Teradata Transparency Series API

User Guide

Release 15.00
B035-2419-034K
March 2014

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS-IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. IN NO EVENT WILL TERADATA CORPORATION BE LIABLE FOR ANY INDIRECT, DIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS OR LOST SAVINGS, EVEN IF EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The information contained in this document may contain references or cross-references to features, functions, products, or services that are not announced or available in your country. Such references do not imply that Teradata Corporation intends to announce such features, functions, products, or services in your country. Please consult your local Teradata Corporation representative for those features, functions, products, or services available in your country.

Information contained in this document may contain technical inaccuracies or typographical errors. Information may be changed or updated without notice. Teradata Corporation may also make improvements or changes in the products or services described in this information at any time without notice.

To maintain the quality of our products and services, we would like your comments on the accuracy, clarity, organization, and value of this document. Please email: teradata-books@lists.teradata.com.

Any comments or materials (collectively referred to as "Feedback") sent to Teradata Corporation will be deemed non-confidential. Teradata Corporation will have no obligation of any kind with respect to Feedback and will be free to use, reproduce, disclose, exhibit, display, transform, create derivative works of, and distribute the Feedback and derivative works thereof without limitation on a royalty-free basis. Further, Teradata Corporation will be free to use any ideas, concepts, know-how, or techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, or marketing products or services incorporating Feedback.

Table of Contents

Preface 11

CHAPTER 5

Problem Management 40

CHAPTER 6

Static SQL and System Catalog Support 48

CHAPTER 7

Command Syntax 57

Preface

Purpose

This book provides information about Transparency Series/Application Program Interface (TS/API), which is a Teradata® Tools and Utilities product. This book provides information about the product and its components, and it describes the operational functions and features of the product. Teradata Tools and Utilities is a group of products designed to work with Teradata Database.

The TS/API application program provides access to relational databases stored on Teradata Database via the QMF product, which is designed to retrieve data stored in DB2 databases.

TS/API intercepts database requests from QMF and passes them to Teradata Database instead of to DB2. Data and error information are returned to QMF in the same format used by DB2.

Audience

This book is intended for use by:

- Users of QMF
- System operators and other database specialists

Supported Releases

This book supports the following releases:

- Teradata Database 15.00.00.00
- Teradata Tools and Utilities 15.00.00.00
- TS/API 15.00.00.00

To locate detailed supported-release information:

1. Go to http://www.info.teradata.com.
2. Under **Online Publications**, click **General Search**.
3. Type `3119` in the **Publication Product ID** box.
4. Under **Sort By**, select **Date**.
5. Click **Search**.
6. Open the version of the *Teradata Tools and Utilities ##.##.## Supported Platforms and Product Versions* spreadheet associated with this release.

Prerequisites

The following prerequisite knowledge is required for this product:

- Basic concepts of the Teradata Database
- IBM systems concepts and terminology for z/OS, CICS, or TSO
- DB2 concepts and terminology
- Teradata SQL concepts and terminology
- QMF concepts and terminology

Changes to this Book

The following changes were made to this book in support of the current release. Changes are marked with change bars. For a complete list of changes to the product, see the Teradata Tools and Utilities Release Definition associated with this release.

Date/Release	Description
March 2014 15.00	Release number, version numbers updated. No content changes.

Additional Information

Additional information that supports this product and Teradata Tools and Utilities is available at the web sites listed in the table that follows.

Type of Information	Description	Access to Information
Release overview, late information	Use the Release Definition for the following information: Overview of all of the products in the release. Information received too late to be lincluded in the manuals. Operating sytems and Teradata Database versions that are certified to work with each product. Version numbers of each product and the documentaiton for each product. Information about available training and the Support Center.	To access information: 1. Go to http://www.info.teradata.com/. 2. Under **Online Publications**, click **General Search**. 3. Type `2029` in the **Publication Product ID** box. 4. Click **Search.** 5. Select the appropriate Release Definitaion from the search results.
Additional product information	Use the Teradata Information Products Web site to view or download specific manuals that supply related or additional information to this manual.	To access information: 1. Go to http://www.info.teradata.com/. 2. Under **Online Publicationss** subcategory, click **Browse by Category**, click **Data Warehousing**. 3. Do one of the following: 4. For a list of Teradata Tools and Utilities documents, click **Teradata Tools and Utilities** and then select an item under **Releases** or **Products**. 5. Select a link to any of the data warehousing publication categories listed**.**
CD-ROM images	Access a link to a downloadable CD-ROM image of all customer documentation for this release. Customers are authorized to create CD-ROMs for their use from this image.	To access information: 1. Go to http://www.info.teradata.com/. 2. Under **Online Publicationss** subcategory, click **Browse by Category**, click **Data Warehousing**. 3. Click **CD-ROM Images**. 4. Follow the ordering instructions.

Ordering information for manuals	Use the Teradata Information Products web site to order printed versions of manuals.	To access information: 1. Go to http://www.info.teradata.com/. 2. Under **Print & CD Publications**, click **How to Order**. 3. Follow the ordering instructions.
General information about Teradata	The Teradata home page provides links to numerous sources of information about Teradata. Links include: Executive reports, case studies of customer experiences with Teradata, and thought leadership. Technical information, solutions, and expert advice. Press releases, mentions, and media resources.	To access information: 1. Go to http://www.teradata.com/. 2. Click a link.

List of Acronyms

The following acronyms are used in this book:

Acronym	Description
CICS	Customer Information Control System
DB2	DATABASE 2
SQL	Structured Query Language
TSO	Time Sharing Option
TS/API	Transparency Series/Application Program Interface
z/OS	IBM z-series Operating System

Product Safety Information

This document may contain information addressing product safety practices related to data or property damage, identified by the work Notice. A notice indicates a situation which, if not avoided, could result in damage to property, such as equipment or data, but not related to personal injury.

Example:

Notice : Improper use of the Reconfiguration utility can result in data loss.

CHAPTER 1

Introduction to TS/API

Overview

This chapter provides an overview of Transparency Series/Application Program Interface (TS/ API) and contains the following information:

- What is TS/API?
- Benefits of TS/API
- What TS/API Supports
- TS/API Usability
- TS/API Certified Products
- Accessing Teradata Database from QMF
- Query Management Facility (QMF)

TS/API provides gateway services allowing QMF, which normally accesses DB2 databases, to access data stored on Teradata Database. TS/API thus lets you take advantage of both the convenient and easy-to-use data access QMF product and the tremendous storage capability and superior processing power of Teradata Database.

TS/API works with Teradata Database version 2 (Teradata mode).

TS/API has been certified to support the following products:

- Query Management Facility (QMF), developed by IBM

See Query Management Facility (QMF) on page 21 for additional information.

What is TS/API?

TS/API is an Application Program Interface (API) that allows you to access relational databases stored on the Teradata Database via QMF, which is designed to retrieve data stored in DB2 databases. You do not need to know Teradata SQL in order to access this data.

TS/API intercepts database requests from QMF and passes them to Teradata Database instead of to DB2. Data and error information are returned to QMF in the same format used by DB2.

Benefits of TS/API

TS/API allows you to take advantage of the tremendous storage capability and processing power of Teradata Database and the Teradata hardware platform while using the features of QMF. You do not need to change your databases or QMF already in place to access them. TS/API provides ease of use coupled with the ability to fully exploit the advantages of Teradata Database. TS/API also supports Teradata SQL Extensions with the QMF passthrough facility.

What TS/API Supports

TS/API is intended for use with QMF operating in the following environments:

- MVS batch
- MVS/TSO
- MVS/CICS

TS/API has been certified to work with Query Management Facility (QMF) Releases 8.1 and 9.1.

TS/API has been certified to work with the following versions of Teradata Database:

- Version 2, Teradata 12.00 (Teradata mode only)
- Version 2, Teradata 13.00 (Teradata mode only)
- Version 2, Teradata 13.10 (Teradata mode only)
- Version 2, Teradata 14.00 (Teradata mode only)

QMF releases certified by Teradata have undergone rigorous function testing to ensure the integrity of all information being passed between QMF and Teradata Database.

Notice:

> While it is possible that other DB2-based program products or user-written DB2 application programs can access Teradata Database through TS/API, Teradata Corporation support for these products is limited at this time. Using TS/API with any software other than the products listed in this guide QMF is at the user's own risk.

Kanji Support

TS/API supports the following features of Kanji/Multi-byte Character Sets (MBCS):

- Character sets:
  - EBCDIC
  - KATAKANAEBCDIC
  - KANJIEBCDIC5026_0I
  - KANJIEBCDIC5035_0I
- Mixed MBCS/Single-byte Character Sets (SBCS) character strings as object names and literals.
- Hexadecimal notation for object names.
- GRAPHIC data types.

Note that all SQL keywords and TS/API directives must be coded in SBCS of the server's corresponding charset.

TS/API Usability

This section explains how TS/API functions and the advantages of using TS/API.

How TS/API Functions

To support QMF, TS/API performs a number of complex operations on SQL requests, which are passed to Teradata Database and on the data that is returned to the QMF. TS/API supports QMF by performing the following :

- Providing a mechanism for a QMF user to log on to Teradata Database
- Providing DB2 Call Attach Facility replacement modules
- Translating QMF queries to Teradata SQL
- Supporting the use of static SQL queries by storing DB2 plans as Teradata Database macros
- Converting incoming QMF data to Teradata Database format
- Converting outgoing Teradata Database data to a format usable by QMF
- Translating Teradata Database error codes to DB2 SQLCODEs and SQLSTATEs
- Creating SQLERRM text inserts from Teradata Database error messages
- Providing DB2 system catalog emulation views based on data stored in the Teradata Database system catalog
- Emulating DB2 unit of work logic
- Supporting updatable cursors

TS/API, DB2, and Application Tools (CICS)

Under the CICS environment, QMF, uses DB2's CICS Call Attach Facility module (DSNCLI) to access DB. TS/API replaces DSNCLI, and passes all DB2 requests to TS/API and thus to Teradata Database.

TS/API, DB2, and Application Tools (TSO)

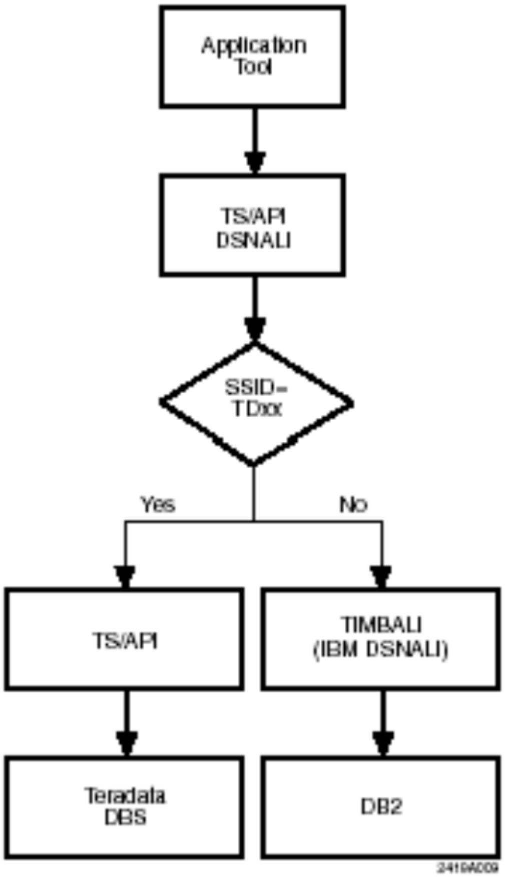
QMF uses DB2's Call Attach Facility module (DSNALI) to access DB2 under TSO. For QMF, TS/API provides a vectoring capability to access data in either DB2 or Teradata Database. To accomplish this vectoring, TS/API provides its own version of DSNALI.

The vectoring is based on the subsystem ID, as follows:

- DSNALI receives the SSID from Application Tools.
- DSNALI checks that SSID.
- If the SSID does not start with a prefix of TD, DSNALI calls IBM's original DSNALI (TIBMALI) and control goes to DB2.
- If the SSID starts with TD, control goes to TS/API and Teradata Database is accessed.

The following image, Flow Control in TSO , depicts the flow of control:

Flow Control in TSO



Supported Environments

The following environments are supported by TS/API:

- z/OS batch
- z/OS TSO
- z/OS CICS

Products that run in the CICS environment use the two-phase commit facility (2PC), which allows CICS (the transaction manager) to synchronize updates on different resources (databases, file systems, etc.). For additional details on the 2PC facility, see the *IBM CICS Interface for Teradata Reference Manual*.

TS/API Certified Products

This section provides brief descriptions of the products that TS/API has been certified to support. In addition, it provides procedures for accessing the Teradata Database with these certified products.

Certified Products

TS/API Release 14.00 has been certified to support:

- Query Management Facility (QMF)
- Version 8, Release 1 (z/OS batch, Z/OS TSO, Z/OS CICS)
- Version 9, Release 1 (z/OS batch, Z/OS TSO, Z/OS CICS)

To certify TS/API with QMF, Teradata performs extensive testing and quality assurance to ensure that TS/API works properly with QMF, except as specifically noted in this guide.

Once Teradata certifies a QMF release, it provides technical support, problem resolution, and software maintenance for TS/API as used with QMF.

The following section describes how TS/API is used with QMF.

Note: User familiarity withQMF is assumed. For detailed information about the operations of QMF as mentioned in this guide, see the appropriate vendor documentation.

Accessing Teradata Database from QMF

To ensure connection to a Teradata Database from QMF, verify the following before starting QMF:

1. TS/API libraries are properly allocated. TS/API load library should be concatenated before the DB2 load library, if any. Allocation occurs as part of either the TSO start-up procedure, the tool starting CLIST, or the tool starting JCL, depending on how you invoke the ISV product.
2. Your DBCLOGON data set or logon exit is properly prepared:
   - If your installation does not provide a logon exit, then you must build and allocate the DBCLOGON data set. See The DBCLOGON File/Table for details.
   - If your installation does provide a logon exit, then your userid and password are automatically provided to Teradata Database at logon time.
3. SSID (subsystem ID) is assigned an appropriate value. Under TSO, TS/API provides vectoring capabilities, allowing access to the data in either Teradata Database or DB2.
   - If you specify the SSID as TDxx, then Teradata Database will be accessed, using that `TDxx` as a <*tdpid*>.
   - If the SSID doesn't have a prefix of TD, then DB2 will be accessed.
     A connection error occurs if you do not have a DB2 installed.

For more information on vectoring, see Vectoring Enabled: Switching Between Teradata Database and DB2. The following are two examples illustrating JCL statements used to invoke QMF 9.1 to access TDP0:

```
QMF invocation:
...
//SYSTSIN   DD  *
PROFILE PREFIX(USERID)
```

```
ISPSTART PGM(DSQQMFE) NEWAPPL PARM(M=B,P=QMF310,S=TDP0)
/*
...
```

Under z/OS CICS

Verify the following:

- TS/API libraries are properly included in the DFHRPL libraries list in CICS start-up JCL. The TS/ API load libraries should be concatenated before the DB2 load library, if any.
- BIRCT and DBCLOGON tables contain appropriate information for the transaction ID you'll use. Under CICS, TS/API obtains logon information from these two tables. See Using Macros for BBIRCT and DBCLOGONUsing Macros for BBIRCT and DBCLOGON for details.

Query Management Facility (QMF)

A tool that has become preeminent in today's DB2 environment is IBM's Query Management Facility (QMF). QMF has emerged as the *de facto* product used to access relational data in order to produce reports, graphs, and charts, either through native SQL commands, prompted queries, or queries by example. Through TS/API, QMF can be used to access information stored in Teradata Database.

The following are examples of some QMF issues:

- Although TS/API supports QMF 8.1 and 9.1, because of the differences between Teradata Database and DB2, it does not support the Table Editor function for default column definitions on a z/OS TSO platform.
- TS/API supports the QMF Edit Table Facility (ETF) in both edit and browse mode. TS/API recognizes when a non-updatable table or view is being accessed with ETF, and automatically places ETF in browse mode. This allows you to view data but not to update or delete it.

For more information on updatable cursors, see Updatable Cursor Support on page 58.

TS/API does not support the SAVE=IMMEDIATE option under the QMF ETF because Teradata SQL does not support the WITH HOLD option of the DECLARE CURSOR statement. TS/API performs a commit with each SAVE and then performs clean up for the given ETF session, thereby removing the cursor.

Removal of the cursor means that you cannot fetch the next row; therefore, the query must be reexecuted to bring the result table back into memory. The QMF Edit Table Command Prompt Screen shows the ETF screen on which you enter the SAVE command.

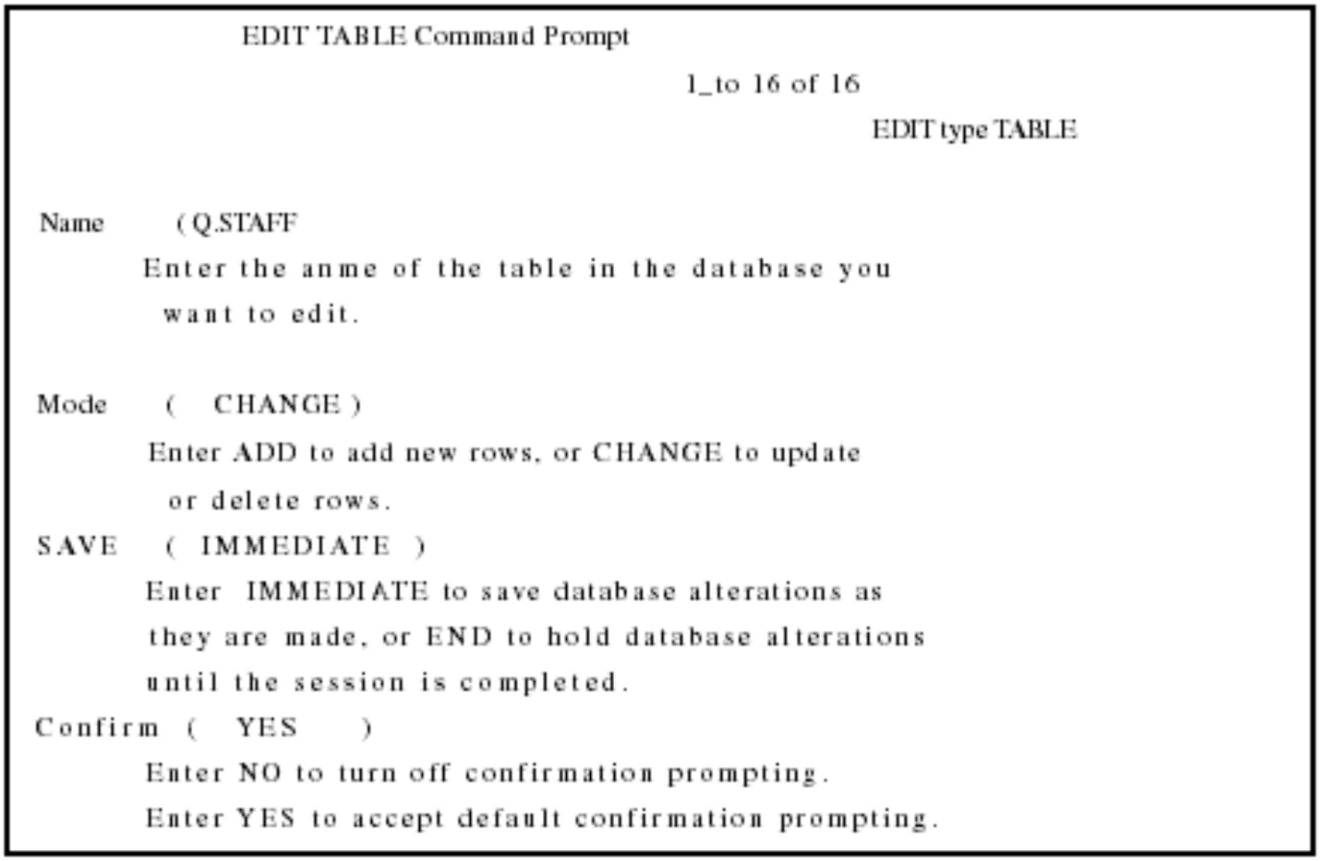
The two options of SAVE are IMMEDIATE and END. QMF issues the message shown at the bottom of the screen when you attempt to use SAVE=IMMEDIATE under TS/API.

In the CICS environment, QMF has additional issues and restrictions, which are:

- ROLLBACK is not supported for Data Definition Language (DDL) statements, since TS/API executes all DDL statements in a one-phase commit (1PC) session as standalone transactions. This means that once a DDL statement is successfully executed, it cannot be rolled back. For example, after you issue a DROP TABLE statement, a QMF prompt panel will request confirmation. Even if you want to undo the DROP TABLE command, you can't because the table will have already been dropped and the statement committed. Be aware of this limitation and use the DROP TABLE/ERASE TABLE statements--and any other DDL statements--with caution.

- QMF Edit Table Facility: There are some problems in TS/API when using the QMF Edit Table Facility. The problems involve:
  - Adding a new row to a table (MODE=ADD)
  - Updating the index values of an existing row (MODE=CHANGE), for which QMF internally INSERTs a row with a new index value

QMF Edit Table Command Prompt Screen



```
                EDIT TABLE Command Prompt
                                              1_to 16 of 16
                                                        EDIT type TABLE

Name      ( Q.STAFF
          Enter the anme of the table in the database you
           want to edit.

Mode      (   CHANGE )
          Enter ADD to add new rows, or CHANGE to update
           or delete rows.
SAVE      (  IMMEDIATE  )
          Enter  IMMEDIATE to save database alterations as
          they are made, or END to hold database alterations
          until the session is completed.
Confirm  (   YES     )
          Enter NO to turn off confirmation prompting.
          Enter YES to accept default confirmation prompting.
```

DB2 Trace Facility

For z/OS TSO DB2 applications, the DB2 Trace Facility is used to collect information about the behavior of QMF when it is used with DB2 itself. After collecting trace information and determining the behavior of QMF when it is used with DB2, you can compare to it the behavior of QMF when it is used with TS/ API.

Use the DB2 Trace Facility in the following cases:

- You may encounter a problem with QMF used with TS/API.

Turning the DB2 Trace Facility On and Off

Set a flag in the TS/API standalone module (stub), BBIACAB, to turn the DB2 Trace Facility on and off.

When the DB2 Trace Facility is active, any DB2 calls which are directed to a DB2 subsystem are traced and written to file *TRACE*.

See Turning the DB2 Trace Facility On and Off on page 22 for details.

