What would you do if you knew?™

Teradata Database

Introduction to Teradata

Release 16.00

B035-1091-160K

December 2016

The product or products described in this book are licensed products of Teradata Corporation or its affiliates.

Teradata, Applications-Within, Aster, BYNET, Claraview, DecisionCast, Gridscale, MyCommerce, QueryGrid, SQL-MapReduce, Teradata Decision Experts, "Teradata Labs" logo, Teradata ServiceConnect, Teradata Source Experts, WebAnalyst, and Xkoto are trademarks or registered trademarks of Teradata Corporation or its affiliates in the United States and other countries.
Adaptec and SCSISelect are trademarks or registered trademarks of Adaptec, Inc.
Amazon Web Services, AWS, [any other AWS Marks used in such materials] are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.
AMD Opteron and Opteron are trademarks of Advanced Micro Devices, Inc.
Apache, Apache Avro, Apache Hadoop, Apache Hive, Hadoop, and the yellow elephant logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries.
Apple, Mac, and OS X all are registered trademarks of Apple Inc.
Axeda is a registered trademark of Axeda Corporation. Axeda Agents, Axeda Applications, Axeda Policy Manager, Axeda Enterprise, Axeda Access, Axeda Software Management, Axeda Service, Axeda ServiceLink, and Firewall-Friendly are trademarks and Maximum Results and Maximum Support are servicemarks of Axeda Corporation.
CENTOS is a trademark of Red Hat, Inc., registered in the U.S. and other countries.
Cloudera, CDH, [any other Cloudera Marks used in such materials] are trademarks or registered trademarks of Cloudera Inc. in the United States, and in jurisdictions throughout the world.
Data Domain, EMC, PowerPath, SRDF, and Symmetrix are registered trademarks of EMC Corporation.
GoldenGate is a trademark of Oracle.
Hewlett-Packard and HP are registered trademarks of Hewlett-Packard Company.
Hortonworks, the Hortonworks logo and other Hortonworks trademarks are trademarks of Hortonworks Inc. in the United States and other countries.
Intel, Pentium, and XEON are registered trademarks of Intel Corporation.
IBM, CICS, RACF, Tivoli, and z/OS are registered trademarks of International Business Machines Corporation.
Linux is a registered trademark of Linus Torvalds.
LSI is a registered trademark of LSI Corporation.
Microsoft, Active Directory, Windows, Windows NT, and Windows Server are registered trademarks of Microsoft Corporation in the United States and other countries.
NetVault is a trademark or registered trademark of Dell Inc. in the United States and/or other countries.
Novell and SUSE are registered trademarks of Novell, Inc., in the United States and other countries.
Oracle, Java, and Solaris are registered trademarks of Oracle and/or its affiliates.
QLogic and SANbox are trademarks or registered trademarks of QLogic Corporation.
Quantum and the Quantum logo are trademarks of Quantum Corporation, registered in the U.S.A. and other countries.
Red Hat is a trademark of Red Hat, Inc., registered in the U.S. and other countries. Used under license.
SAP is the trademark or registered trademark of SAP AG in Germany and in several other countries.
SAS and SAS/C are trademarks or registered trademarks of SAS Institute Inc.
Simba, the Simba logo, SimbaEngine, SimbaEngine C/S, SimbaExpress and SimbaLib are registered trademarks of Simba Technologies Inc.
SPARC is a registered trademark of SPARC International, Inc.
Symantec, NetBackup, and VERITAS are trademarks or registered trademarks of Symantec Corporation or its affiliates in the United States and other countries.
Unicode is a registered trademark of Unicode, Inc. in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Other product and company names mentioned herein may be the trademarks of their respective owners.

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 e-mail: 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.

Copyright © 2000 - 2016 by Teradata. All Rights Reserved.

