
try:
print("– create table: ")
cursor.execute("""
CREATE TABLE contacts (
contact_id INT PRIMARY KEY,
first_name VARCHAR (10) NOT NULL,
last_name VARCHAR (20) NOT NULL,
visited DATE
)
""")
cursor.execute("""
CREATE PROCEDURE find_contact
@last_name VARCHAR(100)
AS BEGIN
SELECT * FROM contacts WHERE last_name like '%'+@last_name+'%'
END
""")
conn.commit()
except Exception as e:
print("Cannot create table: {}".format(e))
exit(1)
```

The example adds contacts to the table:

```
try:
print("– Insert")
```

```
cursor.executemany(
“INSERT INTO contacts VALUES (%d, %s, %s, %s)”,
[
(11, ‘Imelda’, ‘Imani’, ‘20211004’),
(12, ‘Julian’, ‘Joyce’, ‘20211015’),
(13, ‘Katia’, ‘Kirk’, ‘20211027’)
])
conn.commit()
except Exception as e:
print(“Transaction could not be committed: {} “.format(e))
exit(2) # Duplicate key error
```

然后，该示例首先使用库查询表，然后使用 API：psycopg2pymmssql

```
with psycopg2.connect(user=username, password=password, host=server,database=pg_database) as connpg:
with connpg.cursor() as cursorpg:
print(“– Output from UDBTX-TDS: “)
try:
cursorpg.execute(“SELECT * FROM {}.contacts WHERE last_name like ‘%{}%’”.format(pg_schema,”Joyce”))
for row in cursorpg:
print(“{}”.format(row))
except Exception as e:
print(“Read query couldn’t be executed: {}”.format(e))
pass
with pymssql.connect(server, username, password, database) as conn:
with conn.cursor() as cursor:
print(“– Executing PROCEDURE “)
cursor.callproc(‘find_contact’, (‘Joyce’,))
```

for row in cursor:

print(“{}”.format(row))

该子句允许示例在块完成后清理游标和连接，以便更干净地执行。with

最后，将 init 调用添加到函数中：main()

if *_name_* == “main“:

main()

10.4 JAVA语言

利用 JDBC

在关注了其他一些流行的编程语言（如 Python 和 C#）之后，我们还想坐下来弄清楚如何将 UDBTX-TDS 与 Java 和 JDBC 结合使用。同样，将展示一些基本操作来描述如何编写代码。

正如我们之前所看到的，为 Microsoft SQL Server 和 UDBTX-TDS 编写 JDBC 客户端代码之间没有真正的区别。不过，让我们深入研究并看一些例子：

使用 JDBC 的示例代码

要编译 Java 代码，您将需要 Microsoft SQL Server JDBC 驱动程序。为了这个例子，我们使用了9.4（mssql-jdbc-9.4.0.jre11.jar）版本。但是，由于该示例具有非常基本的代码，因此也应该可以使用其他版本的驱动程序运行内容。

代码如下：

```
package sample;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class sample {

public static void main(String[] args) {

Connection connection;

// Setting up MSSQL Credentials

String connectionUrl =
```

```
“jdbc:sqlserver://PUT_HOSTNAME_HERE;”

+ “database=PUT_NAME_OF_DATABASE_HERE;”

+ “user=PUT_USERNAME_HERE;”

+ “password=PUT_PASSWORD_HERE;”;

try {

// Trying to establish connection

connection = DriverManager.getConnection(connectionUrl);

System.out.println(“Connection established for select examples!\n”);

// Select values example

select_all(connection);

// Transaction example

connection.setAutoCommit(false);

// Insert values into sample table

try {

connection.setAutoCommit(false);

String insert = “INSERT INTO sample VALUES(‘Max’, ‘Mustermann’, ‘1020’)”;

Statement statement = connection.createStatement();

statement.execute(insert);

insert = “INSERT INTO sample VALUES(‘Erika’, ‘Musterfrau’, ‘1021’)”;

statement.execute(insert);

connection.commit();

System.out.println(“\nInsert successful!\n”);

}

catch (SQLException e){

connection.rollback();

System.out.println(“\nInsert failed!\n”);
```

```
}

select_all(connection);

connection.setAutoCommit(true);

// Removing inserted values and displaying rowcount

try {

String delete = “DELETE FROM sample WHERE vorname = ‘Max’ or vorname = ‘Erika’”;

Statement statement = connection.createStatement();

int rowcount = statement.executeUpdate(delete);

System.out.println(“\nDeleted Rows: “ + rowcount + “\n”);

// Select column names

String select = “SELECT * FROM sample”;

ResultSet result = statement.executeQuery(select);

String value = “”;

for(int i = 1; i <= result.getMetaData().getColumnCount(); i++) {

value += result.getMetaData().getColumnName(i) + “ “;

}

System.out.println(value);

}

catch (SQLException e) {

e.printStackTrace();

}

}

catch (SQLException e) {

e.printStackTrace();

}

}
```

```
private static void select_all(Connection connection) {

try {

Statement statement = connection.createStatement();

String select = “SELECT * FROM sample”;

ResultSet result = statement.executeQuery(select);

while(result.next()) {

String value = “”;

for(int i = 1; i <= result.getMetaData().getColumnCount(); i++) {

value += result.getString(i) + “ “;

}

System.out.println(value);

}

}

catch (SQLException e) {

e.printStackTrace();