These problems occur only when the QMF user attempts to insert a row with a duplicate index value into a table. Use the QMF SQL Query facility instead of the Edit Table Facility for INSERTing new rows into tables.

Currently the QMF Erase Table command does not work from the QMF tables or QMF objects lists. Use the Drop Table SQL statement or QMF Erase Table command from the QMF command line instead.

CHAPTER 2

TS/API Installation and Customization

Overview

This chapter contains the following information:

- Control Flow Under TSO
- Setting Up Teradata Database for TS/API
- Installing QMF

Control Flow Under TSO

This section describes how TS/API interacts with application tools under TSO and in batch.

Vectoring Enabled: Switching Between Teradata Database and DB2

If you have DB2 installed, TS/API provides a mechanism to switch between Teradata Database and DB2 based on the subsystem ID that the application uses when it connects to DB2. The following steps enable this facility:

1. Use members RECUSRMD and APPUSRMD in <*dbcPfx*>.PROCLIB
2. Modify the JCL to meet your installation requirements, including the JOB statement information and high-level qualifiers.
3. Submit the jobs.

If you have DB2 installed, TS/API provides a mechanism to switch between Teradata Database and DB2 based on the subsystem ID that the application uses when it connects to DBs. The following steps enable this facility:

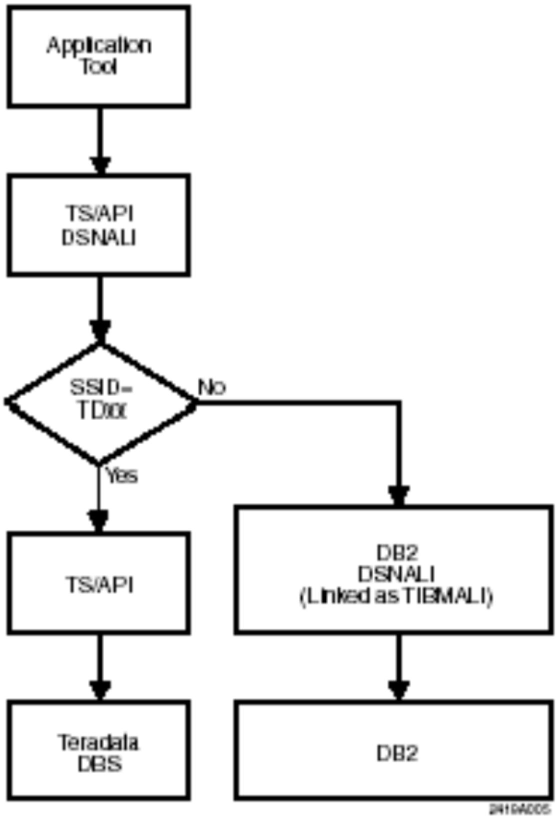
1. Use members RECUSRMD and APPUSRMD in <*dbcPfx*>.PROCLIB
2. Modify the JCl to meet your installation requirements, including the JOB statement information and high-level qualifiers.
3. Submit the jobs.

When vectoring is enabled, TS/API directs all SQL requests to either Teradata Database or DB2, depending upon the subsystem ID that the application supplies at runtime

If The Subsystem ID...	Then...
begins with TD	the request is routed to Teradata Database.
does not begin with TD	the request is routed to DB2.

The following diagram shows the control flow when vectoring is enabled.

Vectoring Enabled



Vectoring Disabled: Teradata Database Access Only

If you do not have DB2 installed, or if you don’t want the vectoring to take place, you may install TS/API with vectoring disabled. For that purpose, Teradata provides its own TIBMALI module, which should be located in the TS/API load library.

When vectoring is disabled and an application specifies a subsystem ID that doesn’t start with TD characters, TS/API’s copy of TIBMALI is invoked and it returns to the application with an error code indicating that DB2 is not active.

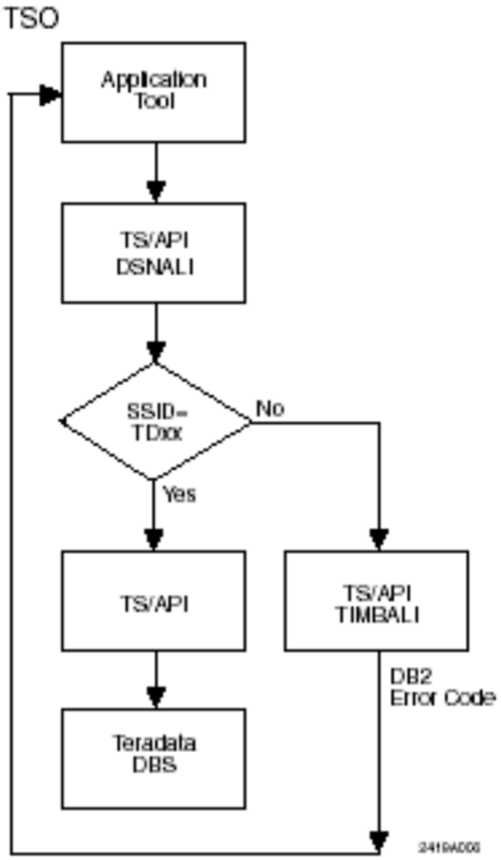
The Vectoring Disabled: Teradata Database Access OnlyVectoring Disabled graphic shows the control flow when vectoring is disabled.

During the original installation of TS/API, vectoring is disabled since TS/API’s copy of TIBMALI is in the load library.

If you customized TS/API to enable vectoring and subsequently decide to disable it, do the following:

1. zrxecute member RSTUSRMD in <*dbcPfx*>.PROCLIB.
2. See the comments in member TDUA001 in <*dbcPfx*>.SAMPLIB.

Vectoring Disabled



Accessing TS/API Load Modules Under TSO and in Batch

Modules from the APILOAD library must be accessible when a program using TS/API is executed. There are several possible ways to make APILOAD modules accessible:

1. Include APILOAD in the system link library list.
2. Include APILOAD modules in the link pack area (LPA).
3. Concatenate APILOAD to JOBLIB or STEPLIB.
4. Concatenate APILOAD to a task library (such as ISPLLIB).

TS/API load libraries cannot be allocated using the LIBDEF command.

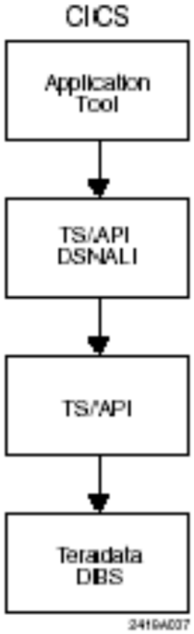
For more flexibility, concatenate APILOAD to your JOBLIB/STEPLIB list for batch applications and to a task library list, such as ISPLLIB, for TSO applications (using application starting CLIST). Also, APPLOAD should be accessible during runtime.

See The DBCLOGON File/Table on page 35 for instructions on allocating the DBCLOGON file.

Control Flow Under CICS

In the CICS environment, QMF uses DB2's CICS Attach Facility module (DSNCLI) to access DB2. TS/API replaces that module with its own DSNCLI, which routes all requests to TS/API. The following graphic, Flow Control in CICS, depicts the control flow.

Flow Control in CICS



TS/API 3.1 does not support vectoring to DB2 under CICS.

Accessing the TS/API Load Modules Under CICS

CICS must be able to access APILOAD and the CLIv2 CICS load library (CXILOAD) in order to run transactions using TS/API. Therefore, concatenate these libraries to the DFHRPL list. The APILOAD library should be placed ahead of the DB2 load library (if any).

For details on how to customize your CICS, see *Teradata Tools and Utilities Installation Guide for IBM z/OS*.

Further Instructions

For instructions on allocating the DBCLOGON file, see The DBCLOGON File/Table on page 35.

After completing the TS/API installation or for instructions on how to set up Teradata Database to enable the TS/API functions, see Setting Up Teradata Database for TS/APISetting Up Teradata Database for TS/API on page 28.

Setting Up Teradata Database for TS/API

This section describes the creation and initialization of a Teradata database, tables, and views used by TS/API. Familiarity with BTEQ is required. For detailed information on BTEQ, see the *Basic Teradata Query Reference Manual*.

Teradata Database Disk Space Requirements

TS/API requires 2 MB of permanent database space on the Teradata hardware platform for storage of special tables, views, and macros.

Additional permanent database space may be required in order to use QMF.

BTEQ Scripts

The BTEQ scripts are included in <*dbcPfx*>.CLIST for z/OS. Running these scripts requires full Teradata Database authority. Userid DBC or SYSADMIN is recommended because these userids have Teradata Database system authority. The BTEQ scripts are subject to change depending on the release level of TS/ API, and reinstallation may be required with new TS/API releases.

If...	Then...
you have more than one Teradata Database	you must run these scripts on each Teradata Database that is used with TS/API.

TS/API Databases

If you are installing TS/API for the first time, you must run BTEQ with the following scripts in the order shown. Run these scripts under the Teradata Database userid DBC or SYSADMIN:

- SYSDBS—Creates TS/API databases SYSAPI, SYSIBM, SYSTEM, SQLDBA.
  If the SYSADMIN database does not have sufficient space, change the line in the SYSDBS BTEQ script that modifies the SYSADMIN PERMSPACE value. Enter the correct amount of PERMSPACE and remove the comment indicator before running the script.
- SYSAPI—Creates TS/API system tables.
- SYSIBM—Creates DB2 system catalog views and tables.

If you have TS/API already installed and are migrating to a new release of TS/API, run the following scripts:
SYSIBM—Replaces DB2 catalog views.

NOTE: SYSIBM does not have DROP TABLE statements. Therefore, these scripts will result in non-0 return codes because the CREATE TABLE will fail (the tables already exist).

The SYSAPI Database

TS/API uses the SYSAPI database in z/OS TSO, and z/OS CICS; it is identical in all environments.

The SYSAPI database contains one table for null mapping used by TS/API. The SYSAPI DBCSQL BTEQ script on the distribution disk creates this table in this database.

The SYSIBM Database

The SYSIBM database contains views and tables TS/API uses to emulate the DB2 system catalog tables. Run the SYSIBM DBCSQL BTEQ script to create the required views and tables.

Installing QMF

This section describes the installation of QMF on z/OS batch, z/OS TSO, and z/OS CICS.

Certified Support - 14.00

TS/API 14.00 provides certified support for:

Under z/OS TSO and in batch

- QMF 8.1 and QMF 9.1

Under z/OS CICS

- QMF 8.1 and QMF 9.1

Usage Notes

QMF does not need to be linked-edited with DSNALI or DSNCLI, and Teradata provides all necessary BTEQ scripts on the installation tape, so the only required step for QMF is to set up Teradata Database using these scripts.

Running the BTEQ scripts accomplishes the following:

- Creates Q database on Teradata Database.
- Creates QMF tables, views, and macros in the Q database.
- Inserts the QMF plan name into the SYSIBM.SYSPLAN table, and INSERTs appropriate cursor names into SYSIBM.SYSCURS.
- Grants the appropriate authority to the above objects.

Run BTEQ Scripts

You have to run BTEQ scripts only once to enable QMF operation on all supported environments (z/OS TSO, batch, and z/OS CICS)

. Run BTEQ scripts to setup TS/API, first. Run the QMF BTEQ scripts after installing the SYSIBM and/or SYSTEM BTEQ scripts.

Before running the BTEQ scripts, you must provide the installation-specific information in JCL.

On z/OS, the scripts are found in the <*dbcPfx*>.CLIST control library, and the BTEQ sample job is in the <*dbcPfx*>.SAMPLIB.

IF...	THEN...
you are installing QMF for TS/API	you must run the following scripts in the given order: QOBJNEW—Defines Q database and QMF system tables. Q—Replaces QMF system views and macros.
you intend to use QMF with Kanji support	you must run: QKANJI—Updates QMF system tables for Kanji support

Storing QMF Objects

The following Teradata Database tables store QMF objects:

- Q.COMMAND_SYNONYMS
- Q.DSQ_RESERVED
- Q.ERROR_LOG
- Q.OBJECT_DATA
- Q.OBJECT_DIRECTORY
- Q.OBJECT_REMARKS
- Q.PROFILES
- Q.RESOURCE_TABLE

The QOBJNEW and Q BTEQ scripts then rebuild the QMF object tables, destroying any data contained in them.

Backing Up QMS Object Tables

If previous releases of TS/API have been installed in your system and if QMF was previously used on your Teradata Database, use the BTEQ *EXPORT* command to back up the QMS object tables to host files before executing the QOBJNEW and Q BTEQ scripts.

Note: If you do not back up these tables, you may lose all QMF stored objects.

Importing the QMS Object Tables

After successfully executing the QOBJNEW and Q BTEQ scripts, use the BTEQ *IMPORT* command to import the QMS object table backup host files to the newly created QMS object tables.

See Basic Teradata Query Reference for information about using the BTEQ EXPORT and IMPORT commands. Also, see the discussion of the QMF export and import functions in the QMF Release Dependencies subsection later in this chapter.

Note: With TS/API 14.00, the COMMAND_SYNONYMS table is actually represented by the following tables to support different environments:

- Q.COMM_SYNS_TSO_E—English command synonyms in MVS/TSO.
- Q.COMM_SYNS_CMS_E—English command synonyms in VM/CMS.
- Q.COMM_SYNS_CMS_K—Kanji command synonyms in VM/CMS.
- Q.COMM_SYNS_CICS_E—English command synonyms in MVS/CICS.
- Q.COMM_SYNS_NULL_E—English command synonyms with QMF 2.4.
- Q.COMM_SYNS_TSO_K—Kanji command synonyms in MVS/TSO.

- Q.COMM_SYNS_CIC_K—Kanji command synonyms in MVS/CICS.

QMF Command Synonyms

TS/API is delivered with default QMF command synonym definitions. If you have added QMF command synonym definitions beyond the delivered default definitions to your current Q.COMMAND_SYNONYMS table, it is your responsibility to make sure they get copied to the new QMF command synonyms table for your QMF environment.

Sample Tables

- Q.APPLICANT
- Q.INTERVIEW
- Q.ORG
- Q.PARTS
- Q.PRODUCTS
- Q.PROJECT
- Q.SALES
- Q.STAFF
- Q.SUPPLIER

Sample Kanji Tables

- Q.APPLICANTK
- Q.INTERVIEWK
- Q.ORGK
- Q.PARTSK
- Q.PRODUCTSK
- Q.PROJECTK
- Q.SALESK
- Q.STAFFK
- Q.SUPPLIERK

QMF Supporting BTEQ Scripts

As a result of running QMF supporting BTEQ scripts, the following tables and views should appear in the Q database:

System Tables

- Q.COMM_SYNS_CICS_E
- Q.COMM_SYNS_CMS_E
- Q.COMM_SYNS_TSO_E
- Q.DSQ_RESERVED

- Q.DSQEC_ALIASES
- Q.DSQEC_COLS_LDB2
- Q.DSQEC_COLS_RDB2
- Q.DSQEC_COLS_SQL
- Q.DSQEC_QMFOBJS
- Q.DSQEC_TABS_LDB2
- Q.DSQEC_TABS_RDB2
- Q.DSQEC_TABS_SQL
- Q.DSQIOLST_AI_VIEW
- Q.DSQIOLST_AU_VIEW
- Q.DSQIOLST_QT_VIEW
- Q.DSQIOLST_TB_VIEW
- Q.COMM_SYNS_CICS_E
- Q.COMM_SYNS_CMS_E
- Q.COMM_SYNS_TSO_E
- Q.DSQ_RESERVED
- Q.DSQEC_ALIASES
- Q.DSQEC_COLS_LDB2

Sample Tables

- Q.APPLICANT
- Q.INTERVIEW
- Q.ORG
- Q.PARTS
- Q.PRODUCTS
- Q.PROJECTS
- Q.SALES
- Q.STAFF
- Q.SUPPLIER

QKANJI Tables / Views

If QKANJI is run, the following tables/views should also appear:

System Tables

- Q.COMM_SYNS_CICS_K
- Q.COMM_SYNS_CMS_K
- Q.COMM_SYNS_TSO_K

Sample Tables

- Q.APPLICANTK
- Q.INTERVIEWK
- Q.ORGK
- Q.PARTSK
- Q.PRODUCTSK
- Q.PROJECTSK
- Q.SALESK
- Q.STAFFK
- Q.SUPPLIER

These views and tables are compatible with those supplied on the IBM QMF installation tape. For a description of these views and tables, see IBM's documentation on QMF.

Teradata Database Macros Created

In addition, required Teradata Database macros are created. On z/OS platform, QMF plan names are registered in the SYSIBM.SYSPLAN table, and required cursors names are INSERTed into the SYSIBM.SYSCURS table.

If you plan to use QMF with several Teradata Databases, you must run these BTEQ scripts on each Teradata Database.

QMF Release Dependencies

The internal format used to store QMF objects (OBJECTLEVEL) is not always consistent between QMF releases. Therefore, a QMF stored object created in one QMF release may not be immediately accessible using a different QMF release.

To prevent QMF stored object access problems from occurring following QMF release migrations, perform the following:

- QMF EXPORT each stored object using the release in which the object was created.
- QMF IMPORT each stored object using the QMF migration release.

Executing these steps converts the internal format of the stored objects to a format compatible with the QMF migration release level.

QMF's failure to access stored objects created under a different release is not a TS/API problem. Contact your IBM representative for more information concerning QMF.

CHAPTER 3

Session Management

Overview

This chapter contains the following information:

- Specifying Teradata Database
- Providing Logon Information

Each subject is covered for z/OS batch, z/OS TSO, and z/OS /CICS.

Specifying Teradata Database

Teradata Database connects to a client through the TDP, which is identified by its tdpid. If more than one Teradata Database is connected to a client, there may be several tdpids you can use.

An MVS *tdpid* can take the form:

TD*xy*

where *x* is one of the following letters:

- P
- Q
- R
- S

and *y* is any single digit or uppercase letter.

A VM tdpid takes the form:

TD*xy*

where *x* and *y* are any single digits or uppercase letters.

In all the supported environments, you can specify the desired tdpid in the DBCLOGON file/ table, as part of your LOGON string (see later in this chapter). In addition, under z/OS TSO and in batch, you may omit the tdpid portion of the LOGON string and specify the tdpid as described below:

Under z/OS TSO and Batch

When a program is invoked that uses DB2, a parameter specifies the DB2 subsystem name (SSN) to be accessed. If the SSN starts with TD, TS/API's Call Attach Facility (DSNALI) directs requests to TS/API. See Teradata Tools and Utilities Installation Guide for IBM z/OS for more information on vectoring. TS/API then uses that name as a tdpid to be accessed, unless it is overridden by a tdpid provided in LOGON string.

Providing Logon Information

In order to log on to Teradata Database, two other values are required besides the tdpid:

- The Teradata Database userid
- The password associated with the Teradata Database userid

Logon parameters can be specified in one of two ways:

- Through a DBCLOGON file/table
- Through a TDP logon exit

The DBCLOGON File / Table

Under z/OS TSO and batch, TS/API uses a DBCLOGON file to obtain logon information. The DBCLOGON file should be allocated to a sequential data set with the following characteristics:

- A record format of F or FB
- A record length of 80
- A block size that is any multiple of 80

The DBCLOGON file may contain any number of TS/API directives, including the LOGON command. (See TS/API Directives on page 72 for a list of TS/API directives.) The LOGON command provides the Teradata Database userid, password, and, optionally, the tdpid. The LOGON command in the DBCLOGON file is formatted like the LOGON command in BTEQ (without the leading dot):

```
LOGON  [tdpid/]userid,password;
```

The semicolon at the end of the LOGON command is optional. If the tdpid is specified, it overrides the value of the tdpid passed to TS/API by other means.

To allocate the DBCLOGON file:

Under TSO, use a DDNAME of DBCLOGON as follows:

```
ALLOCATE DD(DBCLOGON) DA(’ logon.dataset.name’) SHR
```

- In batch, use a ddname of DBCLOGON as follows:

```
//DBCLOGON DD DSN=logon.dataset.name,DISP=SHR
```

Note: Teradata recommends using userid.DBCLOGON as the logon.dataset.name in z/OS.

DBCLOGON must be allocated before you use any program that invokes TS/API. If it is not allocated or if it does not contain a LOGON command, TS/API performs an implicit logon to Teradata Database. The TDP passes an implicit logon request to the logon exit if one is installed (see details later in this chapter).

Under CICS

TS/API uses the following tables in CICS to obtain logon information for a transaction:

- BBIRCT (Resource Control Table in CICS)

  Each entry in BBIRCT contains one or more transaction ids and the corresponding plan name. It also specifies what name associated with a CICS transaction TS/API should use to look up the DBCLOGON table in CICS for that transaction’s logon information. Each transaction that uses TS/API must be defined in the BBIRCT table in CICS.

- DBCLOGON

  Each entry in the DBCLOGON table in CICS contains a key field, which is used during the search, and an information field that contains TS/API directives, including a LOGON command.

Using Macros for BBIRCT and DBCLOGON

TS/API provides macros for creating and updating the BBIRCT and DBCLOGON tables in CICS. These macros support the use of the table generation procedure (DFHAUPLK or DFHAUPLE). The following are the instructions on use of these two macros:

BBIRCT Macro

Use the following statements to build the BBIRCT table in CICS:

```
BBIRCT TYPE=INITIAL
BBIRCT TYPE=ENTRY,
     TXID=(<txid1>,<txid2>,<txid3>, ...),
     PLAN=<plan_name>,
     AUTH=(<auth_type_list>)
BBIRCT TYPE=ENTRY
     TXID=(<txid1>, <txid2>, <txid3>, ...),
     PLAN=<plan_name>,
     AUTH=(<auth_type_lists)

...
BBIRCT TYPE=FINAL
```

The TYPE=INITIAL macro must be specified first. The TYPE=ENTRY macro can be specified as many times as needed to identify specific transactions. The TYPE=FINAL macro is specified last, causing the BBIRCT table in CICS to be generated.

The <auth_type_list> is a list of up to three keywords: USERID, TERMID, and TXID, separated by commas (for example, (USERID,TXID), (TXID,USERID,TERMID), and so forth).

The AUTH parameter controls how TS/API uses the names associated with a CICS transaction to obtain the authorization key, which is used to search the DBCLOGON table in CICS for logon information. The key may consist of one of the following:

- The CICS sign-on ID (specified by the USERID keyword)
- The terminal ID (specified by TERMID)
- The transaction ID (specified by TXID)

The order in which these keywords are specified in the list dictates the order in which TS/API obtains the information from CICS. For example, AUTH=(USERID,TERMID) means that TS/API first checks for the CICS sign-on ID, and only if it is blank, TS/API uses the terminal ID. AUTH=(TERMID,USERID) causes TS/API to obtain the key in the reverse order.

When a key is obtained from CICS, it is used to search the DBCLOGON table for the logon string. If the authorization key is all blanks, or if there is no matching key field in the DBCLOGON table, TS/API issues an implicit logon request with a zero-length logon string, relying therefore on the TDP logon exit.

DBCLOGON Macro

Use the following statements to build the DBCLOGON table in CICS:

```
DBCLOGON TYPE=INITIAL
```

```
    DBCLOGON TYPE=ENTRY,
         AUTHKEY=<auth_key>,
         LSTRING='<DBC_logon_string>'
    DBCLOGON TYPE=ENTRY,
         AUTHKEY=<auth_key>,
         LSTRING='<DBC_logon_string>'
         ...
         DBCLOGON TYPE=FINAL
```

The <DBC_logon_string> must be surrounded by single quotes. TS/API directives must be separated from each other by semicolons.

The TYPE=INITIAL macro must be specified first. The TYPE=ENTRY macro can be specified as many times as needed to identify specific authorization keys. The TYPE=FINAL macro is specified last, causing the DBCLOGON table in CICS to be generated.

The <*auth_key*> is a name of up to 8 characters, which specifies the key of the record. <*DBC_logon_string*> is a string of up to 80 characters, which typically follows a format of:

```
logon tdpid/userid,password;<other TS/API directives>
```

The DBCLOGON and BBIRCT tables must both be linked-edited into a library accessible to CICS for the CICS LOAD command (concatenated to the DFHRPL library list). The tables also must be defined to CICS in the PPT table. See Teradata Tools and Utilities Installation Guide for IBM z/OS for details.

However, you may leave the DBCLOGON table in CICS empty and use a TDP exit in conjunction with TS/API to provide CICS security logic. If the DBCLOGON table in CICS is empty or does not contain a record with a matching key field, then at logon time, TS/API issues an implicit logon request with a zeroHlength logon string.

The JCL for building the BBIRCT and DBCLOGON tables is provided in the <dbcPfx>.SAMPLIB file. BBIRCT and DBCLOGON macros are provided in <dbcPfx>.TDPMAC.

The Logon Exit

An alternate way to provide logon information is via the Teradata Database logon exit. Teradata Database provides two kinds of logon exits:

- TDPLGUX
- TDPUAX

Both exits provide a way of executing implicit logons using pre-assigned userids and passwords. Both exits have levels of security that can prevent unauthorized use of Teradata Database .

IF the logon exit...	THEN...
authorizes access to Teradata Database	TS/API is connected under the userid supplied by the logon exit.
is not installed	the attempt to perform an implicit logon fails.

When a logon exit is used, no password is necessary to authorize access to Teradata Database. The logon exit can directTeradata Database to accept the implicit logon without an associated password. The logon exit is secure, so unauthorized access to Teradata Database cannot occur.

The advantage of using a logon exit is the security of not having userid and password information stored in a logon file. The disadvantages are that you lose the flexibility to control the logon userid and the systems programming staff must support the installation, modification, and maintenance of the logon exit.

CHAPTER 4

Product Management

Overview

This chapter describes how to use QMF with TS/API in different environments and contains the following information:

- z/OS TSO and Batch Products
- z/OS CICS Products

TS/APIRelease 14.00 is certified to support the following products:

- Query Management Facility (QMF) Versions 8.1 and 9.1

z/OS TSO and Batch Products

TS/API supports QMF under z/OS TSO and in the batch environment.

To ensure connection to Teradata Database instead of to DB2, verify the following:

1 TS/API libraries are properly accessible.

The TS/API load library (APILOAD) must be accessible to z/OS through its normal library search order (task library, STEPLIB, JOBLIB, link library) ahead of the DB2 load library (if any). In order to provide a proper concatenation of the TS/API library, you may need to modify CLIST or JCL used to invoke a product. The CLIv2 load library (APPLOAD) also must be accessible to z/OS.

2 Your DBCLOGON file, or TDP logon exit, is properly prepared (see The Logon Exit on page 37 for details).

A proper *tdpid* in form of TD*xx* is specified as a subsystem name when invoking the product. If you specify a tdpid in your DBCLOGON file, it overrides the value specified by other means.

However, in the z/OS TSO and batch environments, the subsystem name is used by the TS/API Call Attach Facility to route the requests either to DB2 or to TS/API. (See Control Flow Under TSO on page 24 for more information about vectoring.) Therefore, even if you have a *tdpid* in the LOGON string, the subsystem name still must start with TD to ensure vectoring the requests to TS/API rather than to DB2.

The following are examples of JCL statements used to invoke QMF with a *tdpid* equal to *TDP0*:

```
//SYSTSIN   DD  *
PROFILE PREFIX(<userid>)
ISPSTART PGM(DSQQMFE) NEWAPPL PARM(M=B,P=QMF310,S=TDP0)
/*
```

z/OS CICS Products

TS/API supports QMF z/OS CICS.

To ensure connection to Teradata Database instead of to DB2, verify the following:

1. The TS/API load library (APILOAD) is properly accessed.

   It must be concatenated in the DFHRPL list ahead of the DB2 library (if any). You will need to modify the CICS startup procedure in order to provide a proper concatenation of the TS/API library. The CICS CLIv2 load library (CXILOAD) also must be accessible to CICS.

2. The BBIRCT and DBCLOGON tables in CICS or the logon exit are properly prepared.
   For additional information, see Using Macros for BBIRCT and DBCLOGON on page 36.

CHAPTER 5

Problem Management

Overview

This chapter describes the problem management features provided by TS/API.

This chapter contains the following information:

TS/API Debug Facility

TS/API includes an error resolution feature controlled by the TS/API directive DEBUG. DEBUG may be issued from any of the following:

- The DBCLOGON file
- The application’s command line
- The application’s batch input file

The DEBUG directive controls the level of debugging output generated by TS/API during a session.

The following table describes the four possible settings.

Debugging Options

Debug Option	Description
DEBUG OFF	The default. No trace output is generated. You can also use DEBUG OFF to stop tracing when DEBUG ON or DEBUG SQL is in effect.
DEBUG SQL	All SQL statements sent to Teradata Database are printed. This command lets you view how TS/API converts DB2 SQL into Teradata SQL.
DEBUG ON	TS/API routine names are printed on entry and exit, along with the contents of all relevant data structures. DEBUG ON generates a large volume of information and is used mainly to trace serious system problems. Usually, you would send this information to the Global Support Center (GSC) for problem resolution.
DEBUG PERF	Performance monitor information is printed (time spent in CLIv2, TDP and Teradata Database). See DEBUG PERF for details.

DEBUG Directive

You can issue the DEBUG directive as a standard SQL request (for instance, from the QUERY PANEL of QMF). This method is particularly useful when you want to see how TS/API translates your SQL before sending it to Teradata Database.

You can also place the DEBUG directive in the DBCLOGON file/table on a separate line. That way, you can trace TS/API activity even before a connection is made with Teradata Database. This is useful in diagnosing errors that occur in trying to start QMF.

When TS/API receives the DEBUG directive, its internal tracing mechanism is activated. If the directive was sent from QMF, TS/API returns an SQLCODE of zero, even though no actual Teradata Database request was performed.

DB2 Trace Facility

TS/API also provides a DB2 trace facility under both z/OS TSO and batch, if you have DB2 installed and vectoring enabled (see DB2 Trace Facility). The DB2 trace facility pinpoints problems with a DB2 application used with TS/API.

The trace provides DB2 session information when QMF is used with DB2. After determining how QMF behaves when used with DB2, a user can compare that to how the QMF behaves when used with TS/API. Determining the differences in behavior, and where these differences occur, is useful in debugging problems that a QMF have when used with TS/API.

Note that DB2 call information is traced both for QMF passing information to DB2 to make a request and for DB2 passing requested information and call status back to QMF.

Turning the DB2 Trace On and Off

The DB2 trace can be turned on and off by setting a flag in the TS/API standalone program BBIACAB. TS/API is delivered with DB2 trace off.

IF you want to...	Then...
turn the DB2 trace on	use members RECUSRMD and APPUSRMD in <*dbcPfx*>.PROCLIB This reassembles and link-edits program BBIACAB with the DB2 Trace flag set to a value greater than zero, indicating that the DB2 trace is active.
turn the DB2 trace off	use member RSTUSRMD in <*dbcPfx*>.PROCLIB. (See comments in member TDUA002 or <*dbcPfx*>.SAMPLIB.) This restores the program to its original state.

Note: The SMP/E RESTORE must be performed before applying any subsequent PTFs to the BBIACAB program.

The OUTPUT and TRACE Files in z/OS

In z/OS, debug output is written to the DDNAME OUTPUT, and DB2 trace output is written to the DDNAME TRACE.

IF OUTPUT...	AND...	THEN...
is explicitly allocated		that allocation is used
is not explicitly allocated	a data set named *userid*.TSAPI.DEBUG exists	that data set is used.

Otherwise, OUTPUT is dynamically allocated to a spool file.

The TRACE file must be explicitly allocated if DB2 trace is turned on. TS/API doesn't do any implicit allocations for the TRACE file.

The OUTPUT and TRACE files should be sequential data sets with a record format of VBA, a record length of 85, and a block size of 6124.

Allocating OUTPUT

When allocating the OUTPUT file, a DDNAME of OUTPUT must be used. For example, in TSO:

```
ALLOCATE DD(OUTPUT) SYSOUT(A) RECFM(VBA) LRECL(85) BLKSIZE(6124)
```

or in batch:

```
//OUTPUT DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=85,BLKSIZE=6124)
```

Allocating TRACE

When allocating the TRACE file, a DDNAME of TRACE must be used. For example, in TSO:

```
ALLOCATE DD(TRACE) SYSOUT(A) RECFM(VBA) LRECL(85) BLKSIZE(6124)
```

or in batch:

```
//TRACE DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=85,BLKSIZE=6124)
```

Note: Use a file disposition of MOD when allocating the OUTPUT and TRACE data sets. If you don't use MOD, only the last cycle of TS/API debug or DB2 trace output is saved in the file.

The TSDB Debug Destination in z/OS CICS

Under CICS, TS/API writes debug information to 'destination name' TSDB. All TS/API CICS users in a given CICS region with the debug turned on will write to the same destination. Therefore, that destination must be defined to CICS as an extrapartition transient data queue. See The TSDB Debug Destination in z/OS CICS for details.

That guarantees that each record in the debug output is prefixed with the CICS terminal ID and the CICS transaction ID, which allows the debug output to be identified with the transaction that produced it. This identification is very important if more than one CICS user is creating debug output at the same time.

OUTPUT Data Set

The data set associated with the TSDB data queue may have any name, but Teradata recommends the name OUTPUT for compatibility with debug output in z/OS TSO.

The data queue can be opened automatically at CICS startup or explicitly via the CEMT SET QUEUE command. The initial status is controlled by the OPEN=INITIAL/DEFERRED parameter on the DFHDCT TYPE=EXTRA macro for DESTID=TSDB.

- OPEN=INITIAL, the default, specifies that the data set be opened by system initialization.
- OPEN=DEFERRED specifies that the data set remain closed until explicitly opened via the master terminal open/close function (see below) or via a DFHOC macro from an application program.

Master Terminal Commands

The master terminal commands to manage the data set are:

```
CEMT SET QUEUE(TSDB) OPEN/CLOSE
```

Closing the debug file will force any blocks/records buffered in memory to be written to the data set.

An external job can then be run to copy or print the contents of the OUTPUT data set. The OUTPUT data set can be viewed via ISPF browse while it is open.

The TSDB data queue must be open while TS/API is writing data to it. If the queue is closed, TS/API will send the following message to the terminal that issued the request:

```
fopen of the debugging file to DDNAME OUTPUT failed
```

The physical sequential file itself must be allocated in the CICS startup JCL. The following statement can be used in the CICS startup JCL:

```
//OUTPUT DD DSN=TSAPI.DEBUG.OUTPUT,DISP=MOD
```

Notice:

> **Use a file disposition of MOD. Otherwise, only the last cycle of TS/API debug data will be captured. With MOD, each time data is written to the file, the file is appended. Consequently, you must initialize the file when it is necessary to clear out old data.**

Basic Problem Determination (DEBUG SQL)

Interactive Application Debugging

The TS/API debugging file can be used for interactive and batch application debugging. Additional debugging features are available interactively in each application.

QMF provides:

- Log
- Display of problem

Pressing the PF1 or HELP key causes SQLCA display. For additional information, see Advanced Problem Determination (DEBUG ON.

OUTPUT File Contents

A TS/API cycle begins at TS/API runtime invocation and ends at TS/API runtime exit. The cycle is equivalent to an SQL request to Teradata Database via TS/API.

First Cycle

TS/API's first cycle is numbered 0. The presence of the first cycle indicates that the TS/API front-end assembler routines are successfully communicating with the TS/API C runtime routines.

Other displays in the first cycle indicate the TS/API release level and the event date and time.

If nothing is displayed, one of three possibilities exists:

- The OUTPUT file allocation is incorrect (or dummied out).

  In this case, the application is successfully communicating with the TS/API runtime routines (via the TS/API front-end), but you would have to prove it by submitting queries (because the output file is disabled) before exiting the application and correcting it. For example, issuing the DATABASE <userid> command would execute successfully on Teradata Database but not in DB2.
- The TS/API library is concatenated behind the DB2 library.

  In this case, the application is communicating with DB2 but thinks it's communicating with Teradata Database. Issuing the DATABASE <userid> command quickly determines the correct environment.
- A problem in the TS/API front-end occurred before invoking the TS/API runtime routines.

  The main function of the TS/API front-end is to establish the correct environment and to invoke the TS/API runtime routines. If an error occurs during setup or invocation, the TS/ API front-end does the following:
  - Updates the SQLCA with an internal DBMS error message
  - Updates the SQLCODE and return/reason code
  - Returns control to the application

This level of error usually requires contacting the Global Support Center (GSC). See Error Reporting Procedures

The Next Few Cycles

Logon information is displayed for a few cycles after the first. This information includes DBCLOGON file allocation messages and a message indicating a successful or unsuccessful logon.

QMF, for example, next issues static SQL requests to set up the application environment for interaction with the DBMS.

The two types of SQL, static and dynamic, are IBM's method of managing SQL provided for application use. Dynamic SQL is input at runtime from the application. This allows user input of native SQL. Static SQL is precompiled/bound as part of the application and is visible at precompile time but not runtime because it is internal to DB2. Static SQL is under application control, not user control.

Teradata Database macros are created at TS/API installation time to support DB2 static SQL. To view this SQL, issue the Teradata Database command SHOW MACRO <macro name>.

The Middle Cycles

The middle cycles contain debugging information on any native SQL entered by the user and commonly sent as is to the TS/API runtime routines. The main information displayed during each of these cycles is the following:

- The call type directing the call
- Any SQL before and after translation by the TS/API parser

- Any Teradata Database or CLIv2 error codes and messages
- Any reported return codes and error codes
- An exit from each call
- The SQLCODE returned to the application

Last Cycles

Logoff information indicating a successful or unsuccessful logoff is displayed.

Unit of Work

QMF automatically manages the unit of work for the user. Typically, the application commits automatically after a single successful query. TS/API's transaction management logic automatically emulates that of DB2 by issuing BT and ET requests where appropriate. Some of these requests are performed implicitly by TS/API; others are performed by the application's explicit request.

Typically, QMF auto-rollback immediately after a query that returns a DBMS error.

Advanced Problem Determination (DEBUG ON)

The DEBUG ON command is useful for advanced problem determination. Each debug setting is cumulative. For example, an output file captured with DEBUG ON is a superset of an output file captured with DEBUG SQL.

Note: The DEBUG ON setting creates a large volume of information, so be certain that the OUTPUT file (either with the TS/API default allocation or as overridden by you) can adequately hold the information. Teradata strongly recommends that you use a spool file when possible.

OUTPUT File Contents

The DEBUG ON setting prints detailed information for each call to TS/API:

- Flow of control within TS/API
- Entry and exit for each TS/API routine invoked
- All relevant application structures for each call to TS/API:
  - RDIIN - The driver structure to TS/API, which specifies call type, pointers to all other structures, and other flags related to the call
  - <sql statement> - Native SQL or information to build static SQL
  - Input SQLDA - The input data structure matching input client variables in the <sql statement>
  - Output SQLDA - The output data structure matching output client variables returned to the user
- CLIv2 structures and data:
  - DBCAREA (equivalent to RDIIN) - The driver structure to Teradata Database, which specifies call type, pointers to all other structures, and other flags related to the call
  - <sql statement> - Native SQL or invocation of a macro
  - USING clause and data parcel (equivalent to Input SQLDA) - USING clause maps the Input SQLDA and is added to the <sql statement>; the data parcel contains the Input SQLDA and is sent to Teradata Database with the <sql statement>
  - Record Parcel (equivalent to the Output SQLDA) - The contents of the Record Parcel are moved to the Output SQLDA
  - Success/Failure Parcels, CLIv2 errors, and TS/API customized errors contain information used to build most of SQLCA

- Additional informational displays
  - Information related to key current functions, such as unit of work messages and special-case parser messages

Error Reporting Procedures

If you encounter a TS/API error condition that you cannot solve, report the error to the Global Support Center (GSC). Be sure to include all of the information in Error Reporting Procedures.

Problem Reporting Information

#	Category	Information
1	The operating system	
2	The operating system release	
3	The release of TS/API	
4	The release of DB2 and PTF level	
5	The Teradata Database release	
6	QMF's release level and PUT level	
7	Problem description if not already in call log	
8	The exact commands entered under the product being used with TS/API, plus the accompanying SQL statements. (In some cases, the customer will not know what the SQL looks like, since the product generates the SQL transparently.)	
9	The TS/API DEBUG output (if any) from OUTPUT file with DEBUG ON	
10	DB2 trace output (if any) from TRACE file with DB2 tracing enabled	

To obtain debug information, do the following:

1. Prepare and allocate a DBCLOGON file/table prior to TS/API invocation. See LOGON for more information on DBLOGON.
2. In the DBCLOGON file, include the following:

   DEBUG ON;

3. LOGON [tdpid/]userid,password;
4. Allocate an OUTPUT file to receive the output of the debugging session. See the beginning of this chapter for more information on the OUTPUT file.
5. Send the OUTPUT file contents to the GSC (in machine-readable form, if possible).

Error Translations

TS/API translates Teradata Database and CLIv2 error codes into the appropriate DB2 SQLCODEs and their corresponding SQLSTATEs.

Error Translation Logic Strategy

However, in some cases, TS/API provides no translation because DB2 has a more limited set of error codes than Teradata Database, and more meaningful information is available in messages.

The Teradata Database error codes that TS/API does not translate retain their own error numbers in the range 1000-7999 but are sign-inverted (made negative) to conform to DB2 requirements.

The CLIv2 error codes that TS/API does not translate retain their own error numbers in the range 1-999 but are sign-inverted (made negative) and have negative 8000 added to them so they can’t be confused with DB2 SQLCODEs or Teradata Database error codes. For example, the following error code:

```
CLI0286
```

would be represented as:

```
SQLCODE -8286
```

Because the Teradata Database and CLIv2 error code range is outside of the normal DB2 range of -999 to 999, an untranslated error code usually results in fatal termination of the application or program.

Error Code Translations for DB2 contains translation tables between DB2 SQLCODEs and Teradata SQL error codes.

CHAPTER 6

Static SQL and System Catalog Support

Overview

This chapter describes how TS/API supports static SQL statements embedded in QMF and how TS/API emulates the system catalog tables of DB2.

This chapter contains the following information:

- Static SQL Support
- Preparing QMF
- System Catalog Support

Static SQL Support

This section explains the following:

- Program preparation of DB2 applications
- TS/API support of DB2 embedded static SQL

Preparing QMF

This section shows the process through which QMF passes in order to become executable. The steps are divided into those performed at IBM and those performed at the customer site.

Steps Performed by IBM

1. The source code of a program product, written in a host language, such as ASSEMBLER or C, is fed into the DB2 precompiler at IBM.
2. The DB2 precompiler changes any static SQL statements embedded in the program into comments and replaces them with valid host language statements.
3. The DB2 precompiler validates SQL syntax and checks the definitions of the host language variables.
4. The DB2 precompiler generates a data set called a database request module (DBRM). DBRMs are used as input to the application program binding process. A DBRM contains the following:
    - The embedded SQL statements extracted from the source program
    - The host variable information extracted from the source program
    - Information that ties the DBRM to the source statements

   The precompiler also generates the expanded host language program, which references the DBRM and is used as input to the compiler. Resulting object code is input to the linkage editor, which produces one or more executable load modules.
5. The DBRMs and the load modules are made available to the customer.