}

}

}
```

首先，创建一个标准的JDBC连接。JDBC是一个抽象层，所以主要的魔力实际上在连接字符串中：“jdbc：sqlserver”将告诉我们的程序使用Microsoft SQL Server驱动程序连接到UDBTX-TDS。

否则，示例代码大多是微不足道的，几乎代表了一个标准的 JDBC 应用程序。

10.5 支持的连接驱动程序

支持以下接口：

OLEDB Provider/MSOLEDBSQL

OLEDB Driver/SQLOLEDB (deprecated by Microsoft)

.NET Data Provider for SQL Server

SQL Server Native Client 11.0 (deprecated by Microsoft)

Microsoft SqlClient Data Provider for SQL Server

Open Database Connectivity (ODBC)

Java Database Connectivity (JDBC)

将来可能会添加更多连接驱动程序。由于 UDBTX-TDS 支持 TDS 协议，因此大多数基于 TDS 的客户端应用程序都应与 UDBTX-TDS 一起使用。

11 OPENXML 移植

11.1 原有SQL 存储过程

```
/*----------------------------------
-- 用途：成批插入虚拟身份好友IP记录
-- 项目名称：中心6数据代理6.1.8
-- 说明：
-- 时间：2010-04-19 16:13:05
-- 编写者:
----------------------------------
-- 修改记录:
-- 编号   修改时间       修改人       修改原因       修改标注
----------------------------------*/
CREATE PROCEDURE [dbo].[DataProxy_NIDC_log_FriendIP_BatchInsert]
(
   @xml    varchar(max)    = NULL       -- 数据
)
AS
   SET NOCOUNT ON;
   DECLARE @TableName nvarchar(100)
   DECLARE @CreateTableSQL nvarchar(max)
   DECLARE @InsertSQL nvarchar(max)
   DECLARE @UpdateSQL nvarchar(max)
   DECLARE @Suffix nvarchar(100)

   DECLARE @docHandle int

   CREATE TABLE #TEMP (
         [MyCyberID] [bigint]  NULL,
         [FriendCyberID] [bigint]  NULL,
         [IP]    nvarchar(50) NULL ,
         [ClientID] UNIQUEIDENTIFIER NULL DEFAULT NEWID(),
         [Time] [datetime] NULL DEFAULT(GETDATE()),
```

```
        [UnitID] [int]  NULL DEFAULT (0),
        [ClientNo] [int]  NULL DEFAULT (0),
        [CyberType] [int] NULL,
        [FriendCyberCode] [nvarchar] (128) NULL
    )

EXEC sp_xml_preparedocument @docHandle OUTPUT, @Xml
INSERT #TEMP
SELECT
     [MyCyberID]
    ,[FriendCyberID]
    ,[IP]
    ,[ClientID]
    ,MAX([Time])
    ,[UnitID]
    ,MAX([ClientNo])
    ,MAX([CyberType])
    ,MAX([FriendCyberCode])
FROM OPENXML(@docHandle, N'/ROOT/ROW')
    WITH (
     [MyCyberID] [bigint]
    ,[FriendCyberID] [bigint]
    ,[IP]    nvarchar(50)
    ,[ClientID] UNIQUEIDENTIFIER
    ,[Time] [datetime]
    ,[UnitID] [int]
    ,[ClientNo] [int]
    ,[CyberType] [int]
    ,[FriendCyberCode] [nvarchar] (128)
    )
WHERE [MyCyberID] <> 0 AND [FriendCyberID] <> 0
AND ClientID <> '00000000-0000-0000-0000-000000000000'    AND[IP] <> ''
GROUP BY
     [MyCyberID]
    ,[FriendCyberID]
    ,[IP]
    ,[ClientID]
    ,[UnitID]
EXEC sp_xml_removedocument @docHandle

SELECT
     @Suffix = CONVERT(VARCHAR(8),Time,112)
    ,@TableName = N'[NIDCLOG' + LEFT(@Suffix,6) + '].[dbo].[NIDC_log_FriendIP' + @Suffix + ']'
    ,@CreateTableSQL = ISNULL(@CreateTableSQL ,N'
```

```
    DECLARE @TableName nvarchar(200);') + N'
    EXEC [dbo].[DataProxy_NIDC_log_FriendIP_CreateTable] @TableName OUTPUT ,''' + @Suffix + ''';'
    ,@UpdateSQL = ISNULL(@UpdateSQL,N'') + N'
    UPDATE D SET
        CyberType = T.CyberType
        ,[FriendCyberCode] = T.[FriendCyberCode]
        ,[ClientNo] = T.[ClientNo]
    FROM ' + @TableName + ' D
    ,#TEMP T
    WHERE D.UnitID = T.UnitID
    AND D.ClientID = T.ClientID
    AND D.[MyCyberID] = T.[MyCyberID]
    AND D.[FriendCyberID] = T.[FriendCyberID]
    AND D.[IP] = T.[IP]
    AND T.Time >= ''' + @Suffix + '''
    AND T.Time < DATEADD(DAY,1,''' + @Suffix + ''')
    '
    ,@InsertSQL = ISNULL(@InsertSQL,N'') + N'
    INSERT ' + @TableName + ' (
        [MyCyberID]
        ,[FriendCyberID]
        ,[IP]
        ,[ClientID]
        ,[Time]
        ,[UnitID]
        ,[ClientNo]
        ,[CyberType]
        ,[FriendCyberCode]
        )
    SELECT
        T.[MyCyberID]
        ,T.[FriendCyberID]
        ,T.[IP]
        ,T.[ClientID]
        ,T.[Time]
        ,T.[UnitID]
        ,T.[ClientNo]
        ,T.[CyberType]
        ,T.[FriendCyberCode]
    FROM #TEMP T
    WHERE T.Time >= ''' + @Suffix + '''
    AND T.Time < DATEADD(DAY,1,''' + @Suffix + ''')
    AND NOT EXISTS (
        SELECT 1
```