Steps Performed by the Customer

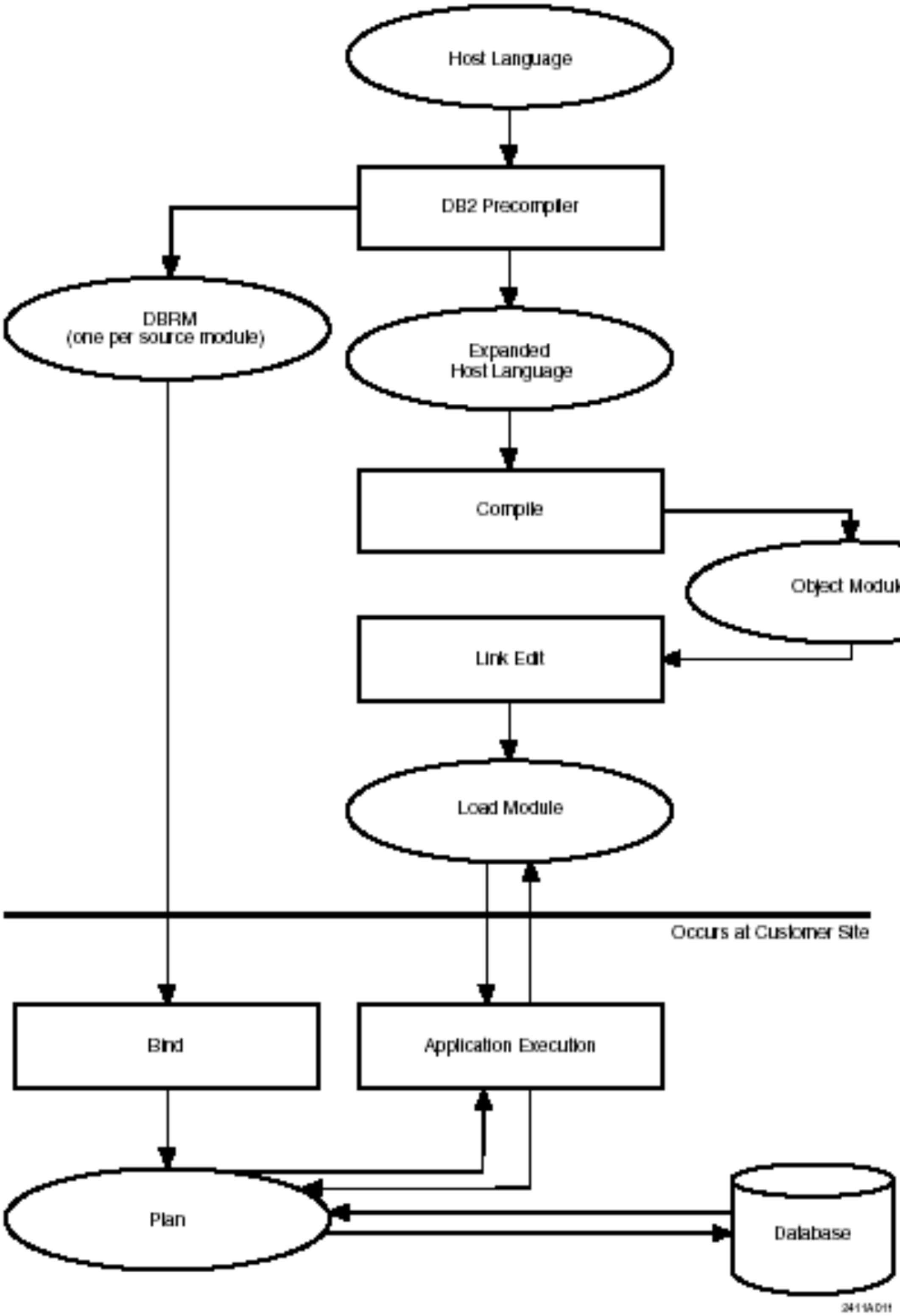
The customer uses the DBRMs as input to the process of binding QMF. Binding produces a control structure, called a plan, which DB2 uses when accessing data during application program execution.

Bind performs the following:

1. Checking syntax
2. Checking operation authorization
3. Determining the optimal plan data access strategy
4. Validating SQL statements, using the DB2 catalog
5. Generating the plan

During the execution, DB2 uses the plan to execute the static SQL statements.

PreparingQMF with Embedded SQL for Execution



TS/API Support of DB2 Static SQL

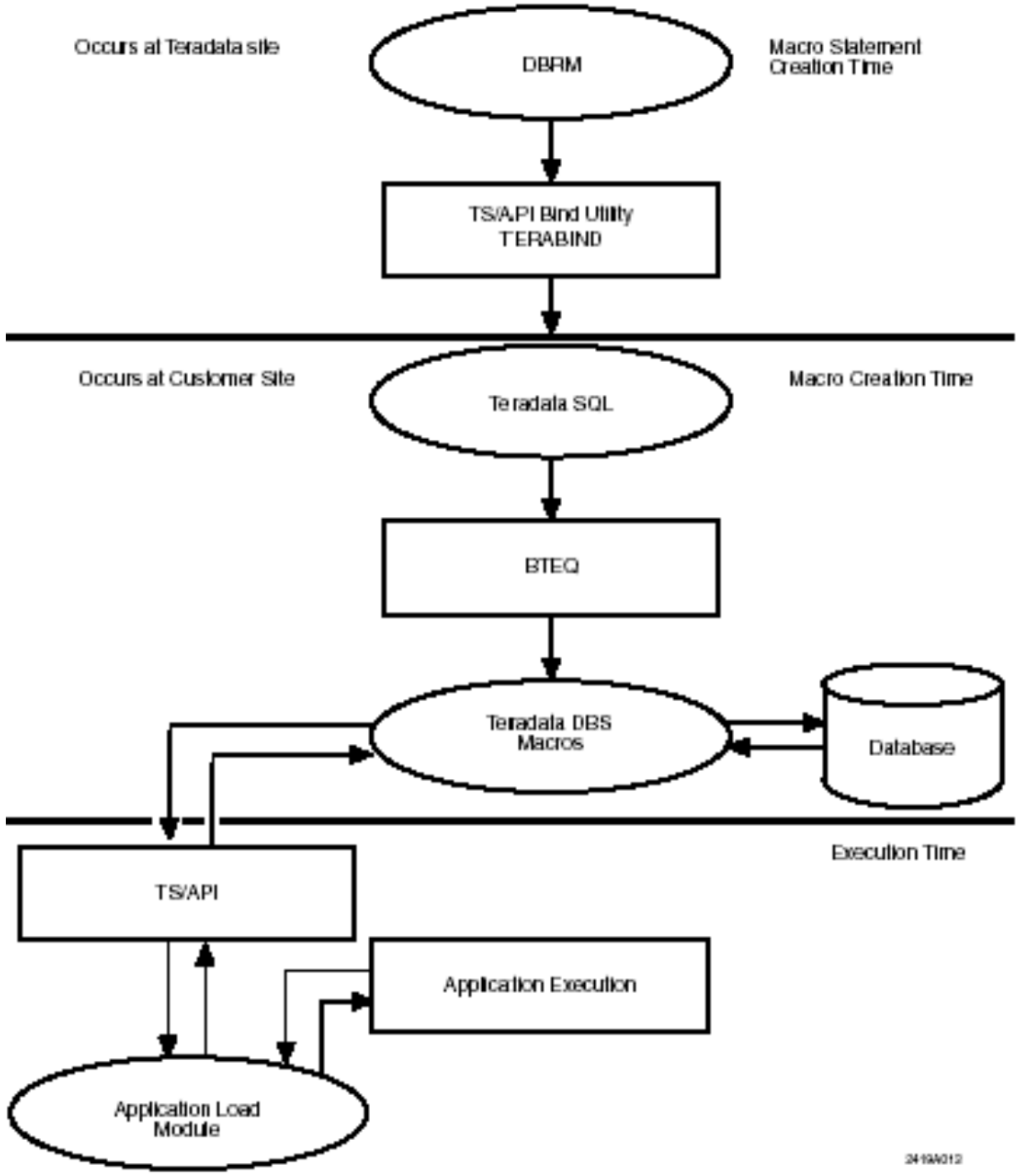
For QMF, Teradata provides the required BTEQ scripts on the installation tape. Use these scripts as an input to BTEQ in order to create the emulation macros on Teradata Database. For details, see TS/API Installation and CustomizationTS/API Installation and Customization.

At execution time, TS/API executes the appropriate Teradata SQL macros, using the macro naming convention as follows:

```
   Database name.Plan name_Program name_Section number
```

Teradata SQL macros are optimized each time they are executed.

DB2 Static SQL Translation to Teradata SQL Macros Process for QMF



System Catalog Support

To provide transparency, TS/API emulates the DB2 system catalog. This section describes the techniques and information sources that TS/API uses for system catalog emulation.

System Catalog Support

TS/API emulates the system catalog tables of DB2 by building equivalent system catalog views/tables in Teradata

Database at installation time.

The DBMS system catalog (qualified by SYSIBM on DB2 and DBC on Teradata Database) contains all information relating to objects and the authorization on those objects for the entire Teradata Database.

System Catalog Views

In most cases, DB2 system catalog tables are emulated using Teradata Database views. During the installation process, most DB2 system catalog tables are created as views of one or more Teradata Database catalog tables, special TS/API tables, and constant or calculated values. These views are referred to as TS/API catalog views.

Several Teradata Database catalog tables, stored in the DBC database, are used to create the TS/API catalog views. The Teradata Database catalog tables that are used include those in the following table.

Table 1:_Teradata Database Catalog Tables

Teradata Database Table	Description
DBC.DBASE	Describes each database and userid on Teradata Database
DBC.TVM	Describes each table, view, or macro onTeradata Database
DBC.TVFIELDS	Describes each column on Teradata Database
DBC.ACCESSRIGHTS	Describes access rights on Teradata Database
DBC.INDEXES	Describes columns contained in indexes in Teradata Database

The SYSAPI database contains one special dummy table that TS/API uses for null mapping, called SYSAPI.SYSDUMMY.

System Catalog Tables

In a few cases, TS/API emulates a DB2 system catalog table by creating a Teradata Database table. When this is necessary, the table is stored in the SYSIBM database. This kind of table is usually required when the Teradata Database catalog tables lack the necessary information, particularly when a DB2 object has no equivalent Teradata Database object.

Long Teradata Database Names

DB2 supports names for databases, tables, columns, authorization ids, and other objects that are either eight (8) or eighteen (18) characters in length. For most of these objects, Teradata Database accepts names up to 30 characters

To accommodate existing Teradata Database tables having longer names, the TS/API catalog views expand most of these columns to the full 30-character width accepted by Teradata Database.

Notice:

> **Due to internal limitations in many applications, use of Teradata Database object names longer than those permitted by DB2 may cause anomalies. If QMF fetches a column name into a client variable that is less than 30 characters, a long Teradata Database object name may be improperly truncated. If the truncated name is then used as the basis for building further queries, subsequent syntax errors may occur.**

Authorization IDs

The Creator columns in DB2 catalog tables refer to the authorization id used by DB2. No authorization id for Teradata Database exists; the database or userid under which the table or view was created is used as the closest Teradata Database equivalent to an authorization id.

This equivalent id is not used to support either the DB2 concept of a database or table space.

An authorization id is an eight-character DB2 object and the Teradata Database name or userid is up 31 characters so the limitations mentioned in the preceding caution apply to these columns also.

DB2 SYSIBM System Catalog Tables Emulation

One database, SYSIBM, that emulates the system catalog of DB2, is created during TS/API installation. Descriptions of this database is contained in Appendix B: TS/API Catalog Emulation.

CHAPTER 7

Command Syntax

Overview

This chapter describes the differences between Teradata SQL and the SQL used by DB2.

TS/API Release 14.00 supports the Teradata Database release 13.10, 13.00. 12.0, and 6.2 in Teradata mode only.

This chapter contains the following information:

- SQL Differences
- Updatable Cursor Support
- Syntax Mapping Strategy
- Teradata SQL Extensions Differences
- DB2 Syntax Mapping
- Teradata SQL Extensions
- TS/API Directives

SQL Differences

A number of differences exist between Teradata SQL (the language used to access databases on Teradata Database) and the SQL language used with DB2. In order for TS/API to execute SQL requests from QMF, TS/API transforms the SQL requests into valid Teradata SQL and submits them to the Teradata Database.

This chapter also describes the syntax mappings and other actions that TS/API performs to emulate DB2 behavior as much as possible.

The following subjects are covered:

- Transaction management
- Emulating updatable cursors
- Syntax mapping between DB2 and Teradata Database
- Teradata SQL extensions
- TS/API directives

Transaction Management

DB2

Under DB2, a unit of work is defined as an entity capable of being committed or rolled back. The first unit of work is the work done from the first access to a relational database until a COMMIT or ROLLBACK occurs. The next unit of work is the work done from the next access to the database until the next COMMIT or ROLLBACK occurs, and so forth. All modifications to the database are capable of being rolled back until a COMMIT is successfully executed.

Teradata Database

Within Teradata Database, a unit of work is defined as an entity that falls between a BEGIN TRANSACTION (BT) statement and an END TRANSACTION (ET) statement. If no BT and ET statements are present, Teradata Database automatically treats each statement as a unit of work.

To emulate DB2's, TS/API issues a BT prior to sending any request to Teradata Database if this is the first request in the session or if the previous request was a COMMIT/ROLLBACK. In this way, TS/API begins a transaction, which then is ended by an ET/ROLLBACK statement when the application issues a COMMIT/ ROLLBACK.

Under CICS, however, applications are using EXEC CICS SYNCPOINT commands instead of COMMIT or ROLLBACK, and CICS performs transaction management in two-phase commit (2PC) mode. Therefore, under CICS, TS/API doesn't issue any BT/ET statements; instead, TS/API provides a syncpoint exit routine through which CICS notifies TS/API of any syncpoints performed. TS/API does only its own clean-up in response to these notifications.

Updatable Cursor Support

A cursor is a named control structure used by an application program to point and retrieve a specific row from a set of rows representing the result of executing a SELECT statement. If a DBMS supports updatable cursors and if the result table is not read only, an application may use the cursor to update or delete rows.

DB2 supports updatable cursors, and QMF uses this feature to accomplish its tasks. Teradata Database pre-V2R2 software did not support updatable cursors, so TS/API had to emulate this feature by other means. This emulation applies to V2R2 Teradata Database as well, since the same TS/API logic is used.

This emulation causes some additional restrictions, namely:

- For a cursor to be updatable, the underlying table must have a unique index; in addition, if the SELECT statement for the cursor refers to a view, the unique index must be visible to that view.
- TS/API supports only repeatable read (RR) cursor isolation level. (See Glossary for term definitions.)

Non-Mapped Error Codes

The following table lists the error codes normally issued by DB2, which are returned by TS/API as a result of error conditions relating to the use of updatable cursors.

Non-Mapped Error Codes

Code	Error Description
-504	THE CURSOR NAME <cursor_name> IS NOT DEFINED
-507	THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT OPENED
-508	THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT POSITIONED ON A ROW
-510	THE TABLE DESIGNATED BY THE CURSOR OF THE UPDATE OR DELETE STATEMENT CANNOT BE MODIFIED
-511	THE FOR UPDATE CLAUSE CANNOT BE SPECIFIED BECAUSE THE TABLE DESIGNATED BY THE CURSOR CANNOT BE MODIFIED
-514	THE CURSOR NAME <cursor_name> IS NOT IN A PREPARED STATE

Syntax Mapping Strategy

The syntax of DB2 SQL is largely compatible with that of Teradata SQL. In cases of incompatibility, TS/API ensures logical mapping between DB2 SQL and Teradata SQL when possible. When a functionally equivalent Teradata SQL syntax exists, TS/ API automatically maps the DB2 SQL syntax into the corresponding Teradata SQL.

If no equivalent exists, TS/API handles the mapping in one of two ways:

- Null mapping
- Syntax error

TS/API's primary goal in handling DB2 syntax that has no Teradata SQL equivalent is to eliminate confusion. In cases where a DB2 SQL statement performs a function not supported by Teradata Database, where no misleading results will occur, and where no Teradata SQL equivalent exists, TS/API maps the DB2 statement into a null statement.

TS/API sends a null statement to Teradata Database. The statement accomplishes no work but returns a valid return code. TS/API does not send a null statement to Teradata Database in cases where misleading results would occur. In such cases, TS/API does no mapping and simply returns a syntax error. Refer to the tables under DB2 Syntax Mapping for examples of null mapping and syntax errors.

The following translation notes define some differences in syntax between Teradata SQL and DB2 SQL.

DB2 Physical Database Structures

The DATABASE, TABLESPACE, BUFFERPOOL, and STOGROUP objects of DB2 are not supported in Teradata SQL. TS/API loosely considers Teradata Database to be analogous to the DB2 authorization id, not the DB2 DATABASE object. When encountered, these constructs are removed from the SQL statement, except in some cases involving the DATABASE object.

Referential Integrity

The referential integrity syntax of DB2 is supported in Teradata SQL.

Subqueries

The following functions are supported in Teradata SQL:

- The EXISTS clause
- ANY, ALL, and SOME predicates
- Correlated subqueries

SQL Reserved Word Conflicts

The reserved word lists of DB2 and Teradata SQL differ greatly. As a result, names for tables, views, columns, and other database objects used in DB2 or SQL/DS may conflict with the Teradata SQL reserved word list.

In situations where a DB2 non-keyword is a Teradata SQL keyword, TS/API adds quotation marks around the offending word to ensure that Teradata Database treats the word as an object name and not as a keyword.

FIELDPROC, EDITPROC, and VALIDPROC

The FIELDPROC, EDITPROC, and VALIDPROC options of DB2 are not currently supported in Teradata SQL, nor does TS/API provide a functional equivalent. If encountered, TS/API strips them from the statement.

DATE and TIME Functions

DATE and TIME functions are not currently supported in Teradata SQL. However, date arithmetic and comparisons are supported within the functionality available with dates on Teradata Database. When encountered, these functions generate a syntax error (see the following table).

Special Registers

Type	Action
USER	Passed to Teradata Database unchanged.
CURRENT DATE	Becomes DATE (FORMAT 'YYYY-MM-DD'). Because of this formatting, arithmetic operations cannot be performed.
CURRENT TIME	Becomes TIME.
CURRENT TIMESTAMP	Becomes DATE (FORMAT 'YYYY-MM-DD)||'-' ||TIME (FORMAT '99:99:99.99999) TITLE TimeStamp. The use of a labeled duration of DAYS causes the DAYS word to be removed from the statement.

	Other labeled durations are not supported.

Teradata SQL Extensions Differences

The following are those Teradata SQL Extensions that may affect the results returned to the application.

Long Teradata SQL Names

Teradata SQL allows objects to have names up to 30 characters in length. This is longer than limits imposed by DB2 or SQL/DS. Depending on QMF, Teradata SQL names that exceed DB2 limits may not function properly.

The WITH Clause

The WITH clause in Teradata SQL causes subtotal rows to be returned with the detail data rows of a SELECT query. Since QMF does not expect such rows, TS/ API excludes the subtotal rows from being returned to the application.

Sorting Nulls

Teradata SQL sorts nulls after all other data in a column. DB2 sorts nulls before any other data in a column. This difference may cause errors in applications that expect nulls to be handled identically to DB2.

DB2 Syntax Mapping

This section covers TS/API syntax mapping from DB2 SQL to Teradata SQL.

The following are DB2 commands with an explanation of how each is mapped to Teradata SQL.

ALTER INDEX

Named indexes are not supported in Teradata SQL, nor does TS/API provide a functional equivalent. TS/API translates ALTER INDEX into a null statement.

ALTER STOGROUP

The STOGROUP object of DB2 is not supported in Teradata SQL. TS/API translates ALTER STOGROUP into a null statement.

ALTER TABLE

TS/API performs data type translations as necessary.

The DB2 TIME data type translates to INTEGER FORMAT ‘99:99:99’, and the DB2 TIMESTAMP data type translates to ‘CHAR(26)’. FIELDPROC and VALIDPROC options of DB2 are not currently supported in Teradata SQL, nor does TS/API provide a functional equivalent. Upon encountering them, TS/API strips them from the statement.

Note: Teradata Database limits table size to a maximum of 256 columns.

ALTER TABLESPACE

The TABLESPACE object of DB2 is not supported in Teradata SQL. TS/API translates ALTER TABLESPACE into a null statement.

COMMENT ON

For a table or a view:

TABLE <tablename>

identical syntax to the Teradata Database COMMENT ON TABLE

TABLE <viewname>

automatically mapped to the Teradata Database COMMENT ON VIEW

For single-column syntax:

COLUMN

identical syntax to the Teradata Database COMMENT ON COLUMN

For multi-column syntax:

<tablename>

automatically translated to one or more Teradata Database COMMENT ON COLUMN statements

<viewname>

automatically translated to one or more Teradata Database COMMENT ON COLUMN statements

COMMIT (WORK)

TS/API issues an ET.

CREATE ALIAS

TS/API issues an error message.

CREATE DATABASE

TS/API removes the STOGROUP and BUFFERPOOL options from the query if they are present. TS/API provides a default PERMSIZE of 10,000 bytes and takes all other defaults in effect for the initial allocation. If the initial space allocation or other parameter values are insufficient, use the Teradata SQL MODIFY DATABASE command to modify them.

CREATE INDEX

Named indexes, VSAM-related options, and descending order are not permitted on the Teradata Database pre-V2R2 software, nor does TS/API provide a functional equivalent. TS/ API strips the named index and VSAM-related information from the query and passes it to Teradata Database. TS/API passes the descending order parameter to Teradata Database and Teradata Database generates an error message indicating that the user must check the syntax of the statement. Named Indexes are supported in V2R2 Teradata SQL. must chercheck the syntax of the statement. Named Indexes are supported in V2R2 Teradata SQL.

CREATE STOGROUP

The STOGROUP object of DB2 is not supported in Teradata SQL. TS/API translates CREATE STOGROUP into a null statement.

CREATE SYNONYM

The SYNONYM object does not exist in Teradata SQL. TS/API automatically translates it to the corresponding Teradata SQL CREATE VIEW statement.

CREATE TABLE

TS/API performs data type translations as necessary.

Teradata Database has no physical or logical equivalent to the DB2 DATABASE or TABLESPACE. The Teradata Database DATABASE is equivalent to a DB2 authorization id, not a DB2 DATABASE. Therefore, TS/API strips the 'IN database.tablespace' and 'IN DATABASE database' clauses from the CREATE TABLE statement.

The DB2 TIME data type translates to INTEGER FORMAT '99:99:99', and the DB2 TIMESTAMP data type translates to 'CHAR(26)'. FIELDPROC, EDITPROC, and VALIDPROC options of DB2 are not currently supported in Teradata SQL, nor does TS/API provide a functional equivalent. If encountered, TS/API strips them from the statement.

DB2 tables can be created with 51 or more columns with one CREATE TABLE statement. The Teradata Database pre-V2R2 software requires that any table of over 50 columns have an accompanying ALTER TABLE statement for each additional 50 columns. TS/API automatically creates and executes any ALTER TABLE statements required to create the table in its entirety and manages the accompanying unit of work.

If a DB2 CREATE INDEX statement immediately follows a DB2 CREATE TABLE statement, TS/API automatically strips the index information from the CREATE INDEX statement and re-creates the table with a Teradata SQL PRIMARY INDEX clause matching the index definition given on the CREATE INDEX statement. This avoids a potential performance and data distribution problem that may be encountered when Teradata Database assumes a default primary index. If performance problems occur, modify the CREATE TABLE statement used to build the table by explicitly using Teradata SQL syntax with the proper PRIMARY INDEX specification.

Note: Teradata Database limits table size to a maximum of 256 columns.

CREATE TABLESPACE

The TABLESPACE object of DB2 is not supported in Teradata SQL. TS/API translates CREATE TABLESPACE into a null statement.

CREATE VIEW

The DB2 syntax is identical to Teradata SQL syntax, with the exception of the WITH CHECK and DISTINCT options, which TS/API strips from the statement.

DELETE FROM

The DB2 syntax is identical to the Teradata SQL syntax.

DROP

ALIAS

TS/API issues an error message.DATABASE

DB2 syntax is identical to the Teradata SQL syntax. TS/API considers the Teradata Database DATABASE to be analogous to the DB2 authorization id, not the DB2 database object. Therefore, when using this command, be certain that you really want to drop the Teradata Database database object.

INDEX

The Teradata Database pre-V2R2 software does not support Named Indexes. TS/API passes the statement unaltered to Teradata Database, which generates a syntax error. To drop indexes through TS/API, use a valid Teradata SQL DROP INDEX statement, specifying the index column names. Named Indexes are supported in V2R2 Teradata SQL.

STOGROUP

TS/API translates to a null statement.

SYNONYM

TS/API translates to the corresponding DROP VIEW statement.

TABLE

DB2 syntax is identical to the Teradata SQL syntax.

INDEX

The Teradata Database pre-V2R2 software does not support Named Indexes. TS/API passes the statement unaltered to Teradata Database, which generates a syntax error. To drop indexes through TS/API, use a valid Teradata SQL DROP INDEX statement, specifying the index column names. Named Indexes are supported in V2R2 Teradata SQL.

STOGROUP

TS/API translates to a null statement.

SYNONYM

TS/API translates to the corresponding DROP VIEW statement.

TABLE

DB2 syntax is identical to the Teradata SQL syntax.

TABLESPACE

TS/API translates to a null statement.

VIEW

DB2 syntax is identical to the Teradata SQL syntax.

EXPLAIN

The DB2 EXPLAIN syntax is not supported in Teradata SQL, nor does TS/API provide a functional equivalent. TS/API passes the DB2 statement as is to Teradata Database, which generates a syntax error. See the Teradata Database EXPLAIN statement in SQL Fundamentals or SQL Quick Reference, for details on getting similar information from Teradata Database.

GRANT

The GRANT syntax is explained in GRANT.

GRANT (DB2 Database Privileges)

DB2 Authority	Maps To	Teradata Database Authority
DBADM		ALL
DBCTRL		DATABASE, MACRO, TABLE, USER, VIEW
DBMAINT		SELECT

CREATETAB		CREATE TABLE
		CREATE VIEW

CREATETS		Null mapping
DISPLAYDB		Null mapping
DROP		DROP DATABASE
IMAGCOPY		Null mapping
LOAD		RESTORE, DUMP
RECOVERDB		Null mapping
REORG		Null mapping
REPAIR		Null mapping
STARTDB		Null mapping
STATS		CHECKPOINT
STOPDB		Null mapping

GRANT (DB2 Plan Privileges)

DB2 Authority	Maps To	Teradata Database Authority
BIND		CREATE MACRO
EXECUTE		EXECUTE

GRANT (DB2 System Privileges)

DB2 Authority	Maps To	Teradata Database Authority
BINDADD		Syntax error
BSDS		Null mapping
CREATEDBA		Syntax error
CREATEDBC		Syntax error
CREATESG		Null mapping
DISPLAY		Null mapping
RECOVER		Null mapping
STOPALL		Null mapping
STOSPACE		Null mapping
SYSADM		ALL
SYSOPR		Syntax error
TRACE		Null mapping

GRANT (DB2 Table Privileges)

DB2 Authority	Maps To	Teradata Database Authority
ALL		ALL
ALTER		Syntax error
DELETE		DELETE
INDEX		Syntax error
INSERT		INSERT
SELECT		SELECT
UPDATE		UPDATE
UPDATE column name		Syntax error

GRANT (DB2 Use Privileges)

DB2 Authority	Maps To	Teradata Database Authority
USE OF		Null statement

INSERT

The DB2 syntax is identical to the Teradata SQL syntax.

LABEL ON

TABLE

TS/API translates to Teradata SQL COMMENT ON TABLE.

COLUMN

TS/API translates to Teradata SQL ALTER TABLE <table name> ADD <column name> TITLE ‘string constant’. VIEW COLUMNS not supported.

table name

TS/API translates to Teradata SQL ALTER TABLE <table name> ADD <column name> TITLE ‘string constant’ for each column specified.

view name

Not supported onTeradata Database. TS/API issues an error message.

LOCK TABLE

The DB2 syntax is identical to the Teradata SQL syntax.

REVOKE

The REVOKE statement is the exact opposite of the GRANT statement in all cases. See GRANT statement explanations for the mapping of the authority of REVOKE.

ROLLBACK [WORK]

TS/API issues ROLLBACK WORK.

SELECT

The DB2 syntax is identical to the Teradata SQL syntax. If the expression is passed without a Teradata SQL NAMED clause, TS/API adds the appropriate name. For information about the extended form of the SELECT statement, see SELECT under Teradata SQL Extensions.

UPDATE

The DB2 syntax is identical to the Teradata SQL syntax.

Teradata SQL Extensions

Teradata SQL Extensions are syntax capabilities that extend beyond normal ANSI, DB2 syntax. The extensions are supported only on Teradata Database. This section explains the extensions by command. For detailed information on Teradata SQL commands, see *SQL Fundamentals* or *SQL Quick Reference*.

These Teradata SQL extensions can easily be exploited by executing QMF SQL queries containing Teradata SQL extensions syntax.

ABORT

The ABORT command may be used in place of the DB2 ROLLBACK. Most applications automatically manage the unit of work for the user. ABORT allows the user to abnormally terminate a unit of work.

BEGIN TRANSACTION (BT)

The BT command may be used in place of the DB2 COMMIT. Most applications automatically manage the unit of work for the user. BEGIN TRANSACTION allows the user to normally terminate a unit of work and begin a new unit of work.

CHECKPOINT

CHECKPOINT places a mark in a journal table, to aid in later recovery.

COLLECT STATISTICS

COLLECT STATISTICS obtains statistical data for one or more columns of a table that may be used by Teradata Database to optimize data access.

COMMENT ON

The COMMENT ON command is extended to support comments on the following:

- DATABASE
- USER
- MACRO

The COMMENT ON command can return data. For the data-returning form of this command, see the SELECT statement detailed later in this section.

CREATE

The CREATE command is extended to support the following:

- DATABASE
- USER
- MACRO
- DATABASE

  The DATABASE command establishes a default database for the current session.

DELETE

The DELETE command is extended to remove all objects from the following:

- DATABASE
- USER

DROP

The DROP command is extended to drop the following (as long as they are empty):

- DATABASE
- USER

DROP INDEX

The DROP INDEX command allows you to drop a secondary index, specifying the index column names.

DROP MACRO

The DROP MACRO command removes a macro definition from a database.

DROP STATISTICS

The DROP STATISTICS command drops statistical data that was created by a previous COLLECT STATISTICS command.

END TRANSACTION (ET)

The ET command may be used in place of the DB2 and SQL/DS COMMIT. Most applications automatically manage the unit of work for the user. END TRANSACTION allows the user to normally terminate a unit of work.

EXECUTE macroname

The EXECUTE macroname command executes a macro that was previously defined using the CREATE MACRO command. The data-returning form of the EXECUTE macroname command must contain a single statement consisting of one of the following data-returning Teradata SQL commands:

- COMMENT ON
- EXPLAIN
- HELP
- SELECT
- SHOW

GIVE

The GIVE command transfers ownership of a database or a user space to another user.

GRANT

The GRANT command is extended to support the following:

- CREATE DATABASE
- CREATE MACRO
- CREATE TABLE
- CREATE USER
- CREATE VIEW
- DROP DATABASE
- DROP MACRO
- DROP TABLE
- DROP USER
- DROP VIEW
- DATABASE
- MACRO
- TABLE
- USER
- VIEW
- EXECUTE
- GRANT (in Version 1 only)
- DUMP
- RESTORE
- CHECKPOINT

HELP

The HELP command obtains Data Dictionary/Directory information about a specified database, user, table, view, macro, column, or index. For the data-returning form of this command, see the Execute macroname statement.

MODIFY

The MODIFY command changes options specified at creation time for the following:

- DATABASE
- USER

Multi-Statement Requests

Teradata Database is capable of receiving two or more Teradata SQL statements in one request and handling them in parallel. This type of request is called a multi-statement request. Three restrictions to this capability exist:

- If present, a DDL statement (ALTER, CREATE, or DROP) must be the last statement in a multi-statement request. This is a Teradata Database restriction.

- The SELECT or any other data-returning command cannot be included in a multi- statement request. This is a TS/API restriction.

Teradata Database's parallel processing capability for multi-statement requests applies to the following commands:

- DELETE
- INSERT
- UPDATE

RENAME

The RENAME command renames the following existing objects:

- MACRO
- TABLE
- VIEW

REPLACE MACRO, REPLACE VIEW

These commands replace an existing macro or view.

REVOKE

The Extensions to the REVOKE statement are the exact opposite of the Extensions to the GRANT statement in all cases. See the Teradata SQL Extensions Differences statement Extensions earlier in this section for the authority of REVOKE.

SELECT

The normal Teradata SQL SELECT can be used to return data to the user. Additionally, the Extensions to the following data-returning commands return data to the user when preceded by the SELECT:

- COMMENT ON
- EXECUTE or EXEC
- EXPLAIN
- HELP
- SHOW

Teradata SQL statements do not properly return data to the user unless the above commands are preceded by a SELECT command. This is due to QMF data retrieval logic.

TS/API automatically removes the SELECT command from the statement and passes the remainder of the data-returning statement on to Teradata Database for normal handling. This technique provides the capability of using the Teradata SQL Extensions data-returning commands via TS/API. The following example shows how to use the data-returning form of the SHOW command:

```
SELECT SHOW TABLE PERSONNEL.EMPLOYEE
```

SHOW

The SHOW command displays the data definition statement most recently used to create, modify, or replace the specified macro, view, or table. For the data-returning form of this command, see Execute macroname statement detailed earlier in this section.

TS/API Directives

Directives are statements that can be issued from within the DBCLOGON file or from the application passthrough screen. Most commonly, they control CLIv2 options or some form of special processing not directly related to Teradata SQL.

The following are TS/API Directives, with a brief description of each.

DEBUG

Activates the TS/API debug facility. See TS/API Debug Facility for details.

LOGON

The LOGON statement takes the form:

```
LOGON [<tdpid>/] <userid>, <password>;
```

Note: The symbols, < and >, are not part of the specification. Replace these symbols and the text within them with the appropriate information.

Unless you use a logon exit, the logon statement in the DBCLOGON file is used to build the CLIv2 logon request to Teradata Database.

LOGOFF

The LOGOFF command logs you off the current session.

Any outstanding unit of work is automatically committed prior to the execution of either the LOGON or LOGOFF commands.

DEBUG PERF

This directive causes TS/API to create performance monitoring information. TS/API places the performance monitoring information in the TS/API OUTPUT file.

Performance monitoring is acquired from the timestamp fields located in the DBCAREA CLIv2 communication block. The following table describes these fields.

Timestamp fields in CLIv2's DBCAREA communication block

HSISVC_Time	The date and time that the request left TS/API.
TDPWAIT_Time	The time the request arrived at the TDP from CLIv2.
TDPDBO_Time	The time the request was sent to Teradata Database by the TDP.
TDPDBI_Time	The time the response arrived at the TDP from Teradata Database.
TDPXMM_Time	The time the TDP sent the response to CLIv2.
SRBSCHED_Time	The time the CLI response was sent to TS/API.

The TS/API OUTPUT file displays the following performance information:

- The actual clock time when the performance data displayed, retrieving from the system clock.
- The amount of time CLIv2 spent to process the request
- The amount of time the TDP spent to process the request.
- The amount of time Teradata Database spent to process the request.
- The amount of time the TDP spent to process the response.
- The amount of time the CLIv2 spent to process the response.
- The longest Teradata Database response time during the performance trace, and when the corresponding request was issued.

Note: All time durations are represented in seconds.

See TS/API Debug Facility for details on the TS/API OUTPUT file.

SET SESSION (SS)

The following SET SESSION commands are supported:

CHARSET

If an error occurs in naming the character set, the character set is automatically reset to the CLIv2 default.

COLLATION

Changes collating sequence to specified value.

DATABASE

Changes the current database to the specified database. It is identical to the Teradata SQL DATABASE command.

DATEFORM

Sets the import-export dateform mode for a session.

SET SESSION CHARSET

The SET SESSION CHARSET command is compatible with its BTEQ equivalent. The SET SESSION CHARSET command sets the name of the character set for the current session. Two views exist in conjunction with this command: HostsInfo View and CharTranslation View. The HostsInfo View defines the default character set for your client. The CharTranslation View defines the character sets available in Teradata Database.

The SET SESSION CHARSET command specifies the name of the character set to be used for the current session. The character sets are user-definable and defined in the CharTranslations View.

You can choose the character set by name. The name must exist in the CharSetName column in the CharTranslations View.

The SET SESSION CHARSET command takes the following form:

```
SET SESSION CHARSET ['charstring']
```

where:

```
charstring
```

specifies the name of the character set to be used. The name cannot exceed 30 characters and it must be enclosed in either single (‘) or double (“) quotes.

This command takes effect only after TS/API logs on to Teradata Database. That means the logon string itself should be coded using the client's default character set.

SET SESSION COLLATION

The SET SESSION COLLATION command is a Teradata SQL command. The SET SESSION COLLATION command overrides the collation currently in effect for the session.

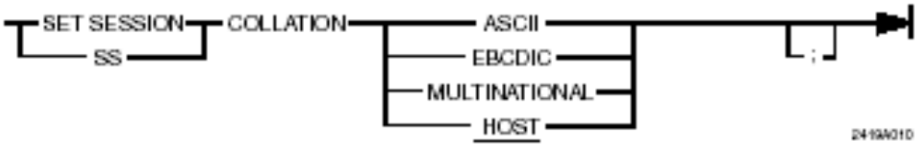
The collation for a session determines the ordering of data characters during comparison operations, and when sorting data in response to a SELECT request that includes a WITH...BY or ORDER BY clause.

COLLATION can be defined as an attribute of the user.

If the attribute is not defined, then collation for the session defaults to that of the logon client.

The SET SESSION COLLATION command is used to define collation after a session is started.

The SET SESSION COLLATION command takes the form:



where:

Syntax element...	is the...
ASCII	Specifies that comparison and sort operations are to use ASCII collation.
EBCDIC	Specifies that comparison and sort operations are to use EBCDIC collation.
MULTINATIONAL	Specifies that comparison and sort operations are to use the International sort sequence. If the MULTINATIONAL option is selected, Teradata Database must be set up for International Character Support and the hashing algorithm must be defined for diacritical characters.
HOST	This is the default. Specifies that collation for the session is to agree with the collation of the logon client (EBCDIC for IBM mainframe clients, ASCII for all others).

The MULTINATIONAL option is available only if the hashing algorithm of Teradata Database is set up for International Character Support. Otherwise, this option returns an error. MULTINATIONAL sets collation for the session to the International sort sequence, as described under the ORDER BY clause of the SELECT statement.

Teradata Database converts data characters to their uppercase values for comparison and sorting operations unless the CASESPECIFIC option is included in the Teradata SQL request, or was defined at creation time for the column being queried.

Character data is sorted in ascending order unless the DESC (descending) option is included in the Teradata SQL request. The results of CASESPECIFIC on sorted results is discussed under the ORDER BY clause of the SELECT statement.

SET SESSION DATABASE

The SET SESSION DATABASE command is a Teradata SQL command. It identifies the database to be used during the current session for all Teradata SQL statements that are entered without fully qualified table, view, or macro names.

This default database is used until the end of the session, or until a subsequent SET SESSION DATABASE command is entered.

The SET SESSION DATABASE command takes the form:

```
SET SESSION DATABASE dbname
```

where:

`dbname`

specifies the name of the default database.

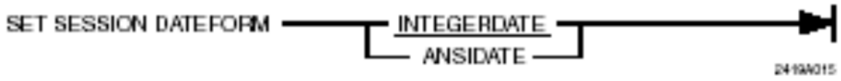
SET SESSION DATEFORM

The SET SESSION DATEFORM command is a Teradata SQL command. It sets the import-export dateform mode for a session.

The following rules apply to SET SESSION DATEFORM:

- Enter the SET SESSION DATEFORM statement as follows:
  - SET SESSION DATEFORM = INTEGERDATE
  - SET SESSION DATEFORM = ANSIDATE
- Depending on the DATEFORM mode specified, if INTEGERDATE, the DATE values are returned in INTEGER form; if ANSIDATE, the DATE values are returned in CHAR (10) form.
- The DATEFORM mode can also be set after a session is logged on.
- A setting of DATEFORM after logon will continue only for the life of the session, or until another SET SESSION DATEFORM statement is issued.
- The default DATEFORM for a session is INTEGERDATE

The SET SESSION DATEFORM command takes the form:



APPENDIX A

Translation Tables

Overview

This appendix contains error code translation tables.

Error Codes

This appendix contains translation tables that supplement the information provided in Advanced Problem Determination (DEBUG ON) For additional information, see the *Teradata Messages Guide*.

Error Code Translations for DB2

The following two tables list DB2 SQLCODEs and the corresponding Teradata SQL error codes. The content of the first table is listed in SQLCODE sequence. The content of the second table is listed in Teradata Database error code sequence.

Error Code Translations for DB2

SQL-CODE	Teradata Database Error Code
-010	3760
-060	3527, 3528, 3529, 3530, 3617
-101	2664, 3509, 3540, 3597, 3609, 3629, 3702, 3705, 3710, 3712, 3714, 3741, 3850, 3851, 3867, 3896
-102	3738
-103	3751, 3752, 3759
-104	2644, 2667, 3521, 3525, 3531, 3536, 3541, 3543, 3544, 3551, 3552, 3553, 3557, 3558, 3559, 3561, 3562, 3567, 3576, 3579, 3585, 3588, 3590, 3605, 3612, 3623, 3624, 3625, 3630, 3631, 3632, 3633, 3634, 3636, 3645, 3646, 3648, 3649, 3703, 3706, 3707, 3708, 3709, 3727, 3728, 3735, 3736, 3739, 3761, 3763, 3764, 3765, 3766, 3768, 3770, 3771, 3772, 3773, 3776, 3779, 3781, 3782, 3783, 3784, 3786, 3788, 3792, 3793, 3796, 3797, 3798, 3806, 3808, 3815, 3849, 3852, 3853, 3854, 3855, 3859, 3860, 3861, 3863, 3870, 3871, 3873, 3874, 3875, 3876, 3877, 3878, 3879, 3882, 3886, 3887, 3888, 3890, 3926, 3933, CLI530, 3750, 3952, 3958, 3964, 3965, 3967, 5316

-107	3737
-109	3507
-110	3704, 3775, 3956
-112	3568, 3627, 3628
-113	3696, 3697, 3957
-117	3812, 3813
-119	3554
-120	3569, 3574, 3872
-121	3606, 3885
-122	3504, 3883
-125	3637
-128	3731
-137	3578
-138	2662, 2663
-150	3823
-151	3659
-156	3891
-170	3816, 3817, 3818, 3820, 3821
-171	2603, 2604, 2605, 2606, 2607, 2608, 2622, 2623, 3580, 3581, 3647, 3660, 3662, 3663, 3819, 3857, 3949, 3950, 3951, 3963, 3966
-172	3732
-182	2665, 2666
-199	3516
-203	3809. 3822, 3868
-204	3526, 3539, 3656, 3802, 3807, 3824
-205	3810, 5628
-207	3848
-312	3594, 3595, 3599, 3600
-313	2673, 3593
-401	2147, 2149, 3639, 3640, 3959, 3960
-402	3622, 3643, 3644
-404	3520, 3564
-405	3753

-407	2689, 3604, 3811
-408	3814
-415	3654
-421	3607, 3608, 3653
-530	2700
-551	3523, 3524, 3856, 3858, 3865, 3866, 3880, 3881
-552	3545
-554	3542
-601	3519, 3744, 3801, 3803, 3804, 3805
-602	3518
-603	3534
-604	3546, 3729
-612	3515, 3517, 3560
-618	3571
-637	3733, 3789, 3889
-680	3556, 3582, 3919
-802	2161, 2162, 2163, 2164, 2165, 2166, 2232, 2233, 2239, 2240, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2661, 2674, 2675, 2676, 2677, 2678, 2679, 2682, 2683, 2684, 2685, 2686, 2687, 3532, 3535, 3641, 3642, 3754, 3755, 3756, 3757, 3758, 3953, 3954, 3955, 3961, 3962
-803	2801, 2802, 2803
-811	3669
-901	2827, 2828, 2938, 3110, 3120, 3513, 3514, 3897
-905	2805, 2843, 2977, 3130, 3149,3150,3151,3152,3153,3157,3159, 3566, 3577, 3638, 3661
-911	2450, 2631, 2825, 2826, 3111, 3603
-922	3002, 3003, 3004, 3014, 3015, 3016, 3023, CLI040, CLI041, CLI272, CLI370, CLI521, CLI524, CLI527
-923	3006, 3007, CLI151, CLI155, CLI280, CLI282, CLI368, CLI369, CLI426, CLI427, CLI512, CLI513, CLI514

Error Code Translations for DB2

Teradata Database Error Code	SQLCODE
2147	-401
2149	-401
2161	-802
2162	-802

2163	-802
2164	-802
2165	-802
2166	-802
2232	-802
2233	-802
2239	-802
2240	-802
2450	-911
2603	-171
2604	-171
2605	-171
2606	-171
2607	-171
2608	-171
2614	-802
2615	-802
2616	-802
2617	-802
2618	-802
2619	-802
2620	-802
2621	-802
2622	-171
2623	-171
2631	-911
2644	-104
2661	-802
2662	-138
2663	-138
2664	-101
2665	-182
2666	-182

2667	-104
2673	-313
2674	-802
2675	-802
2676	-802
2677	-802
2678	-802
2679	-802
2682	-802
2683	-802
2684	-802
2685	-802
2686	-802
2687	-802
2689	-407
2700	-530
2801	-803
2802	-803
2803	-803
2805	-905
2825	-911
2826	-911
2827	-901
2828	-901
2843	-905
2938	-901
2977	-905
3002	-922
3003	-922
3004	-922
3006	-923
3007	-923
3014	-922

3015	-922
3016	-922
3023	-922
3110	-901
3111	-911
3120	-901
3130	-905
3149	-905
3150	-905
3151	-905
3152	-905
3153	-905
3157	-905
3159	-905
3504	-122
3507	-109
3509	-101
3513	-901
3514	-901
3515	-612
3516	-199
3517	-612
3518	-602
3519	-601
3520	-404
3521	-104
3523	-551
3524	-551
3526	-204
3527	-060
3528	-060
3529	-060
3530	-060