```
        FROM ' + @TableName + ' D --WITH(NOLOCK)                --001
        WHERE D.UnitID = T.UnitID
        AND D.ClientID = T.ClientID
        AND D.[MyCyberID] = T.[MyCyberID]
        AND D.[FriendCyberID] = T.[FriendCyberID]
        AND D.[IP] = T.[IP]
        )
    '
FROM #TEMP
GROUP BY CONVERT(VARCHAR(8),Time,112)


-- 检查表是否存在，不存在则创建
IF ISNULL(@CreateTableSQL,N'') <> N''
    EXEC (@CreateTableSQL);

-- 更新顾客虚拟身份关系记录表
/*
IF ISNULL(@UpdateSQL,N'') <> N''
    EXEC (@UpdateSQL);
*/
-- 插入顾客虚拟身份关系记录表
IF ISNULL(@InsertSQL,N'') <> N''
    EXEC (@InsertSQL);
```

此存储过程的主要功能逻辑如下：

1. 创建一个临时表#TEMP；
2. 解析要导入的XML文件，XML中有多条record数据；
3. 导入XML数据记录到临时表#TEMP中，只有满足条件的记录才导入到TEMP表中。
4. 创建日志分库分表。获取当前系统时间，用NIDC_log_FriendIP + 日期得到表名称，用NIDCLOG+日期的年月 来得到数据库名，注意数据库不能自动创建，需要预先创建好。生成建表语句@CreateTableSQL。
5. 调用存储过程[dbo].[DataProxy_NIDC_log_FriendIP_CreateTable] 来自动创建表。
6. 从临时表TEMP中查询满足条件的记录，生成 INSERT 语句@InsertSQL。
7. 执行@CreateTableSQL 和 @InsertSQL 语句。
8. 到此，XML文件中的记录被插入到日志分库分表中。

11.2 移植方案

1. 由于TDS不支持XML导入，但是UDBTX支持XML的导入，因此XML解析和插入到临时表的工作可以放到UDBTX的函数中，在SQL Server存储过程中调用UDBTX封装的导入XML数据的函数来完成XML到表的处理。
2. 由于SQL Server存储过程本身也有大量的代码，这部分代码UDBTX-TDS本身还是大部分支持的，所以这部分不需要全部改为UDBTX的函数来实现，即尽量保留原有的存储过程代码。

11.3 移植后的代码

1. UDBTX 函数部分

CREATE OR REPLACE FUNCTION nidc_dbo.importXmlToTable_DataProxy_NIDC_log_FriendIP_BatchInsert(xml_path TEXT)
RETURNS INTEGER AS $$
DECLARE

```
data xml;
records XML[];
record XML;

MyCyberID bigint;
FriendCyberID bigint;
IP varchar(50);
ClientID varchar(64);
TimeS TIMESTAMP;
UnitID int;
ClientNo int;
CyberType int;
FriendCyberCode varchar;
BEGIN

-- Load the XML file
--data := pg_catalog.pg_read_binary_file(xml_path, 0, 10000000); -- Adjust the size accordingly
data := XMLPARSE(DOCUMENT convert_from(pg_catalog.pg_read_binary_file(xml_path), 'UTF8')); -- Adjust the size accordingly

-- Extract records from the XML data
records := xpath('/dataset/record', data); -- Assuming records are wrapped in <record> tags under a <root> element

-- Loop through each record
FOREACH record IN ARRAY records LOOP
    -- Extract the ID value using xpath
    MyCyberID := (xpath('//MyCyberID/text()', record))[1]::text::bigint;
    FriendCyberID := (xpath('//FriendCyberID/text()', record))[1]::text::bigint;
    IP := (xpath('//IP/text()', record))[1]::text;
    ClientID := (xpath('//ClientID/text()', record))[1]::text;
    TimeS := (xpath('//Time/text()', record))[1]::text::datetime;
    UnitID := (xpath('//UnitID/text()', record))[1]::text::int;
    ClientNo := (xpath('//ClientNo/text()', record))[1]::text::int;
    CyberType := (xpath('//CyberType/text()', record))[1]::text::int;
    FriendCyberCode := (xpath('//FriendCyberCode/text()', record))[1]::text;

    -- Check if ID is greater than 100
```

```
37.
38.     IF MyCyberID <> 0 AND FriendCyberID <> 0 AND ClientID <> '00000000-0000-0000-0000-
    000000000000' AND IP <> '' THEN
39.
40.         -- Build the INSERT query for the record
41.         INSERT INTO TEMP (MyCyberID, FriendCyberID, IP, ClientID, Time, UnitID, ClientNo, CyberType,
    FriendCyberCode)
42.             VALUES (MyCyberID, FriendCyberID, IP, ClientID, TimeS, UnitID, ClientNo, CyberType,
    FriendCyberCode);
43.
44.     END IF;
45. END LOOP;
46.
47. RETURN 0;
48. END;
49. $$ LANGUAGE plpgsql;
```

上述函数的功能是把指定的XML格式的数据保存到临时表TEMP中，需要注意的是临时表在SQL Server存储过程中创建和删除。

此函数不太容易写成通用的函数，因为每个存储过程入库时有各种条件判断，根据客户现有的情况，针对每个OPENXML的存储过程定制写一个，函数的逻辑基本相同。

上述函数中用到了pg_read_binary_file 系统函数，此函数需要额外授权。授权命令如下（注意不是授权给 sa用户，而是nidc_dbo）：

```
GRANT EXECUTE ON FUNCTION pg_read_binary_file(text,bigint,bigint,boolean) TO nidc_dbo;
GRANT EXECUTE ON FUNCTION pg_read_binary_file(text,bigint,bigint) TO nidc_dbo;
GRANT EXECUTE ON FUNCTION pg_read_binary_file(text) TO nidc_dbo;
```

2. 存储过程部分

```
1. CREATE PROCEDURE [dbo].[DataProxy_NIDC_log_FriendIP_BatchInsert]
2. (
3.     @xml     varchar(max)     = NULL         -- 数据
4. )
5. AS
6.     SET NOCOUNT ON;
7.     DECLARE @TableName nvarchar(100)
8.     DECLARE @CreateTableSQL nvarchar(max)
9.     DECLARE @InsertSQL nvarchar(max)
10.    DECLARE @UpdateSQL nvarchar(max)
11.    DECLARE @Suffix nvarchar(100)
12.
13.    DROP TABLE IF EXISTS TEMP;
```

```
14.
15.  CREATE TABLE TEMP (
16.          [MyCyberID] [bigint]  NULL,
17.          [FriendCyberID] [bigint]  NULL,
18.          [IP]    nvarchar(50) NULL ,
19.          [ClientID] UNIQUEIDENTIFIER NULL DEFAULT NEWID(),
20.          [Time] [datetime] NULL DEFAULT(GETDATE()),
21.          [UnitID] [int]  NULL DEFAULT (0),
22.          [ClientNo] [int]  NULL DEFAULT (0),
23.          [CyberType] [int] NULL,
24.          [FriendCyberCode] [nvarchar] (128) NULL
25.      );
26.
27.  SELECT [dbo].[importXmlToTable_DataProxy_NIDC_log_FriendIP_BatchInsert](@xml);
28.
29.  SELECT @Suffix = CONVERT(VARCHAR(8),GETDATE(),112)
30.
31.  SELECT @TableName = N'[NIDCLOG' + LEFT(@Suffix,6) + '].[dbo].[NIDC_log_FriendIP' + @Suffix + ']'
32.
33.  SELECT
34.      -- @Suffix = CONVERT(VARCHAR(8),Time,112)
35.      --,@TableName = N'[NIDCLOG' + LEFT(@Suffix,6) + '].[dbo].[NIDC_log_FriendIP' + @Suffix + ']'
36.      --,@CreateTableSQL = ISNULL(@CreateTableSQL ,N'
37.      @CreateTableSQL = ISNULL(@CreateTableSQL ,N'
38.      DECLARE @TableName nvarchar(200);') + N'
39.      EXEC [dbo].[DataProxy_NIDC_log_FriendIP_CreateTable] @TableName OUTPUT ,''' + @Suffix + ''';'
40.      ,@UpdateSQL = ISNULL(@UpdateSQL,N'') + N'
41.      UPDATE D SET
42.          CyberType = T.CyberType
43.          ,[FriendCyberCode] = T.[FriendCyberCode]
44.          ,[ClientNo] = T.[ClientNo]
45.      FROM ' + @TableName + ' D
46.      ,TEMP T
47.      WHERE D.UnitID = T.UnitID
48.      AND D.ClientID = T.ClientID
49.      AND D.[MyCyberID] = T.[MyCyberID]
50.      AND D.[FriendCyberID] = T.[FriendCyberID]
51.      AND D.[IP] = T.[IP]
52.      AND T.Time >= @Suffix
53.      AND T.Time < DATEADD(DAY,1, @Suffix)
54.      '
55.      ,@InsertSQL = ISNULL(@InsertSQL,N'') + N'
56.      INSERT ' + @TableName + ' (
57.          [MyCyberID]
```

```
58.          ,[FriendCyberID]
59.          ,[IP]
60.          ,[ClientID]
61.          ,[Time]
62.          ,[UnitID]
63.          ,[ClientNo]
64.          ,[CyberType]
65.          ,[FriendCyberCode]
66.          )
67.      SELECT
68.          T.[MyCyberID]
69.          ,T.[FriendCyberID]
70.          ,T.[IP]
71.          ,T.[ClientID]
72.          ,T.[Time]
73.          ,T.[UnitID]
74.          ,T.[ClientNo]
75.          ,T.[CyberType]
76.          ,T.[FriendCyberCode]
77.      FROM TEMP T
78.      WHERE T.Time >= ''' + @Suffix + '''
79.      AND T.Time < DATEADD(DAY,1,''' + @Suffix + ''')
80.      AND NOT EXISTS (
81.          SELECT 1
82.          FROM ' + @TableName + ' D --WITH(NOLOCK)                --001
83.          WHERE D.UnitID = T.UnitID
84.          AND D.ClientID = T.ClientID
85.          AND D.[MyCyberID] = T.[MyCyberID]
86.          AND D.[FriendCyberID] = T.[FriendCyberID]
87.          AND D.[IP] = T.[IP]
88.          )
89.      '
90.  FROM TEMP
91.  --GROUP BY CONVERT(VARCHAR(8),Time,112)
92.  GROUP BY CONVERT(VARCHAR(8),GETDATE(),112)
93.
94.  -- 检查表是否存在，不存在则创建
95.  IF ISNULL(@CreateTableSQL,N'') <> N''
96.      EXEC (@CreateTableSQL);
97.
98.  -- 更新顾客虚拟身份关系记录表
99.  /*
100. IF ISNULL(@UpdateSQL,N'') <> N''
101.     EXEC (@UpdateSQL);
```