3531	-104
3532	-802
3534	-603
3535	-802
3536	-104
3539	-204
3540	-101
3541	-104
3542	-554
3543	-104
3544	-104
3545	-552
3546	-604
3551	-104
3552	-104
3553	-104
3554	-119
3556	-680
3557	-104
3558	-104
3559	-104
3560	-612
3561	-104
3564	-404
3566	-905
3567	-104
3568	-112
3569	-120
3571	-618
3574	-120
3576	-104
3577	-905
3578	-137

3579	-104
3580	-171
3581	-171
3582	-680
3585	-104
3588	-104
3590	-104
3593	-313
3594	-312
3595	-312
3597	-101
3599	-312
3600	-312
3603	-911
3604	-407
3605	-104
3606	-121
3607	-421
3608	-421
3609	-101
3612	-104
3617	-060
3622	-402
3623	-104
3624	-104
3625	-104
3627	-112
3628	-112
3629	-101
3631	-104
3632	-104
3633	-104
3634	-104

3636	-104
3637	-125
3638	-905
3639	-401
3640	-401
3641	-802
3642	-802
3643	-402
3644	-402
3645	-104
3646	-104
3647	-171
3648	-104
3649	-104
3653	-421
3654	-415
3656	-204
3659	-151
3660	-171
3661	-905
3662	-171
3663	-171
3669	-811
3696	-113
3697	-113
3702	-101
3703	-104
3704	-110
3705	-101
3706	-104
3707	-104
3708	-104
3709	-104

3710	-101
3712	-101
3714	-101
3727	-104
3728	-104
3729	-604
3731	-128
3732	-172
3733	-637
3735	-104
3736	-104
3737	-107
3738	-102
3739	-104
3741	-101
3744	-601
3750	-104
3751	-103
3752	-103
3753	-405
3754	-802
3755	-802
3756	-802
3757	-802
3758	-802
3759	-103
3760	-010
3761	-104
3763	-104
3764	-104
3765	-104
3766	-104
3768	-104

3770	-104
3771	-104
3772	-104
3773	-104
3775	-110
3776	-104
3779	-104
3781	-104
3782	-104
3783	-104
3784	-104
3786	-104
3788	-104
3789	-637
3792	-104
3793	-104
3796	-104
3797	-104
3798	-104
3801	-601
3802	-204
3803	-601
3804	-601
3805	-601
3806	-104
3807	-204
3808	-104
3809	-203
3810	-205
3811	-407
3812	-117
3813	-117
3814	-408

3815	-104
3816	-170
3817	-170
3818	-170
3819	-171
3820	-170
3821	-170
3822	-203
3823	-150
3824	-204
3848	-207
3849	-104
3850	-101
3851	-101
3852	-104
3853	-104
3854	-104
3855	-104
3856	-551
3857	-171
3858	-551
3859	-104
3860	-104
3861	-104
3863	-104
3865	-551
3866	-551
3867	-101
3868	-203
3870	-104
3871	-104
3872	-120
3873	-104

3874	-104
3875	-104
3876	-104
3877	-104
3878	-104
3879	-104
3880	-551
3881	-551
3882	-104
3883	-122
3885	-121
3886	-104
3887	-104
3888	-104
3889	-637
3890	-104
3891	-156
3896	-101
3897	-901
3919	-680
3926	-104
3933	-104
3949	-171
3950	-171
3951	-171
3952	-104
3953	-802
3954	-802
3955	-802
3956	-110
3957	-113
3958	-104
3959	-401

3960	-401
3961	-802
3962	-802
3963	-171
3964	-104
3965	-104
3966	-171
3967	-104
3978	-104
5316	-104
5317	-104
5628	-205
6706	-104
CLI040	-922
CLI041	-922
CLI151	-923
CLI155	-923
CLI272	-922
CLI280	-923
CLI282	-923
CLI368	-923
CLI369	-923
CLI370	-922
CLI426	-923
CLI427	-923
CLI512	-923
CLI513	-923
CLI514	-923
CLI521	-922
CLI524	-922
CLI527	-922
CLI530	-104

APPENDIX B

TS/API Catalog Emulation

Overview

This appendix describes the SYSIBM and SYSTEM databases on Teradata Database.

Catalog Tables and Views

This appendix contains a description of the SYSIBM database on Teradata Database. It supplements the information in DB2 SYSIBM System Catalog Tables Emulation. TS/API provides this database to emulate the system catalog for DB2.

Catalog Emulation Table/View

This appendix describes each TS/API catalog emulation table/view, showing the following:

- The name, type, and length of each column in the DB2 table being emulated
- The Teradata Database table and name, type, and length of the column used to derive the value

Sometimes the value for a column in one of the DB2 system catalog tables cannot be derived from the Teradata Database system catalog tables. For these columns, a constant or calculation may be used instead of an actual Teradata Database catalog column. If that is the case, the Teradata Database table will not be shown, and a constant value or the italicized word *calculated* will appear in place of the column name.

To determine how a calculated column value is derived, you must inspect the view definition for the TS/API catalog view. Do this by doing one of the following:

- Issuing the SELECT SHOW VIEW command to TS/API
- Issuing the SHOW VIEW command in BTEQ
- Looking at the BTEQ scripts used to create the TS/API catalog views

The SYSIBM database uses views of the Teradata Database catalog tables to emulate DB2's SYSIBM system catalog tables. Catalog Tables and Views describes the DB2 catalog tables.

DB2 SYSIBM Catalog

DB2 Table Name	What It Describes
SYSIBM.SYSCOLAUTH	DB2 update privileges on columns of tables and views
SYSIBM.SYSCOLUMNS	Each column in a table or view
SYSIBM.SYSCOPY	Recovery information
SYSIBM.SYSDATABASE	Database DSNDB04
SYSIBM.SYSDBAUTH	Database privileges
SYSIBM.SYSDBRM	DB2 DBRM for each application plan
SYSIBM.SYSFIELDS	Field procedure for each column of tables or views
SYSIBM.SYSFOREIGNKEYS	Table foreign keys
SYSIBM.SYSINDEXES	Table indexes
SYSIBM.SYSINDEXPART	Unpartitioned indexes
SYSIBM.SYSKEYS	Columns that are part of index keys
SYSIBM.SYSLINKS	Links between tables
SYSIBM.SYSPLAN	DB2 plans
SYSIBM.SYSPLANAUTH	Application plan privileges
SYSIBM.SYSPLANDEP	DB2 plan dependencies
SYSIBM.SYSRELS	Link characteristics

DB2 SYSIBM Catalog

DB2 Table Name	What It Describes
SYSIBM.SYSRESAUTH	DB2 resource privileges
SYSIBM.SYSSTMT	SQL statements of each DBRM
SYSIBM.SYSSTOGROUP	Storage groups
SYSIBM.SYSSYNONYMS	DB2 synonyms of tables and views
SYSIBM.SYSTABAUTH	Table and view privileges
SYSIBM.SYSTABLEPART	Unpartitioned table spaces
SYSIBM.SYSTABLES	Tables or views
SYSIBM.SYSTABLESPACE	DB2 table spaces
SYSIBM.SYSUSERAUTH	System privileges
SYSIBM.SYSVIEWDEP	View dependencies on a table and other views
SYSIBM.SYSVIEWS	Views
SYSIBM.SYSVLTREE	Remaining part of DB2 parse tree representations

SYSIBM.SYSVOLUMES	Volume of each DB2 storage group
SYSIBM.SYSVTREE	DB2 parse tree of views

SYSIBM.SYSCOLAUTH

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. This view is furnished to support DB2 application programs that may interrogate for DB2 column authorizations. Because Teradata Database does not support column authorizations, zero rows are always returned from any query. SYSIBM.SYSCOLAUTH describes SYSIBM.SYSCOLAUTH.

SYSIBM.SYSCOLAUTH Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
GRANTOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
GRANTEE	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
GRANTEETYPE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
CREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
TNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
TIMESTAMP	CHAR(12)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(12)
DATEGRANTED	CHAR(6)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(6)
TIMEGRANTED	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
COLNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
FILLER	CHAR(16)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(16)
COLLID	CHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(18)
CONTOKEN	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)

SYSIBM.SYSCOLUMNS

This view joins data from the DBC.DBASE, DBC.TVM, and DBC.TVFIELDS system catalog tables to emulate the SYSIBM.SYSCOLUMNS table. Each row defines one column from a table or view description stored in the Teradata Database system catalog. SYSIBM.SYSCOLUMNS describes SYSIBM.SYSCOLUMNS.

SYSIBM.SYSCOLUMNS Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	VARCHAR(18)	DBC.TVFIELDS	FIELDNAME	VARCHAR(31)
TBNAME	VARCHAR(18)	DBC.TVM	TVMNAME	VARCHAR(31)
TBCREATOR	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
COLNO	SMALLINT	DBC.TVFIELDS	calculated from FIELDID	SMALLINT
COLTYPE	CHAR(8)	DBC.TVFIELDS	calculated from FIELDTYPE	CHAR(8)
LENGTH	SMALLINT	DBC.TVFIELDS	calculated from FIELDTYPE	SMALLINT
SCALE	SMALLINT	DBC.TVFIELDS	calculated from FIELDTYPE	SMALLINT
			NULLABLE	
NULLS	CHAR(1)	DBC.TVFIELDS	-1	CHAR(1)
COLCARD	INTEGER		‘ ’	INTEGER
HIGH2KEY	CHAR(8)		‘ ’	CHAR(8)
LOW2KEY	CHAR(8)		‘Y’	CHAR(8)
UPDATES	CHAR(1)	DBC.TVFIELDS	‘N’	CHAR(1)
IBMREQD	CHAR(1)		COMMENTSTRING	CHAR(1)
REMARKS	VARCHAR(254)		‘N’	VARCHAR(254)
DEFAULT	CHAR(1)		0	CHAR(1)
KEYSEQ	SMALLINT		‘N’	SMALLINT
FOREIGNKEY	CHAR(1)	DBC.TVFIELDS	‘N’	CHAR(1)
FLDPROC	CHAR(1)		FIELDTITLE	CHAR(1)
LABEL	VARCHAR(30)			VARCHAR(60)

The following table explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSCOLUMNS are Supported

IBM Name	Description
NAME	Column name. The DB2 length of 18 is expanded to a length of 31 to accommodate the longer fieldnames available in Teradata SQL
TBNAME	Table or view names. The DB2 length of 18 is expanded to a length of 31 to accommodate the longer table names available in Teradata SQL
TBCREATOR	The Teradata Database userid under which the table or view was created. The DB2 length of eight is expanded to 31 to accommodate the longer userid names available in Teradata SQL
COLNO	Column number
COLTYPE	Column type of the field in DB2 format
LENGTH	Field length in DB2 format
SCALE	Scale information for DECIMAL fields in DB2 format
NULLS	Whether the field can be set to nulls

COLCARD	Set to -1 indicating that statistics have not been gathered. Teradata Database statistics are not relevant to DB2 use.
HIGH2KEY	Set to blank indicating no HIGH2KEY information. Teradata Database does not have relevant key range information.
LOW2KEY	Set to blank indicating no LOW2KEY information. Teradata Database does not have relevant key range information.
UPDATES	Set to ‘Y’ indicating that the field is updatable. If this is a view, the field may not be updatable. In that case, an error occurs if an application attempts to update the field.
IBMREQD	Set to ‘N’ indicating that the row does not come from the basic machine-readable tape.
REMARKS	Column comments
DEFAULT	Set to ‘N’ indicating that the field does not contain a default value. This may not be correct if the field contains a DEFAULT clause in the table description.
KEYSEQ	Set to zero indicating that the field is not part of the primary key. This setting may not be correct if the field is part of the primary key.
FOREIGNKEY	Set to ‘N’ indicating that the field is not part of a foreign key. TS/API does not provide support for foreign keys.
FLDPROC	Set to ‘N’ indicating that the field does not have a field procedure. Teradata Database currently does not support field procedures.
LABEL	The field title. The DB2 length of 30 is expanded to a length of 60 to accommodate the longer field titles available in Teradata SQL.

SYSIBM.SYSCOPY

This view is a SELECT of constants used to emulate the SYSIBM.SYSCOPY table, which contains information needed for recovery. The following table describes SYSIBM.SYSCOPY

SYSIBM.SYSCOPY Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
DBNAME	CHAR(8)		‘DSNDB04’	CHAR(8)
TSNAME	CHAR(8)		‘DSNDB04’	CHAR(8)
DSNUM	INTEGER		1	INTEGER
ICTYPE	CHAR(1)		‘F’	CHAR(1)
ICDATE	CHAR(6)		‘880101’	CHAR(6)
START_RBA	CHAR(6)		‘ ’	CHAR(6)
FILESEQNO	INTEGER		1	INTEGER
DEVTYPE	CHAR(8)		‘ ’	CHAR(8)
IBMREQD	CHAR(1)		‘N’	CHAR(1)
DSNAME	CHAR(44)		‘ ’	CHAR(44)
ICTIME	CHAR(6)		‘000000’	CHAR(6)
SHRLEVEL	CHAR(1)		‘ ’	CHAR(1)
DSVOLSER	VARCHAR(1784)		‘000000’	VARCHAR(1784)
TIMESTAMP	CHAR(12)		‘?’	CHAR(12)
ICBACUP	CHAR(2)		‘ ’	CHAR(2)
ICUNIT	CHAR(1)		‘ ’	CHAR(1)

The following table explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSCOPY Are Supported

IBM Name	Description
DBNAME	Database name; set to 'DSNDB04'.
TSNAME	Table space; set to 'DSNDB04'. Table spaces are not defined on Teradata Database.
DSNUM	Data set number within table space; set to 1.
ICTYPE	Set to 'F'. The operation is always full-image copy on Teradata Database.
ICDATE	Set to '880101'. The date of the entry is not defined on Teradata Database.
START_RBA	Set to blank. The DB2 log is not used on Teradata Database.
FILESEQNO	Tape file sequence number; set to 1.
DEVTYPE	Set to blank.
IBMREQD	Set to 'N', indicating that the row does not come from the basic machine-readable tape.
DSNAME	Data set name; set to blanks. The data set name is not available on Teradata Database.
ICTIME	The time at which the row was inserted; set to 000000. Not available on Teradata Database.
SHRLEVEL	Set to blank. Does not describe an image copy.
DSVOLSER	Set to '000000'. Volume serial number unavailable.
TIMESTAMP	Set to '?', indicating that no timestamp is available. This column is generally unused by users or application programs.
ICBACUP	Indicates whether the Image Copy dataset is for the primary or secondary system.
ICUNIT	Device used for Image Copy dataset.

SYSIBM.SYSDATABASE

This view is a SELECT of constants used to emulate the SYSIBM.SYSDATABASE table. The table contains one row for the DSNDB04 database. SYSIBM.SYSDATABASE describes SYSIBM.SYSDATABASE in detail.

SYSIBM.SYSDATABASE Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	CHAR(8)		'DSNDB04'	CHAR(8)
CREATOR	CHAR(8)		'SYSIBM'	CHAR(8)
STGROUP	CHAR(8)		'SYSDEFLT'	CHAR(8)
BPOOL	CHAR(8)		'BP0'	CHAR(8)
DBID	SMALLINT		4	SMALLINT
IBMREQD	CHAR(1)		'N'	CHAR(1)
CREATEDBY	CHAR(8)		'SYSIBM'	CHAR(8)

ROSHARE	CHAR(1)		‘ ’	CHAR(8)
TIMESTAMP	CHAR(12)		‘?’	CHAR(12)

The following table explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSDATABASE Are Supported

IBM Name	Description
NAME	Database name; set to ‘DSNDB04’.
CREATOR	Owner’s authorization ID; set to ‘SYSIBM’.
STGROUP	Default storage group; set to SYSDEFLT.
BPOOL	Default buffer pool; set to ‘BP0’.
DBID	Database internal identifier; set to 4.
IBMREQD	Set to ‘N’, indicating that the row does not come from the basic machine-readable tape
CREATEDBY	Primary authorization ID; set to ‘SYSIBM’
ROSHARE	Indicates whether database is read-only shared data
TIMESTAMP	The time the database became shared

SYSIBM.SYSDBAUTH

This view joins data from the DBC.DBASE and DBC.ACCESSRIGHTS catalog tables to emulate the SYSIBM.SYSDBAUTH table, which records user privileges on databases. SYSIBM.SYSDBAUTH describes SYSIBM.SYSDBAUTH.

SYSIBM.SYSDBAUTH Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
GRANTOR	CHAR(8)	DBC.ACCESSRIGHTS	GRANTORNAME	CHAR(31)
GRANTEE	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
NAME	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
TIMESTAMP	CHAR(12)		‘?’	CHAR(12)
DATEGRANTED	CHAR(6)		‘?’	CHAR(6)
TIMEGRANTED	CHAR(8)		‘?’	CHAR(8)
GRANTEETYPE	CHAR(1)		‘ ’	CHAR(1)
AUTHHOWGOT	CHAR(1)		‘D’	CHAR(1)
CREATETABAUTH	CHAR(1)	DBC.ACCESSRIGHTS	calculated from ACCESSRIGHT	CHAR(1)
CREATETSAUTH	CHAR(1)	DBC.ACCESSRIGHTS	calculated from ACCESSRIGHT	CHAR(1)
DBADMAUTH	CHAR(1)	DBC.ACCESSRIGHTS	calculated from ACCESSRIGHT ‘ ’	CHAR(1)

DBCTRLAUTH	CHAR(1)	DBC.ACCESSRIGHTS	‘ ’	CHAR(1)
DBMAINTAUTH	CHAR(1)		‘ ’	CHAR(1)
DISPLAYDBAUTH	CHAR(1)		calculated from ACCESSRIGHT	CHAR(1)
DROPAUTH	CHAR(1)		‘ ’	CHAR(1)
			‘ ’	
IMAGECOPYAUTH	CHAR(1)		‘ ’	CHAR(1)
LOADAUTH	CHAR(1)		‘ ’	CHAR(1)
REORGAUTH	CHAR(1)		‘ ’	CHAR(1)
RECOVERDBAUTH	CHAR(1)		‘ ’	CHAR(1)
REPAIRAUTH	CHAR(1)		‘ ’	CHAR(1)
STARTDBAUTH	CHAR(1)		‘ ’	CHAR(1)
STATSAUTH	CHAR(1)		‘N’	CHAR(1)
STOPAUTH	CHAR(1)			CHAR(1)
IBMREQD	CHAR(1)			CHAR(1)

Catalog Emulation Table/View explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSDBAUTH Are Supported

IBM Name	Description
GRANTOR	The Teradata Database user who granted an access right. The DB2 length of eight is expanded to 31 to accommodate the longer userid names available in Teradata SQL.
GRANTEE	The Teradata Database user who was granted an access right. The DB2 length of eight is expanded to 31 to accommodate the longer userid names available in Teradata SQL.
NAME	The name of a table or view in Teradata Database. The DB2 length of 18 characters is expanded to a length of 31 to allow for longer table names on Teradata Database.
TIMESTAMP	Set to ‘?’, indicating that no timestamp is available. This column is generally not used by users or application programs.

IBM Name	Description
GRANTOR	The Teradata Database user who granted an access right. The DB2 length of eight is expanded to 31 to accommodate the longer userid names available in Teradata SQL.
GRANTEE	The Teradata Database user who was granted an access right. The DB2 length of eight is expanded to 31 to accommodate the longer userid names available in Teradata SQL.
NAME	The name of a table or view in Teradata Database. The DB2 length of 18 characters is expanded to a length of 31 to allow for longer table names on Teradata Database.
TIMESTAMP	Set to ‘?’, indicating that no timestamp is available. This column is generally not used by users or application programs.
DATEGRANTED	Set to ‘?’, indicating that no date is available. This column is generally not used by users or application programs.
TIMEGRANTED	Set to ‘?’, indicating that no time is available. This column is generally not used by users or application programs.

GRANTEETYPE	Set to blank, reflecting that the grantee is a user. Because Teradata Database cannot grant access rights to programs, only blank is valid.
AUTHHOWGOT	Authorization of the user from whom privileges were received. Always set to ‘D’ (DBADM).
CREATETABAUTH	Whether GRANTEE can create tables within a database; two types exist: •Authority not held •Privilege held without GRANT option Authority held with GRANT option is not defined to Teradata Database.
CREATETSAUTH	Whether GRANTEE can create table spaces; two types exist: •Authority not held •Privilege held without GRANT option Authority held with GRANT option is not defined to Teradata Database.
DBADMAUTH	Whether GRANTEE has DBADM authority over database; two types exist: •Authority not held •Privilege held without GRANT option Authority held with GRANT option is not defined to Teradata Database.
DBCTRLAUTH	Whether GRANTEE has DBCTRL authority over a database; not defined to Teradata Database. Blank is returned.
DBMAINTAUTH	Whether GRANTEE has DBMAINT authority over database; not defined to Teradata Database. Blank is returned.
DISPLAYDBAUTH	Whether GRANTEE can issue the DISPLAY command for the database; not defined to Teradata Database . Blank is returned.
DROPAUTH	Whether GRANTEE can drop the database; two types exist: •Authority not held •Privilege held without GRANT option Authority held with GRANT option is not defined to Teradata Database.
IMAGCOPYAUTH	Whether GRANTEE can use the COPY and MERGECOPY utilities; not defined to Teradata Database . Blank is returned.
LOADAUTH	Whether GRANTEE can use the LOAD utility; not defined to Teradata Database. Blank is returned.
REORGAUTH	Whether GRANTEE can use the REORG utility; not defined to Teradata Database. Blank is returned.
RECOVERDBAUTH	Whether GRANTEE can use the RECOVER utility; not defined to Teradata Database. Blank is returned.
REPAIRAUTH	Whether GRANTEE can use REPAIR utility; not defined to Teradata Database. Blank is returned.
STARTDBAUTH	Whether GRANTEE can issue the START command; not defined to Teradata Database. Blank is returned.
STATSAUTH	Whether GRANTEE can use the RUNSTATS utility; not defined to Teradata Database. Blank is returned.
STOPAUTH	Whether GRANTEE can issue the STOP command; not defined to Teradata Database. Blank is

	returned.
IBMREQD	Set to ‘N’, indicating that the row does not come from the basic machine-readable tape.

SYSIBM.SYSDBRM

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. This view is furnished solely for the purpose of properly supporting QMF if it needs to interrogate for a DB2 DBRM. Teradata Database does not support DBRMs, so zero rows are always returned from any query. See SYSIBM.SYSDBRM for a description of SYSIBM.SYSDBRM.