```
    */
    -- 插入顾客虚拟身份关系记录表
    IF ISNULL(@InsertSQL,N'') <> N''
        EXEC (@InsertSQL);
```

上述存储过程是在原有代码的基础上做了稍微做了一些修改而来：

1. 创建临时表：由于SQL Server的临时表表名必须是#开头，比如#TEMP，而UDBTX不允许表名中包含特殊字符。因此无法与SQL Server兼容。所以在存储过程中，只能使用普通表，在存储过程执行完成时显式的删除表。
2. 删除存储过程中原有的处理XML，并插入到临时表TEMP的代码。改为调用UDBTX的函数SELECT [dbo].importXmlToTable_DataProxy_NIDC_log_FriendIP_BatchInsert;
3. 修改构造@CreateTableSQL、@UpdateSQL 和 @InsertSQL 语句的代码。此部分代码的执行结果与SQL Server不一致，应该是运算的优先级问题，原有的代码运行结果@Suffix、@TableName为NULL，解决方法是：把拼接语句中用到的变量放到前面单独的语句中，先进行赋值。如下：
   SELECT @Suffix = CONVERT(VARCHAR(8),GETDATE(),112)
   SELECT @TableName = N'[NIDCLOG' + LEFT(@Suffix,6) + '].[dbo].[NIDC_log_FriendIP' + @Suffix + ']'

关于此类问题，可能需要对所有的存储过程进行测试验证，存在问题的都需要调整。

11.4 执行结果

1. XML样例

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dataset>
    <record>
        <MyCyberID>2</MyCyberID>
        <FriendCyberID>2</FriendCyberID>
        <IP>192.168.1.10</IP>
        <ClientID>A4156412CB6A4F3FA68C88941EDB2B1C</ClientID>
        <Time>2024-03-29 10:10:10</Time>
        <UnitID>2</UnitID>
        <ClientNo>2</ClientNo>
        <CyberType>2</CyberType>
        <FriendCyberCode>Friend cyber code</FriendCyberCode>
    </record>
    <record>
        <MyCyberID>3</MyCyberID>
        <FriendCyberID>3</FriendCyberID>
        <IP>192.168.1.10</IP>
        <ClientID>A4156412CB6A4F3FA68C88941EDB2B1C</ClientID>
        <Time>2024-03-29 10:10:10</Time>
        <UnitID>3</UnitID>
        <ClientNo>3</ClientNo>
        <CyberType>3</CyberType>
```

```
            <FriendCyberCode>Friend cyber code</FriendCyberCode>
        </record>
</dataset>
```

2. 插入到TEMP的数据
   （略）

3. 插入到分库分表中的数据
   （略）

11.5 总结

1. OPENXML 相关的存储过程，功能都差不多，基本上按照上述思路可以解决。按照此方案进行移植，可以保持存储过程的接口不变，现有的应用软件也不修要修改。
2. 在移植过程中，也发现UDBTX-TDS存在语句执行结果与原有的语句执行结果不一致的问题，所以所有存储过程都需要进行测试验证，不一致的地方要修改。

12 UDBTX-TDS 迁移 SQL Server 实践

12.1 前言

从传统的 SQL Server 数据库迁移可能非常耗时且需耗费大量资源，任何迁移都涉及三个主要步骤：移动架构、迁移数据和修改客户端应用程序。正如下图中我们所见：迁移数据库时，通常需要完成更多的工作，包括重写与数据库交互的应用程序代码，将 T-SQL 代码迁移到 PL/pgSQL 中，这是复杂、耗时且有风险的。

UDBTX-TDS 是 UDBTX 兼容版本的一项新功能，可以理解 Microsoft SQL Server 专有的 SQL 语言 T-SQL，并支持相同的通信协议，因此，修改 SQL Server 上运行的应用程序并将其移动到 UDBTX 所需的工作量将减少，从而可实现更快、风险更低且更具成本效益的迁移。

UDBTX-TDS 通过支持 UDBTX 的 Microsoft SQL Server 数据类型、语法和函数来支持 T-SQL 和 SQL Server 行为。但请注意，UDBTX-TDS 并不提供对 T-SQL 的100%完整支持，仍然有一些差异和限制，某些情况下需要做手工的代码转换。

本文将列举并演示一些高频及常见的典型代码转换案例，帮助您更高效快速地完成迁移工作。

12.2 环境准备

在开始我们的演示之前，假设在您的工作环境，已有一个准备迁移的 SQL Server 源库，那么除此之外，您还需要设置好以下相关的组件：

udbtxtds_compass

这是一个SQL Server 迁移到 UDBTX-TDS 的语法评估工具，它能在Windows和Linux平台下运行，需要 Java 环境支持，当前的版本是 v2024-04

UDBTX-TDS

您可以根据官方文档说明来操作，只需简单几步即可创建一个 UDBTX-TDS 集群环境。配置过程中需要注意的就是数据库迁移模式的选择，还有如果有中文数据的话那么在排序规则中请选择“chinese_prc_ci_as”

到目前为止，一个包含SQL Server源和UDBTX-TDS目标以及迁移评估工具的环境已经准备好。接下来，请参考本文的内容，您只需要花短短的几分钟就能生成一个UDBTX-TDS迁移评估报告。

12.3 代码转换

12.3.1 转换评估

udbtxtds_compass 工具生成的评估报告是评估迁移工作内容和工作量的指引，您可以根据其中列出的需要修改的项目，逐一编写 SQL 代码转换内容。

评估报告的Summary 章节列出了迁移 SQL Server 源到 UDBTX-TDS 目标的 T-SQL 的语法特性兼容统计，包括支持、不支持、语义审查、手动审查及可忽略项。其中最关键的是不支持特性的内容，这些含有不支持特性的 SQL 语句，如果不作修改，在 UDBTX-TDS 环境中大部分执行会报错：“‘???’ is not currently supported in UDBTX-TDS”，而其他的一些 SQL 语句虽然没有报错，但不会真正生效。

在评估报告中我们可以查看这些不支持特性的 SQL 分类统计，下图中的评估报告列出了每一类不支持特性的 SQL 语句，它显示了我们的案例所用的 DDL 脚本在 Babelfish 中不支持的特性主要有对表增加约束语句，Merge 语句、修改数据库、修改角色、执行某些系统存储过程等。

12.3.2 转换原则

UDBTX-TDS 为 UDBTX 数据库集群提供了一个额外的端点，使其能够了解 SQL Server 线路级协议和常用的 SQL Server 语句。迁移之后，您仍然可以使用相同的 T-SQL 开发工具和驱动，连接到TDS端口完成相关的开发。您也可以使用原生 UDBTX 连接在 UDBTX 这一端做开发，再从 T-SQL 这端进行调用。这一种兼容模式，能帮助我们解决大部分的UDBTX-TDS 对 T-SQL 的兼容性问题。

选择转换模式：如上所述，对于部分不支持的 SQL 语句，我们可以选择在 T-SQL 中进行改写，也可以在 UDBTX 中修改再从 T-SQL 中调用。转换的原则是根据应用的连接开发模式而定，例如 .net 应用连接到 TDS 端开发，那么首选转换模式就是在T-SQL中进行转换。如果在T-SQL这端无法改写或存在修改后的性能问题，那么可以尝试在 UDBTX 中进行修改。

代码可读性：对于要修改的SQL语句，可能有好几种的改写方法。简单、高效、可读性好永远都是首选。例如，大部分情况下，使用Case语句比使用..Then更容易理解。

12.3.3 简单代码转换

归于这一类的代码转换，其特点就是修改简单，但其数量常常在评估报告中列出的所有不支持特性的 SQL 语句中占绝大部分。此类代码转换工作一般而言只需屏蔽相关选项、注释整条语句或简单修改即可。如此修改的原因，是缘于 UDBTX 和 SQL Server 的两者间的特性差异或 UDBTX-TDS 的限制。SQL Server 中的某些选项或操作，在 UDBTX-TDS 不支持且不会对功能执行有影响，可以直接忽略。虽然这类 SQL 语句改写简单，但能达到了相同的效果。

演示之前，让我们看看接下来都会使用到两张表的结构：

```
1. create table dept(
2.     deptno int NOT NULL PRIMARY KEY,
3.     dname varchar(14),
```

```
    loc varchar(13)
)

create table employees (
    empno int NOT NULL PRIMARY KEY,
    ename varchar(10),
    job varchar(9),
    mgr int,
    hiredate datetime,
    sal money,
    comm money,
    deptno int
)
```

ALTER TABLE..CHECK CONSTRAINT

原语句

```
ALTER TABLE [dbo].[employees]  WITH CHECK ADD CONSTRAINT [FK_DEPT] FOREIGN KEY([deptno])
REFERENCES [dbo].[dept] ([deptno])
GO
ALTER TABLE [dbo].[Employees] CHECK CONSTRAINT [FK_DEPT]
GO
```

修改后语句

```
ALTER TABLE [dbo].[employees] ADD CONSTRAINT [FK_DEPT] FOREIGN KEY([deptno])
REFERENCES [dbo].[dept] ([deptno])
GO
```

说明：

在 UDBTX-TDS 中不支持 CHECK CONSTRAINT 语句启用表的约束，ALTER 表添加约束后会自动启用约束

在 UDBTX-TDS 中添加约束时不支持 WITH CHECK/NOHECK 选项对已有数据进行约束检查

这种不支持的 ALTER TABLE 特性的语句是迁移过程中最常见的，一般是修改后在 UDBTX-TDS 上新建表和约束，再导入表的数据，表的约束会自动检查导入的数据，保证数据约束有效

ALTER ROLE..

原语句

```
ALTER ROLE [???] ADD MEMBER [NT AUTHORITY\SYSTEM]
GO
```

修改后语句（注释掉）

```
/* ALTER ROLE [???] ADD MEMBER [NT AUTHORITY\SYSTEM]
GO */
```

说明：

目前 UDBTX-TDS 只支持用户数据库中的 dbo 用户，您不能创建具有较低权限的用户，例如对某些表的只读权限

大部分此类语句都是用户以操作系统权限登陆 SQL Server源后倒出的 DDL 语句，可以直接注释屏蔽语句

ALTER DATABASE..