SYSIBM.SYSDBRM Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
TIMESTAMP	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
PDSNAME	CHAR(44)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(44)
PLNAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
PLCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
PRECOMPTIME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
PRECOMPDATE	CHAR(6)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(6)
QUOTE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
COMMA	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
HOSTLANG	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
CHARSET	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
MIXED	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
DEC31	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
VERSION	VARCHAR(64)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(64)

SYSIBM.SYSFIELDS

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. Teradata Database does not support FIELDPROCS, and TS/API does not emulate them. Therefore, zero rows are always retrieved from any query. This view is furnished solely for the purpose of QMF if it needs to interrogate which fields have FIELDPROCS. SYSIBM.SYSFIELDS describes SYSIBM.SYSFIELDS.

SYSIBM.SYSFIELDS Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
TBCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
TBNAME	VARCHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(8)
COLNO	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
NAME	VARCHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(8)
FLDTYPE	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
LENGTH	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
SCALE	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
FLDPROC	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
WORKAREA	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
EXITPARML	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
PARMLIST	VARCHAR(254)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(254)
EXITPARM	VARCHAR(1530)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(1530)

SYSIBM.SYSFOREIGNKEYS

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. This view is furnished solely for the purpose of properly QMF if it needs to interrogate for DB2 foreign keys. TS/API does not support foreign key definitions. SYSIBM.SYSFOREIGNKEYS describes SYSIBM.SYSFOREIGNKEYS.

SYSIBM.SYSFOREIGNKEYS Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
CREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
TBNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
RELNAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
COLNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(8)
COLNO	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
COLSEQ	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

SYSIBM.SYSINDEXES

This view joins data from DBC.DBASE, DBC.TVM, and DBC.INDEXES catalog tables to emulate the SYSIBM.SYSINDEXES table. The table contains one row for every index. SYSIBM.SYSINDEXAS describes SYSIBM.SYSINDEXES.

SYSIBM.SYSINDEXES Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	VARCHAR(18)	DBC.INDEXES	calculated from TABLEID	VARCHAR(18)
			DATABASENAME	
CREATOR	CHAR(8)	DBC.DBASE	TVMNAME	CHAR(31)
TBNAME	VARCHAR(18)	DBC.TVM	DATABASENAME	VARCHAR(18)
TBCREATOR	CHAR(8)	DBC.DBASE	calculated from UNIQUEFLAG	CHAR(31)
UNIQUERULE	CHAR(1)	DBC.INDEXES	FIELDPOSITION	CHAR(1)
COLCOUNT	SMALLINT	DBC.INDEXES	‘N’	SMALLINT
CLUSTERING	CHAR(1)		‘N’	CHAR(1)
CLUSTERED	CHAR(1)		0	CHAR(1)
DBID	SMALLINT		INDEXNUMBER	SMALLINT
ODBID	SMALLINT	DBC.INDEXES	INDEXNUMBER	SMALLINT
ISODBID	SMALLINT	DBC.INDEXES	DATABASENAME	SMALLINT
DBNAME	CHAR(8)	DBC.DBASE	’NOIXNAME’	CHAR(31)
INDEXSPACE	CHAR(8)		-1	CHAR(8)
FIRSTKEYCARD	INTEGER		-1	INTEGER
FULLKEYCARD	INTEGER		-1	INTEGER
NLEAF	INTEGER		-1	INTEGER
NLEVELS	SMALLINT		‘BP0’	SMALLINT
BPOOL	CHAR(8)		4096	CHAR(8)
PGSIZE	SMALLINT		‘Y’	SMALLINT
ERASURERULE	CHAR(1)		‘ ’	CHAR(1)
DSETPASS	CHAR(8)		‘Y’	CHAR(8)
CLOSERULE	CHAR(1)		0	CHAR(1)
SPACE	INTEGER		‘N’	INTEGER
IBMREQD	CHAR(1)		0	CHAR(1)
CLUSTERRATIO	SMALLINT		‘ ’	SMALLINT
CREATEDBY	CHAR(8)			CHAR(8)

Catalog Emulation Table/View explains how each column is supported and how its value varies from the expected DB2 value.

How Columns in SYSIBM.SYSINDEXES Are Supported

IBM Name	Description
NAME	Teradata Database does not support named indexes, so index name is derived from the TABLEID and INDEXNUMBER fields in the DBC.INDEXES table.
CREATOR	Authorization ID of the index owner. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.

TBNAME	Name of the table on which the index is defined.
TBCREATOR	Authorization ID of the table owner. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.
UNIQUERULE	States whether the index is unique.
COLCOUNT	The number of columns in the key
CLUSTERING	Set to ‘N’. Cluster is not defined in the Teradata SQL environment.
CLUSTERED	Set to ‘N’. Cluster is not defined in the Teradata SQL environment.
DBID	Internal identifier of database; set to 0
OBID	Internal identifier of index
ISOBID	Internal identifier of index space; set to OBID.
DBNAME	Name of the database containing the index. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.
INDEXSPACE	Set to ‘NOIXNAME’. Index spaces are not defined in Teradata SQL environment.
FIRSTKEYCARD	The number of distinct values of the first 8 bytes of the key; set to 1.
FULLKEYCARD	The number of distinct values of the key; set to 1.
NLEAF	The number of levels in index tree; set to 1. Index tree is not defined in the Teradata SQL environment.
NLEVELS	The number of levels in index tree; set to -1.
BPOOL	Name of buffer pool; set to ‘BP0’.
PGSIZE	Size of subpages; set to 4096.
ERASURERULE	Set to ‘Y’. Data sets considered erased when dropped.
DSETPASS	Password for the data sets of the index; set to blanks. Not defined in Teradata SQL environment.
CLOSERULE	Set to ‘Y’. Data sets considered closed when the index is not in use.
SPACE	Set to 0. DASD storage for indexes not defined in the Teradata SQL environment.
IBMREQD	Set to ‘N’, indicating that the row does not come from the basic machine-readable tape.
CLUSTERRATIO	Set to 0. Clusters are not defined in Teradata SQL environment.
CREATEDBY	Set to blanks.

SYSIBM.SYSINDEXPART

This view joins data from the DBC.DBASE, DBC.TVM, and DBC.INDEXES catalog tables to emulate the SYSIBM.SYSINDEXPART table. The table contains one row for each index. SYSIBM.SYSINDEXPART describes SYSIBM.SYSINDEXPART.

SYSIBM.SYSINDEXPART Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
PARTITION	SMALLINT		0	CHAR(31)
IXNAME	VARCHAR(18)	DBC.INDEXES	calculated from TABLEID	CHAR(31)
IXCREATOR	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
PQTY	INTEGER		0	INTEGER
SQTY	SMALLINT		0	SMALLINT
STORTYPE	CHAR(1)		‘E’	CHAR(1)
STORNAME	CHAR(8)		‘ ’	CHAR(8)
VCATNAME	CHAR(8)		‘ ’	CHAR(8)
CARD	INTEGER		-1	INTEGER
FAROFFPOS	INTEGER		-1	INTEGER
LEAFDIST	INTEGER		-1	INTEGER
NEAROFFPOS	INTEGER		0	INTEGER
IBMREQD	CHAR(1)		‘N’	CHAR(1)
LIMITKEY	VARCHAR(512)		‘0’	VARCHAR(512)
FREEPAGE	SMALLINT		0	SMALLINT
PCTFREE	SMALLINT		0	SMALLINT

Catalog Emulation Table/View explains how each column is supported and how its value varies from the expected DB2 value.

How Columns in SYSIBM.SYSINDEXPART Are Supported

IBM Name	Description
PARTITION	Index not partitioned. Set to 0. Not defined in Teradata SQL.
IXNAME	Teradata Database does not support named indexes, so the index name is derived from the TABLEID and INDEXNUMBER fields in the DBC.INDEXES table. Not defined in Teradata SQL.
IXCREATOR	Authorization ID of the index owner. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.
PQTY	Storage group. Set to 0. Not defined in Teradata SQL.
SQTY	Set to 0.
STORTYPE	Storage allocation considered explicit. Set to ‘E’.
STORNAME	Storage group. Set to blanks. Not defined in Teradata SQL.
VCATNAME	VSAM. Set to blanks. Not defined in Teradata SQL.
CARD	Statistics not gathered for the number of rows referenced by the index. Set to -1.

FAROFFPOS	Statistics not gathered for optimal positioning information in relation to index. Set to -1.
LEAFDIST	Set to -1. Not defined in Teradata SQL.
NEAROFFPOS	Set to 0. Not defined in Teradata SQL.
IBMREQD	Indicates whether a row comes from the basic machine-readable tape. Set to ‘N’.
LIMITKEY	Set to 0. Not defined in Teradata SQL.
FREEPAGE	Set to 0. Not defined in Teradata SQL.
PCTFREE	Set to 0. Not defined in Teradata SQL.

SYSIBM.SYSKEYS

This view joins data from the DBC.DBASE, DBC.INDEXES, DBC.TVM, and DBC.TVFIELDS catalog tables to emulate the SYSIBM.SYSKEYS table, which contains one row for each column of an index key. SYSIBM.SYSKEYS describes SYSIBM.SYSKEYS.

SYSIBM.SYSKEYS Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
IXNAME	VARCHAR(18)	DBC.INDEXES	calculated from TABLEID	VARCHAR(18)
IXCREATOR	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
COLNAME	VARCHAR(8)	DBC.TVFIELDS	FIELDNAME	VARCHAR(31)
COLNO	SMALLINT	DBC.INDEXES	calculated from FIELDID	SMALLINT
COLSEQ	SMALLINT	DBC.INDEXES	FIELDPOSITION	SMALLINT
ORDERING	CHAR(1)		‘A’	CHAR(1)
IBMREQD	CHAR(1)		‘N’	CHAR(1)

Catalog Emulation Table/View explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSKEYS Are Supported

IBM Name	Description
IXNAME	Teradata Database does not support named indexes, so index name is derived from the TABLEID and INDEXNUMBER fields in the DBC.INDEXES table.
IXCREATOR	Authorization ID of the index owner. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.
COLNAME	Name of the column of the key. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.
COLNO	Numerical position of column in the row.
COLSEQ	Numerical position of column in the key.
ORDERING	Set to ‘A’. All columns are ascending.

IBMREQD	Set to ‘N’, indicating that the row does not come from the basic machine-readable tape.

SYSIBM.SYSLINKS

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. This view is furnished only to support QMF if it needs to interrogate for DB2 links. Teradata Database does not support DB2 links. SYSIBM.SYSLINKS describes SYSIBM.SYSLINKS.

SYSIBM.SYSLINKS Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
CREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
TBNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
LINKNAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
PARENTNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
PARENTCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
CHILDSEQ	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
DATEGRANTED	CHAR(6)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(6)
DBNAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
DBID	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
OBID	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
COLCOUNT	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
INSERTRULE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

SYSIBM.SYSPLAN

The SYSIBM.SYSPLAN table correlates the names of Teradata Database macros emulating static SQL with the DB2 plan names that are passed from QMF. SYSIBM.SYSPLAN is built and maintained by TS/API and its installation procedures. SYSIBM.SYSPLAN describes SYSIBM.SYSPLAN. Catalog Emulation Table/View describes how columns in SYSIBM.SYSPLAN are supported.

SYSIBM.SYSPLAN Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	CHAR(8)	SYSIBM.SYSPLAN	NAME	CHAR(8)
CREATOR	CHAR(8)	SYSIBM.SYSPLAN	CREATOR	CHAR(8)
BINDDATE	CHAR(6)	SYSIBM.SYSPLAN	BINDDATE	CHAR(6)
VALIDATE	CHAR(1)	SYSIBM.SYSPLAN	VALIDATE	CHAR(1)
ISOLATION	CHAR(1)	SYSIBM.SYSPLAN	ISOLATION	CHAR(1)
VALID	CHAR(1)	SYSIBM.SYSPLAN	VALID	CHAR(1)
OPERATIVE	CHAR(1)	SYSIBM.SYSPLAN	OPERATIVE	CHAR(1)
BINDTIME	CHAR(8)	SYSIBM.SYSPLAN	BINDTIME	CHAR(8)
PLSIZE	INTEGER	SYSIBM.SYSPLAN	PLSIZE	INTEGER
IBMREQD	CHAR(1)	SYSIBM.SYSPLAN	IBMREQD	CHAR(1)
AVGSIZE	INTEGER	SYSIBM.SYSPLAN	AVGSIZE	INTEGER

ACQUIRE	CHAR(1)	SYSIBM.SYSPLAN	ACQUIRE	CHAR(1)
RELEASE	CHAR(1)	SYSIBM.SYSPLAN	RELEASE	CHAR(1)
filler	CHAR(1)	SYSIBM.SYSPLAN	filler	CHAR(1)
filler	CHAR(1)	SYSIBM.SYSPLAN	filler	CHAR(1)
filler	CHAR(1)	SYSIBM.SYSPLAN	filler	CHAR(1)
EXPLAN	CHAR(1)	SYSIBM.SYSPLAN	EXPLAN	CHAR(1)
EXPREDICATE	CHAR(1)	SYSIBM.SYSPLAN	EXPREDICATE	CHAR(1)
BOUNDBY	CHAR(8)	SYSIBM.SYSPLAN	BOUNDBY	CHAR(8)
QUALIFIER	CHAR(8)	SYSIBM.SYSPLAN	QUALIFIER	CHAR(8)
CACHESIZE	SMALLINT	SYSIBM.SYSPLAN	CACHESIZE	SMALLINT
PLENTRIES	SMALLINT	SYSIBM.SYSPLAN	PLENTRIES	SMALLINT
DEFERPREP	CHAR(1)	SYSIBM.SYSPLAN	DEFERPREP	CHAR(1)
CURRENTSERVER	CHAR(16)	SYSIBM.SYSPLAN	CURRENTSERVER	CHAR(16)
SYSENTRIES	SMALLINT	SYSIBM.SYSPLAN	SYSENTRIES	SMALLINT

Notice: Any user modification of SYSIBM.SYSPLAN may invalidate the integrity of TS/API.

How Columns in SYSIBM.SYSPLAN are Supported

IBM Name	Description
NAME	The plan name that is passed from an application program at execution time.
CREATOR	The userid under which the plan macros are stored. This is currently limited to eight characters, since that is the size of the authorization ID passed from a DB2 application.

The remaining columns in SYSIBM.SYSPLAN are set to default values and not used by TS/ API.

SYSIBM.SYSPLANAUTH

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. Teradata Database does not support user privileges on application plans. SYSIBM.SYSPLANAUTH describes SYSIBM.SYSPLANAUTH.

SYSIBM.SYSPLANAUTH Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
GRANTOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
GRANTEE	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
NAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
TIMESTAMP	CHAR(12)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(12)
DATEGRANTED	CHAR(6)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(6)
TIMEGRANTED	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
GRANTEETYPE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
AUTHHOWGOT	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
BINDAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
EXECUTEAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

SYSIBM.SYSPLANDEP

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. Teradata Database does not support, nor does TS/API emulate, a referential integrity relationship between plans and the tables, views, or indexes contained in them. SYSIBM.SYSPLANDEP describes SYSIBM.SYSPLANDEP.

SYSIBM.SYSPLANDEP Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
BNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
BCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
BTYPE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
DNAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

SYSIBM.SYSRELS

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. Teradata Database does not support relationships between tables. SYSIBM.SYSRELS describes SYSIBM.SYSRELS.

SYSIBM.SYSRELS Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
CREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
TBNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
RELNAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
REFTNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
REFTBCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
COLCOUNT	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
DELETERULE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
RELOBID1	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
RELOBID2	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
TIMESTAMP	CHAR(12)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(12)

SYSIBM.SYSRESAUTH

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. Teradata Database does not support privileges on buffer pools, storage groups, or table spaces. SYSIBM.SYSRESAUTH describes SYSIBM.SYSRESAUTH.

SYSIBM.SYSRESAUTH Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
GRANTOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)

GRANTEE	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
QUALIFIER	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
NAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
GRANTEETYPE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
AUTHHOWGOT	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
OBTYPE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
TIMESTAMP	CHAR(12)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(12)
DATEGRANTED	CHAR(6)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(6)
TIMEGRANTED	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
USEAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

SYSIBM.SYSSTMT

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. Teradata Database does not support the SYSIBM.SYSSTMT table. SYSIBM.SYSSTMT describes SYSIBM.SYSSTMT.

SYSIBM.SYSSTMT Description

IBM Name	**IBM Type**	**Teradata Database Table**	**Teradata Database Name**	**Teradata Database Type**
NAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
PLNAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
PLCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
SEQNO	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
STMTNO	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
SECTNO	SMALLINT	SYSAPI.SYSDUMMY	DUMMYNUM	SMALLINT
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
TEXT	VARCHAR(254)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(254)

SYSIBM.SYSSTOGROUP

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. Teradata Database does not support storage groups; therefore, zero rows are always returned from any query of this table. SYSIBM.SYSTOGROUP describes SYSIBM.SYSSTOGROUP.

SYSIBM.SYSSTOGROUP Description

IBM Name	**IBM Type**	**Teradata Database Table**	**Teradata Database Name**	**Teradata Database Type**
NAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
CREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
VCATNAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
VPASSWORD	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
SPACE	INTEGER	SYSAPI.SYSDUMMY	DUMMYNUM	INTEGER
SPCDATE	CHAR(5)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(5)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

CREATEDBY	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)

SYSIBM.SYSSYNONYMS

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. This view is furnished solely for the purpose of properly supporting QMF if it needs to interrogate for DB2 synonyms. Teradata Database does not support synonyms, so zero rows are always returned from any query. SYSIBM.SYSSYNONYMS describes SYSIBM.SYSSYNONYMS.

SYSIBM.SYSSYNONYMS Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
CREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
TBNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
TBCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

SYSIBM.SYSTABAUTH

This view joins data from the DBC.ACCESSRIGHTS, DBC.TVM, and DBC.DBASE catalog tables to emulate the SYSIBM.SYSTABAUTH table. SYSIBM.SYSTABAUTH records user privileges on tables and views. SYSIBM.SYSTABAUTH describes SYSIBM.SYSTABAUTH.

SYSIBM.SYSTABAUTH Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
GRANTOR	CHAR(8)	DBC.ACCESSRIGHTS	GRANTORNAME	CHAR(31)
GRANTEE	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
GRANTEETYPE	CHAR(1)		‘ ’	CHAR(1)
DBNAME	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
SCREATOR	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
STNAME	CHAR(18)	DBC.TVM	TVMNAME	CHAR(31)
TCREATOR	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
TTNAME	CHAR(18)	DBC.TVM	TVMNAME	CHAR(31)
AUTHHOWGOT	CHAR(1)		‘ ’	CHAR(1)
TIMESTAMP	CHAR(12)		‘?’	CHAR(12)
DATEGRANTED	CHAR(6)		‘?’	CHAR(6)
TIMEGRANTED	CHAR(8)		‘?’	CHAR(8)
UPDATECOLS	CHAR(1)		‘ ’	CHAR(1)
ALTERAUTH	CHAR(1)	DBC.ACCESSRIGHTS	calculated from ACCESSRIGHT	CHAR(1)
DELETEAUTH	CHAR(1)	DBC.ACCESSRIGHTS	calculated from ACCESSRIGHT	CHAR(1)
INDEXAUTH	CHAR(1)	DBC.ACCESSRIGHTS	calculated from ACCESSRIGHT	CHAR(1)
INSERTAUTH	CHAR(1)	DBC.ACCESSRIGHTS	calculated from ACCESSRIGHT	CHAR(1)
SELECTAUTH	CHAR(1)	DBC.ACCESSRIGHTS	calculated from ACCESSRIGHT	CHAR(1)

UPDATEAUTH	CHAR(1)	DBC.ACCESSRIGHTS	calculated from ACCESSRIGHT ‘N’ ‘ ’	CHAR(1)
IBMREQD	CHAR(1)		‘’	CHAR(1)
GRANTEELOCATION	CHAR(16)		‘ ’	CHAR(16)
LOCATION	CHAR(16)		‘’	CHAR(16)
COLLID	CHAR(18)		‘’	CHAR(18)
COLTOKEN	CHAR(8)			CHAR(8)
CAPTUREAUTH	CHAR(1)			CHAR(1)

Catalog Emulation Table/View describes how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSTABAUTH Are Supported

IBM Name	Description
GRANTOR	The Teradata Database user who granted the access right. The DB2 length of eight is expanded to 31 to accommodate the longer userid names available in Teradata SQL.
GRANTEE	The Teradata Database user who was granted the access right. The DB2 length of eight is expanded to 31 to accommodate the longer userid names available in Teradata SQL.
GRANTEETYPE	Set to blank, reflecting that the grantee is a user. Teradata Database cannot grant access rights to programs, so only blank is valid.
DBNAME	Name of the database on which GRANTOR has authority. The DB2 length of eight is expanded to 31 to accommodate the longer database names available in Teradata SQL.
SCREATOR	The Teradata Database userid who created the table or view on which rights have been granted. The DB2 length of eight is expanded to 31 to accommodate the longer userid names available in Teradata SQL.
STNAME	The name of the table or view on which rights have been granted. The DB2 length of eight is expanded to 31 to accommodate the longer table names available in Teradata SQL.
TCREATOR	The same value as SCREATOR. This value is not valid for views because Teradata Database provides information only about the rights to the view itself, not about underlying tables. Like SCREATOR, this column is expanded to 31 characters.
TTNAME	The same value as STNAME. This value is not valid for views because Teradata Database provides information only about the rights to the view itself, not about underlying tables. Like STNAME, this column is expanded to 31 characters.
AUTHHOWGOT	Set to blank. Teradata Database does not support authorization level.
TIMESTAMP	Set to ‘?’, indicating that no timestamp is available. This column is generally not used by users or application programs.
DATEGRANTED	Set to ‘?’, indicating that no grant date is available. This column is generally not used by users or application programs.
TIMEGRANTED	Set to ‘?’, indicating that no time granted is available. This column is generally not used by users or

	application programs.
UPDATECOLS	Set to blank, indicating that any update privileges apply equally to all columns in a table.
ALTERAUTH	Whether GRANTEE can alter table: •Blank - no privilege •Y - privilege
DELETEAUTH	Whether GRANTEE can delete rows from the table or view: •blank - no privilege •Y - privilege
INDEXAUTH	Whether GRANTEE can create indexes on the table: •blank - no privilege •Y - privilege
INSERTAUTH	Whether GRANTEE can insert rows into a table or view: •blank - no privilege •Y - privilege
SELECTAUTH	Whether GRANTEE can select rows from a table or view: •blank - no privilege •Y - privilege
UPDATEAUTH	Whether GRANTEE can update rows in a table or view: •blank - no privilege •Y - privilege
IBMREQD	Set to 'N', indicating that the row does not come from the basic machine-readable tape.
GRANTEELOCATION	Not used.
LOCATION	If the GRANTEE is package-id, the location name.
COLLID	If the GRANTEE is package-id, the collection name.
CONTOKEN	If the GRANTEE is package-id, the consistency token.
CAPTUREAUTH	Not used.