原语句

```
ALTER DATABASE [???] SET RECOVERY FULL
GO
```

修改后语句

```
/* ALTER DATABASE [???] SET RECOVERY FULL
GO */
```

说明：

UDBTX-TDS 不支持ALTER DATABASE 语法，Aurora PostgreSQL是一个全托管型数据库，会限制一些数据库修改语句，这些语句可以直接注释屏蔽

ALTER AUTHORIZATION ON object

原语句

```
ALTER AUTHORIZATION ON [dbo].[employees] TO  SCHEMA OWNER
GO
```

修改后语句

```
/* ALTER AUTHORIZATION ON [dbo].[employees] TO  SCHEMA OWNER
GO */
```

说明：

UDBTX-TDS 不支持AUTHORIZATION的创建、修改和删除，可以直接注释屏蔽

EXEC sys.sp_addextendedproperty

原语句

```
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'编号' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',@level1name=N'dept',
@level2type=N'COLUMN',@level2name=N'deptno'
GO
```

修改后语句（在 UDBTX 端修改）

```
COMMENT ON COLUMN dept.deptno IS '编号';
```

说明：

UDBTX-TDS 不支持使用系统存储过程 sp_addextendedproperty 为字段增加说明，可以直接注释屏蔽此 SQL 语句，并连接到 UDBTX 端使用comment增加字段说明

OBJECTPROPERTY

原语句

```
select name from sysobjects where objectproperty(id, N'IsTable') = 1 and name not like N'#%%' order by name
select * from sysobjects where id = object_id(N'temp_tableSpaceInfo') AND objectproperty(id, N'IsUserTable') =
1
```

修改后语句

```
select name from sysobjects where xtype in ('U','IT','S') and name not like N'#%%' order by name
select * from sysobjects where id = object_id(N'temp_tableSpaceInfo') AND xtype='U'
```

说明：

UDBTX-TDS 不支持内置的元数据函数 OBJECTPROPERTYEX，可根据SQL语义进行适当改写

SET ROWCOUNT

原语句

```
CREATE PROCEDURE [dbo].[P_Rowcount]
@id int
as
set nocount on
set rowcount @id
```

```
begin
select * from employees order by empno
end
GO
```

修改后语句

```
CREATE PROCEDURE [dbo].[P_Rowcount]
@id int
as
set nocount on
begin
select top (@id) * from employees order by empno
end
GO
```

第二种修改

```
CREATE PROCEDURE [dbo].[P_Rowcount]
@id int
as
set nocount on
begin
select * from employees order by empno offset 0 rows fetch first @id rows only;
end
GO
```

说明：

UDBTX-TDS 不支持 SET ROWCOUNT 语句来返回指定的行数，可根据 SQL 语义进行适当改写。从示例中我们看到可以有多种的改写方法，在业务复杂的场景下应从代码的可读性和性能影响方面做选择

CURRENT OF

原语句

```
CREATE PROCEDURE [dbo].[P_CurrentOf] AS
BEGIN
  DECLARE @empno int
  DECLARE NoResponce CURSOR FOR
    SELECT empno FROM employees;
  OPEN NoResponce;
  FETCH NEXT FROM NoResponce INTO @empno;
  DELETE FROM employees WHERE CURRENT OF NoResponce;
```

```
END
GO
```

修改后语句

```
CREATE PROCEDURE [dbo].[P_CurrentOf] AS
BEGIN
  DECLARE @empno int
  DECLARE NoResponce CURSOR FOR
    SELECT empno FROM employees;
  OPEN NoResponce;
  FETCH NEXT FROM NoResponce INTO @empno;
  DELETE FROM employees WHERE empno = @empno;
END
GO
```

备注:

Where Current Of 语句允许您更新或者是删除最后由 cursor 取的记录，Babelfish 不支持 Current Of 语句，可根据 SQL 语句上下文语义选取变量

IDENTITY

原语句

```
SELECT IDENTITY(INT,1,1) AS rowid,* INTO #tmp
FROM employees
ORDER BY empno
```

修改后语句

```
SELECT row_number() over () as rowid, * INTO #tmp
FROM employees
ORDER BY empno
```

说明：

UDBTX-TDS 不支持 IDENTITY 函数，用于在带有 INTO 子句的 SELECT 语句中将标识列插入到新表中，可使用 row_number() over ()方式改写

3.4 复杂代码转换

相对于前面介绍简单代码转换，接下来的这些 SQL 语句会复杂一些，修改内容也比较多。同时，您还需要仔细地审查 SQL 语句中上下文之间的关系，以确保修改后的语句和原语句执行得到相同的效果。

MERGE

在这个案例演示之前，创建两张 MERGE 使用的源表和目标表

```
create table source
(
  id      int not null primary key ,
  country varchar(20) null,
  city    varchar(20)
);

insert into source
 (id, country, city)
 VALUES
 (1, 'RUSSIA',  'MOSCOW'),
 (2, 'FRANCE',  'PARIS'),
 (3, 'ENGLAND', 'LONDON'),
 (4, 'USA',     'NEW YORK'),
 (5, 'GERMANY', 'BERLIN'),
 (6, 'BRAZIL',  'BRASILIA');

create table target
(
  id      int not null primary key ,
  country varchar(20) null,
  city    varchar(20)
);

insert into target
 (id, country, city)
 VALUES
 (1, 'JAPAN',   'TOKYO'),
 (4, 'USA',     'DENVER'),
 (7, 'CHINA',   'BEI JING');
```

原语句

```
MERGE INTO target AS C2
USING source AS C1
ON C2.id = C1.id
WHEN MATCHED
  THEN UPDATE
    SET
      C2.country = C1.country,
```

```
        C2.city = c1.city
WHEN NOT MATCHED
  THEN INSERT (id, country, city)
          VALUES (C1.id, C1.country, C1.city);
```

修改后语句

```
begin
update target set country = C1.country, city = C1.city from (select id, country, city from source) C1 where target.id = C1.id;
insert into target (id, country, city) select * from source as C1 where not exists (select id from target where id = C1.id);
end
go
```

第二种修改（在UDBTX端修改）

```
with upsert as
(update target c2 set country=c1.country, city=c1.city
 from source c1 where c1.id=c2.id
  RETURNING c2.*
)
insert into target select a.id, a.country, a.city
from source a where a.id not in (select b.id from upsert b);
```

说明：

MERGE 是常用的一种数据合并更新语句，Babelfish 不支持 MERGE 语句，一般来说可根据 SQL 语义在 T-SQL 中拆分成多个 DML 语句，也可以在 PostgreSQL 端进行等价的改写。

UDBTX 目前还不支持 MERGE 语句，可以使用 UPSET 或 CONFLICT 语句实现，INSERT ON CONFLICT 的执行开销要小于 UPDATE 语句

FULLTEXT 全文搜索

UDBTX-TDS 不支持 SQL Server的全文搜索，不支持以下的语句及系统存储过程

```
CREATE、ALTER、DROP FULLTEXT CATALOG
CREATE、ALTER、DROP FULLTEXT INDEX
CREATE、ALTER、DROP FULLTEXT STOPLIST
exec sp_fulltext_database 'enable';
```

UDBTX 增加了对 pg_bigm 扩展程序的支持。pg_bigm 扩展程序在 UDBTX-TDS 中提供有全文搜索功能。此扩展程序允许用户创建 2-gram（双组），以提高全文搜索速度。以下案例演示如何在UDBTX端通过扩展启用全文搜索功能

```
set search_path=dbo;
create extension pg_bigm;

CREATE TABLE fulltext_doc (doc text);
INSERT INTO fulltext_doc VALUES('Babelfish助力SQL迁移 成本优化');
INSERT INTO fulltext_doc VALUES('Babelfish助力SQL迁移 性能优化');
INSERT INTO fulltext_doc VALUES('Babelfish助力SQL迁移 提升使用体验');
INSERT INTO fulltext_doc VALUES('Babelfish助力SQL迁移 中提供 2-gram 全文搜索功能的工具');
INSERT INTO fulltext_doc VALUES('Babelfish助力SQL迁移 中提供 3-gram 全文搜索功能的工具');

CREATE INDEX fulltext_doc_idx ON fulltext_doc USING gin (doc gin_bigm_ops);
alter table fulltext_doc owner to dbo;
```

全文搜索设置成功后可以在 TDS 端口通过 T-SQL 调用

SWITCHOFFSET

原语句

SELECT CAST(SWITCHOFFSET(TODATETIMEOFFSET(SYSUTCDATETIME(),'+00:00'),'+08:00') AS DATETIME)

修改后语句（在 UDBTX 端创建自定义函数）

```
CREATE OR REPLACE FUNCTION dbo.f_get_cst()
RETURNS sys.datetime AS $$
  BEGIN
      RETURN cast(timezone('Asia/Shanghai',now()) as sys.datetime);
END;
$$ LANGUAGE plpgsql;
```

说明：

UDBTX-TDS 不支持SWITCHOFFSET和TODATETIMEOFFSET之类的时区偏移量内置函数，可以在UDBTX端创建自定义函数并在TDS端口通过T-SQL调用来实现相同功能

XML 方法

在本案例演示之前，创建一张和 xml 解析相关的表

```
create table t_xml_test (
  id int,
  country  nvarchar(max),
  industry nvarchar(max)
);

```

```
insert t_xml_test values(1, 'China', 'Manufacturing and foreign trade business');
insert t_xml_test values(2, 'USA', 'Financial and Bioindustry');
insert t_xml_test values(3, 'Russia', 'Resource export');
```

原语句

```
create procedure p_xml_test
 @xml xml
as
begin
  set nocount on
  select * from t_xml_test
  where id in (select imgXML.Item.value('id[1]','int') from @xml.nodes('/root/country') as imgXML(Item));
  set nocount off
end
go
```

修改后语句（首先在 UDBTX 端创建自定义函数解析 XML）

```
CREATE OR REPLACE FUNCTION xmlQueryID(in_xml xml)
RETURNS TABLE (id text)
AS $$
DECLARE
BEGIN
  RETURN QUERY
    select * from (
      WITH xmldata(data) AS (VALUES (in_xml::xml))
      SELECT xmltable.*
      FROM XMLTABLE('/root/country' PASSING (SELECT data FROM xmldata) COLUMNS id text)) as foo;
END;
$$ LANGUAGE plpgsql;
```

连接 TDS 端口在 T-SQL 中修改 SQL 语句并调用 UDBTX 端创建的自定义函数

```
create procedure p_xml_test
 @xml xml
as
begin
  set nocount on
  select * from t_xml_test
  where id in (select * from xmlQueryID(@xml)) ;
  set nocount off
end
```

```
10. go
```

在 T-SQL 中调用存储过程测试，查询结果显示 xml 解析正常，数据显示正确

说明：

UDBTX-TDS 不支持解析 XML 数据的方法，包括 VALUES、XML.NODES 和其他方法，可以在 UDBTX 端创建自定义函数并在 TDS 端口通过 T-SQL 调用来完成 XML 数据的解析工作

12.4 总结

通过前面的案例介绍，我们为您展示了使用 UDBTX-TDS 迁移 SQL Server时一些最常见的不支持特性 SQL 的转换方法。当前，UDBTX-TDS续向前发展，版本在不断更新。每个新的版本都会添加一些重要的功能，包括增加语法的兼容和 SQL Server 原生功能的支持。建议您在规划和实施 SQL Server 迁移时经常检查 UDBTX-TDS 的特性支持说明，使用最新的特性支持来完成代码的转换。