SYSIBM.SYSTABLEPART

This table emulates the SYSIBM.SYSTABLEPART table. The table contains one row for the DSNDB04 table space. SYSIBM.SYSTABLEPART describes SYSIBM.SYSTABLEPART.

SYSIBM.SYSTABLEPART Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
PARTITION	SMALLINT		0	SMALLINT
TSNAME	CHAR(8)		‘DSNDB04’	CHAR(8)
DBNAME	CHAR(8)		‘ ’	CHAR(8)
IXNAME	VARCHAR(18)		‘ ’	VARCHAR(18)
IXCREATOR	CHAR(8)		‘ ’	CHAR(8)
PQTY	INTEGER		0	INTEGER
SQTY	SMALLINT		0	SMALLINT
STORTYPE	CHAR(1)		‘E’	CHAR(1)
STORNAME	CHAR(8)		‘ ’	CHAR(8)
VCATNAME	CHAR(8)		‘ ’	CHAR(8)
CARD	INTEGER		-1	INTEGER
FARINDREF	INTEGER		-1	INTEGER
NEARINDREF	INTEGER		-1	INTEGER
PERCACTIVE	SMALLINT		-1	SMALLINT
PERCDROP	SMALLINT		-1	SMALLINT
IBMREQD	CHAR(1)		‘N’	CHAR(1)
LIMITKEY	VARCHAR(512)		‘0’	VARCHAR(512)
FREEPAGE	SMALLINT		0	SMALLINT
PCTFREE	SMALLINT		0	SMALLINT
CHECKFLAG	CHAR(1)		‘ ’	CHAR(1)
CHECKRID	CHAR(4)		‘ ’	CHAR(4)

Catalog Emulation Table/View explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSTABLEPART Are Supported

IBM Name	Description	Setting
PARTITION	Table not partitioned	0
TSNAME	Table space name	‘DSNDB04’
DBNAME	Database name	Blanks
IXNAME	Partitioned indexes not defined under Teradata SQL environment	Blanks
IXCREATOR	Owner of the index	Blanks
PQTY	Storage groups not used	0
SQTY	Storage groups not used	0

STORTYPE	Storage allocation explicit. Storage groups not used	‘E’
STORNAME		Blanks
VCATNAME	VSAM not supported in Teradata SQL environment	Blanks
CARD	Setting indicates statistics not gathered	-1
FARINDEF	Setting indicates statistics not gathered	-1
NEARINDREF	Setting indicates statistics not gathered	-1
PERCACTIVE	Setting indicates statistics not gathered	-1
PERCDROP	Setting indicates statistics not gathered	-1
IBMREQD	Whether the row comes from the basic machine-readable tape	N
LIMITKEY	Table space not defined	0
FREEPAGE	Pages left as free space not defined under Teradata SQL environment	0
PCTFREE	No page left as free space	0
CHECKFLAG	Table not partitioned	Blank
CHECKRID	Table not partitioned	Blanks

SYSIBM.SYSTABLES

This view joins data from the DBC.DBASE and DBC.TVM catalog tables to emulate the SYSIBM.SYSTABLES table. Each row defines one table or view stored in the Teradata Database system catalog. SYSIBM.SYSTABLES describes SYSIBM.SYSTABLES.

SYSIBM.SYSTABLES Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	VARCHAR (18)	DBC.TVM	TVMNAME	VARCHAR(31)
CREATOR	CHAR(8)	DBC.DBASE	DATABASENAME	CHAR(31)
TYPE	CHAR(1)	DBC.TVM	TABLEKIND	CHAR(1)
DBNAME	CHAR(8)	DBC.TVM	TVMNAME	VARCHAR(31)
DBID	SMALLINT		0	SMALLINT
OBID	SMALLINT		0	SMALLINT
COLCOUNT	SMALLINT	DBC.TVM	calculate from NEXT- FIELDID	SMALLINT
EDPROC	CHAR(8)		‘ ’	CHAR(8)
VALPROC	CHAR(8)		‘ ’	CHAR(8)

CLUSTERTYPE	CHAR(1)		‘ ’	CHAR(1)
CLUSTERID	INTEGER		0	INTEGER
CARD	INTEGER		-1	INTEGER
NPAGES	INTEGER		-1	INTEGER
PCTPAGES	SMALLINT		-1	SMALLINT
IBMREQD	CHAR(1)		‘N’	CHAR(1)
REMARKS	VARCHAR(254)	DBC.TVM	COMMENTSTRING	VARCHAR(254)
PARENTS	SMALLINT		0	SMALLINT
CHILDREN	SMALLINT		0	SMALLINT
KEYCOLUMNS	SMALLINT		0	SMALLINT
RECLENGTH	SMALLINT		32000	SMALLINT
STATUS	CHAR(1)		‘ ’	CHAR(1)
KEYOBID	SMALLINT		0	SMALLINT
LABEL	VARCHAR(30)	DBC.TVM	COMMENTSTRING	VARCHAR(254)
CHECKFLAG	CHAR(1)		‘ ’	CHAR(1)
CHECRID	CHAR(4)		‘ ’	CHAR(4)
AUDITING	CHAR(1)		‘ ’	CHAR(1)
CREATEDBY	CHAR(8)		‘SYSIBM’	CHAR(8)
LOCATION	CHAR(16)		‘ ’	CHAR(16)
TBCREATOR	CHAR(8)		‘ ’	CHAR(8)
TBNAME	VARCHAR(18)		‘ ’	VARCHAR(18)
CREATEDTS	CHAR(12)		‘ ’	CHAR(12)
ALTEREDTS	CHAR(12)		‘ ’	CHAR(12)
DATACAPTURE	CHAR(1)		‘ ’	CHAR(1)
RBA1	CHAR(6)		‘X000000000000’	CHAR(6)
RBA2	CHAR(6)		‘X000000000000’	CHAR(6)

Catalog Emulation Table/View explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSTABLES Are Supported

IBM Name	Description
NAME	The name of a table or view in Teradata Database. The DB2 length of 18 characters is expanded to a length of 31 to allow for longer table names on Teradata Database

CREATOR	The name of the user who created the table or view. The DB2 length of 8 characters is expanded to a length of 31 to allow for longer userids on Teradata Database.
TYPE	Indicates whether this description applies to a table (T) or a view (V).
DBNAME	The name of the user who created the table or view. This column is identical to CREATOR. The DB2 length of 8 characters is expanded to a length of 31 to allow for longer userids on Teradata Database.
TSNAME	Teradata Database does not support the concept of table spaces. The table space name reflected to the application is the same as the table name.
DBID	Set to zero, indicating that this is a DB2 view. This indication may generate erroneous results if an application depends on the value stored in this column. A valid database ID is not available in Teradata Database for this column.
OBID	Set to zero, indicating that this is a DB2 view. This indication may generate erroneous results if an application depends on the value stored in this column. A valid table ID is not available in Teradata Database for this column.
COLCOUNT	Number of columns in a table or view.
EDPROC	Set to blank, indicating that an edit procedure does not exist for this table. Teradata Database does not support edit procedures.
VALPROC	Set to blank, indicating that a validation procedure does not exist for this table. Teradata Database does not support validation procedures.
CLUSTERTYPE	Set to blank and not used in DB2 or in Teradata Database.
CLUSTERRID	Set to zero and not used in DB2 or in Teradata Database.
CARD	Set to -1, indicating that statistics have not been collected on this table.
NPAGES	Set to -1, indicating that statistics have not been collected on this table.
PCTPAGES	Set to -1, indicating that statistics have not been collected on this table.
IBMREQD	Set to ‘N’, indicating that the row does not come from the basic machine-readable tape.
REMARKS	Contains the table comment.
PARENTS	Set to zero. Parent relationship information is not available on Teradata Database.
CHILDREN	Set to zero. Child relationship information is not available on Teradata Database.
KEYCOLUMNS	Set to zero, indicating that a DB2 primary key does not exist.
RECLENGTH	Set to 32000, the largest possible record length that can be returned from tTeradata Database. RECLENGTH does not reflect the actual record length. If an application uses this information to allocate buffer space, excess memory may be used. If the application depends on this value to be correct, the application may not function properly.
STATUS	Set to blank, indicating that a primary key does not exist.
KEYOBID	Set to zero, indicating that a link to a primary key or index does not exist.

LABEL	The same as the REMARKS column because Teradata Database stores only one comment per table. The DB2 length of 30 characters is expanded to a length of 254 because the table comment for Teradata Database is much longer than the label allowed in DB2.
CHECKFLAG	Set to blanks. The table does not contain rows that violate referential constraints.
CHECKRID	Set to blanks. The table is not in a check pending state.
AUDITING	Set to blanks. Audit none.
CREATEDBY	Set to 'SYSIBM'.
LOCATION	Set to blanks. Not used.
TBCREATOR	Set to blanks. Not used.
TBNAME	Set to blanks. Not used.
CREATEDTS	The timestamp of the CREATE for table, view, alias
ALTERDTS	The timestamp of ALTER for tables
DATACAPTURE	The value of the DATA CAPTURE option for tables (Y or blank)
RBA1	The log RBA when the tables were created
RBA2	The log RBA when the tables were last altered

SYSIBM.SYSTABLESPACE

This view emulates the SYSIBM.SYSTABLESPACE table. The table contains a row for the DSNDB04 table space. SYSIBM.SYSTABLESPACE describes SYSIBM.SYSTABLESPACE.

SYSIBM.SYSTABLESPACE Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	CHAR(8)		'DSNDB04'	CHAR(8)
CREATOR	CHAR(8)		'SYSIBM'	CHAR(8)
DBNAME	CHAR(8)		'DSNDB04'	CHAR(8)
DBID	SMALLINT		4	SMALLINT
OBID	SMALLINT		4	SMALLINT
PSID	SMALLINT		4	SMALLINT
BPOOL	CHAR(8)		'BP0'	CHAR(8)
PARTITIONS	SMALLINT		0	SMALLINT
LOCRULE	CHAR(1)		'A'	CHAR(1)
PGSIZE	SMALLINT		4	SMALLINT
ERASERULE	CHAR(1)		'N'	CHAR(1)
STATUS	CHAR(1)		'A'	CHAR(1)
IMPLICIT	CHAR(1)		'Y'	CHAR(1)
NTABLES	SMALLINT		32767	SMALLINT
NACTIVE	INTEGER		0	INTEGER

DSETPASS	CHAR(8)		‘ ’	CHAR(8)
CLOSERULE	CHAR(1)		‘Y’	CHAR(1)
SPACE	INTEGER		‘0’	INTEGER
IBMREQD	CHAR(1)		‘N’	CHAR(1)
ROOTNAME	VARCHAR(18)		‘ ’	VARCHAR(18)
ROOTCREATOR	CHAR(8)		‘ ’	CHAR(8)
SEGSIZE	SMALLINT		0	SMALLINT
CREATEDBY	CHAR(8)		‘SYSIBM’	CHAR(8)

Catalog Emulation Table/View explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSTABLESPACE Are Supported

IBM Name	Description	Setting
NAME	Table space name	‘DSNDB04’
CREATOR	Owner’s authorization ID	‘SYSIBM’
DBNAME	Database name	‘DSNDB04’
DBID	Database’s internal identifier	4
OBID	Table space file’s internal identifier	4
PSID	Table space’s internal identifier	4
BPOOL	Name of buffer pool used for table space	‘BP0’
PARTITIONS	Table space not partitioned	0
LOCKRULE	Lock size is any	‘A’
PGSIZE	Page size in kilobytes	4
ERASERULE	Erasure of data sets	N
STATUS	Table space is available	A
IMPLICIT	Table space is created implicitly	Y
NTABLES	Number of tables defined in the table space	32767
NACTIVE	Statistics not gathered	0
DSETPASS	Not supported in the Teradata SQL environment	Blanks
CLOSERULE	Are data sets closed when table space is not used?	Y
SPACE	Is there DASD storage for table space?	0
IBMREQD	Does the row come from the basic machine-readable tape?	N
ROOTNAME	Not a structured table space	Blanks

ROOTCREATOR	Root table does not exist	Blanks
SEGSIZE	Table space not segmented	0
CREATEDBY	Primary authorization ID of user who created the table space	‘SYSIBM’

SYSIBM.SYSUSERAUTH

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. This view is furnished only to support QMF if it needs to interrogate for DB2 system privileges. Teradata Database does not support system privileges held by users, so zero rows are always returned from any query. SYSIBM.SYSUSERAUTH describes SYSIBM.SYSUSERAUTH.

SYSIBM.SYSUSERAUTH Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
GRANTOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
GRANTEE	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
TIMESTAMP	CHAR(12)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(12)
DATEGRANTED	CHAR(6)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(6)
TIMEGRANTED	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
GRANTEETYPE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
AUTHHOWGOT	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
ALTERBPAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
BINDADDAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
BSDSAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
CREATEDBAAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
CREATEDBCAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
CREATESGAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
DISPLAYAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
RECOVERAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
STOPALLAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
STOSPACEAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
SYSADMAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
SYSOPRAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
MON1AUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
MON2AUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
CREATEALIASAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
SYSCTRLAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
BINDAGENTAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
ARCHIVEAUTH	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
FILLER1	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
FILLER2	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

SYSIBM.SYSVIEWDEP

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. This view is furnished only to support QMF if it needs to interrogate for the dependencies of views on tables and other views. Teradata SQL does not support this table, so zero rows are always returned from any query. SYSIBM.SYSVIEWDEP describes SYSIBM.SYSVIEWDEP.

SYSIBM.SYSVIEWDEP Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
BNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
BCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
BTYPE	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
DNAME	VARCHAR(18)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(18)
DCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

SYSIBM.SYSVIEWS

This view contains data from the DBC.TVM catalog table to emulate the SYSIBM.SYSVIEWS table. The table contains one or more rows for each view. SYSIBM.SYSVIEWS describes SYSIBM.SYSVIEWS.

SYSIBM.SYSVIEWS Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	VARCHAR(8)	DBC.TVM	TVMNAME	VARCHAR(31)
CREATOR	CHAR(8)	DBC.TVM	CREATORNAME	CHAR(31)
SEQNO	SMALLINT		1	SMALLINT
CHECK	CHAR(1)		‘N’	CHAR(1)
IBMREQD	CHAR(1)		‘N’	CHAR(1)
TEXT	VARCHAR(254)		‘?’	VARCHAR(254)

Catalog Emulation Table/View explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSVIEWS Are Supported

IBM Name	Description
NAME	View name. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.
CREATOR	View owner’s authorization ID. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.
SEQNO	Sequence number of row; set to 1.

CHECK	Set to ‘N’. Indicates CHECK option of the DB2 CREATE VIEW statement not used
IBMREQD	Set to ‘N’, indicating the row does not come from the basic machine-readable tape.
TEXT	Set to ‘?’. Text portion not supported by the Teradata SQL environment.

SYSIBM.SYSVLTREE

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. This view is furnished only to support QMF if it needs to interrogate SYSIBM.SYSVLTREE. Teradata Database does not support parse trees, so zero rows are always returned from any query. SYSIBM.SYSVTREE describes SYSIBM.SYSVLTREE.

SYSIBM.SYSVLTREE Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)
VTREE	VARCHAR(4000)	SYSAPI.SYSDUMMY	DUMMYCHR	VARCHAR(4000)

SYSIBM.SYSVOLUMES

This is a view of the empty table, SYSAPI.SYSDUMMY. The columns are described exactly as in DB2. This view is furnished solely to properly support QMF if it needs to interrogate for storage group volumes. Teradata Database does not support storage groups, so zero rows are always returned from any query. SYSIBM.SYSVOLUMES describes SYSIBM.SYSVOLUMES.

SYSIBM.SYSVOLUMES Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
SGNAME	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
SGCREATOR	CHAR(8)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(8)
VOLID	CHAR(6)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(6)
IBMREQD	CHAR(1)	SYSAPI.SYSDUMMY	DUMMYCHR	CHAR(1)

SYSIBM.SYSVTREE

This view contains data from the DBC.TVM catalog table to emulate the SYSIBM.SYSVTREE table. For each view, the table contains one or more rows, with the parse tree of the view. SYSIBM.SYSVTREE describes SYSIBM.SYSVTREE.

SYSIBM.SYSVTREE Description

IBM Name	IBM Type	Teradata Database Table	Teradata Database Name	Teradata Database Type
NAME	VARCHAR(18)	DBC.TVM	TVMNAME	VARCHAR(31)

SYSIBM.SYSVTREE Description

CREATOR	CHAR(8)	DBC.TVM	CREATORNAME	CHAR(31)
TOTLEN	INTEGER		0	INTEGER
IBMREQD	CHAR(1)		‘N’	CHAR(1)
VTREE	VARCHAR(4000)		‘ ’	VARCHAR(4000)

Catalog Emulation Table/View explains how each column is supported and how its value may vary from the expected DB2 value.

How Columns in SYSIBM.SYSVTREE Are Supported

IBM Name	Description
NAME	View name. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.
CREATOR	View owner’s authorization ID. Field length of eight is expanded to 31 to accommodate the longer names in Teradata SQL.
TOTLEN	Total length of parse tree; set to 0. Parse trees are not defined in the Teradata SQL environment.
IBMREQD	Set to ‘N’, indicating that the row does not come from the basic machine-readable tape.
VTREE	Set to blanks. Parse trees are not defined in the Teradata SQL environment.

Glossary

A

administrator

A special user responsible for allocating Teradata Database resources to a community of users.

application program

A program that performs a particular function or set of functions that the user desires to perform.

B

bind

Process by which the output from a precompiler is converted to a usable control structure called a plan. Access paths to data are selected and some authorization checking is performed.

C

Call Attach Facility

Software that allows application programs to connect to and use either the DB2 or *Teradata Database*.

channel

The means by which a central processor is attached to peripheral units; the path by which data is transferred between the mainframe host and the Teradata Database hardware platform.

client

A system that can execute application programs that access and manipulate data in Teradata Database.

Customer Information Control System

(CICS) An IBM monitor program for application programs that are optimized for real-time user interaction. CICS runs under the MVS operating system.

cursor

The mechanism in SQL that moves through a multi-row response to a SELECT or other data-returning statement. The cursor can be considered as pointing to a current row of data.

cursor isolation

One of two levels of locking for a cursor. Cursor isolation levels are repeatable read and cursor stability.

cursor stability

A level of cursor isolation used by DB2 and SQL/DS that ensures that a transaction acquires a read lock on data when it obtains addressability to the data. The read lock is relinquished when the transaction relinquishes its addressability to the data if the transaction has not performed any updates or deletions. If the transaction performs any updates or deletions, the read lock is automatically upgraded to a write lock and relinquished at end-of-transaction time.

D

database

In Teradata SQL, a related set of tables that share a common space allocation and owner.

database computer

See database computing system.

database computing system

A complete hardware/software system that provides all of the functions of a traditional database management system and more: a non-procedural, user-friendly query language; fault-tolerant operation with no single point of failure; multiuser access; and interactive and batch environments.

relational database management system (RDBMS)

Computer procedures that permit the database to be maintained independently of application programs. A database management system provides services for data definition, data manipulation, and data integrity.

database server

A hardware/software system that processes requests from users (clients) of a relational database management system.

data integrity

Data preserved in its whole state without accidental or intentional destruction or modification.

RDBMS

See relational database management system.

DBRM

Database request module. A data set member created by the DB2 precompiler that contains information regarding SQL statements. DBRMs are used in the bind process.

Data Base 2

(DB2) IBM's relational database management system running under MVS.

DB2 Plan

A usable control structure containing access paths to data and some access authorization that is derived from output from a precompiler.

directive

A TS/API command.

dynamic SQL

SQL statements that are prepared and executed while the application program is running. The SQL source is contained in client language variables rather than being coded in the application program.

E

embedded SQL

All SQL statements that are contained in the application program. Embedded SQL can be either static or dynamic.

L

local area network (LAN)

A means of connecting workstations that allows them to communicate with one another. The LAN is usually confined to a limited area, such as a building.M

M

microprocessor

A computer with miniaturized elements.

Multiple Virtual Storage (MVS)

One of the primary operating systems (or system control programs) for medium and large IBM computers.

P

parallel processing

The division of a database request into two or more components and the processing of each component separately and simultaneously.

pass-thru facility

The capability of an application (for example, SAS and QMF) to accept SQL and pass it through for processing.

program

A unit of software that performs a set of operations to satisfy the needs of users or other programs. A program consists of one or more modules.

Q

query

A request from a database to retrieve, modify, or delete data.

Query Management Facility (QMF)

IBM's online relational database query and reporting system that can access both IBM's DB2 and SQL/DS databases.

R

redundancy

The presence of more than one component to perform a required function. The presence of more than one copy of a block of data to provide fallback in case of data loss.

repeatable read

A level of cursor isolation used by DB2 and TS/API that ensures that a transaction acquires a read lock on data when it obtains addressability to the data. The read lock is not relinquished when the transaction relinquishes its addressability to the data, whether or not the transaction has performed any updates or deletions. If the transaction performs any updates or deletions, the read lock is automatically upgraded to a write lock. Isolation level RR ensures that both read and write locks are not relinquished until end-of- transaction time.

request

A message sent from an application program to Teradata Database. Also, an application program's petition for action from the Teradata Database, and a response from Teradata Database as a result of that petition.

S

SQLCODE

The field in the SQLCA that contains a return code following completion of a database request.

SQLERRM

The field in the SQLCA that contains error message text related to the corresponding SQLCODE following completion of a database request.

Static SQL

SQL statements that are coded within an application program and prepared via a precompiler before program execution. The precompiler replaces the SQL statements with statements recognizable by the client language compiler. For DB2, output from the precompiler includes source code (which is submitted to the compiler) and a database request module (DBRM) that is input to the bind process.

T

Teradata SQL

The Teradata Structured Query Language (Teradata SQL) for use with the Teradata Database.

Teradata SQL Extensions

SQL syntax capabilities that extend beyond normal ANSI and DB2 syntax and that are supported only on the Teradata Database.

Time Sharing Option (TSO)

A multi-user monitor subsystem that runs under the MVS operating system.

Transparency Series/Application Program Interface (TS/API)

An application program interface that allows you to access relational databases stored onTeradata Database.

U

unit of work

A sequence of operations within an application process which is recoverable.

updatable cursors

A cursor used to modify (update or delete) the current row in a multi- row response.

V

Virtual Machine (VM)

One of the primary operating systems (or system control programs) for medium and large IBM computers.

Index