Preface

Purpose

This book provides an introduction to Teradata covering the following broad topics:

- The data warehouse and active Teradata
- The relational model and Teradata Database architecture
- Teradata Database hardware and software architecture
- Teradata Database RASUI (reliability, availability, serviceability, usability, and installability)
- Communication between the client and Teradata Database
- Data definitions and data manipulation using SQL
- SQL application development
- Data distribution and data access methods
- Concurrent control and transaction recovery
- The Data Dictionary
- International character support
- Query and database analysis tools
- Database security and system administration
- Managing and monitoring Teradata Database

Audience

This book is intended for users interested in a broad overview of Teradata. Such individuals may include database users or administrators.

Supported Software Releases and Operating Systems

This book supports Teradata® Database 16.00.

Teradata Database 16.00 is supported on:

- SUSE Linux Enterprise Server (SLES) 11 SP1
- SUSE Linux Enterprise Server (SLES) 11 SP3

Teradata Database client applications support other operating systems.

Prerequisites

To gain an understanding of Teradata, you should be familiar with the following:

- Basic computer and relational database technology
- System hardware

Changes to This Book

Release	Description
Teradata Database 16.00 December 2016	Added the DATASET data type. Removed references to SLES 10, Priority Scheduler, and Meta Data Services, which are not supported in this release.

Product Safety Information

This document may contain information addressing product safety practices related to data or property damage, identified by the word *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.

Additional Information

URL	Description
http://www.info.teradata.com	Use the Teradata Information Products Publishing Library site to: • View or download a manual: **1.** Under **Online Publications**, select **General Search**. **2.** Enter your search criteria and click **Search**. • Download a documentation CD-ROM: **1.** Under **Online Publications**, select **General Search**. **2.** In the **Title or Keyword** field, enter *CD-ROM*, and click **Search**.
www.teradata.com	The Teradata home page provides links to numerous sources of information about Teradata. Links include: • Executive reports, white papers, case studies of customer experiences with Teradata, and thought leadership • Technical information, solutions, and expert advice • Press releases, mentions and media resources
www.teradata.com/TEN/	Teradata Customer Education delivers training that builds skills and capabilities for our customers, enabling them to maximize their Teradata investment.

URL	Description
https://tays.teradata.com	Use Teradata @ Your Service to access Orange Books, technical alerts, and knowledge repositories, view and join forums, and download software patches.
Teradata Community - Developer Zone	Developer Zone provides new ideas and developer information.
Teradata Downloads	Provides publicly available downloads from Teradata.

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.

Teradata Database Optional Features

This book may include descriptions of the following optional Teradata Database features and products:

- In-Memory Optimization
- Teradata Columnar
- Teradata Row-Level Security
- Teradata Secure Zones
- Teradata Temporal
- Teradata Virtual Storage (VS)

You may not use these features without the appropriate licenses. The fact that these features may be included in product media or downloads, or described in documentation that you receive, does not authorize you to use them without the appropriate licenses.

Contact your Teradata sales representative to purchase and enable optional features.

CHAPTER 1

Introduction: The Data Warehouse

Overview of the Data Warehouse

Initially, the data warehouse was a *historical* database, enterprise-wide and centralized, containing data derived from an operational database.

The data in the data warehouse was:

- Subject-oriented
- Integrated
- Usually identified by a timestamp
- Nonvolatile, that is, nothing was added or removed

Rows in the tables supporting the operational database were loaded into the data warehouse (the historical database) after they exceeded some well-defined date.

Data could be queried, but the responses returned only reflected historical information. In this sense, a data warehouse was initially *static*, and even if a historical data warehouse contained data that was being updated, it would still not be an *active* data warehouse.

The Active Data Warehouse

An active data warehouse:

- Provides a single up-to-date view of the enterprise on one platform.
- Represents a logically consistent store of detailed data available for strategic, tactical, and event driven business decision making.
- Relies on timely updates to the critical data (as close to real time as needed).
- Supports short, tactical queries that return in seconds, alongside of traditional decision support.

Strategic Queries

Strategic queries represent business questions that are intended to draw strategic advantage from large stores of data.

Strategic queries are often complex, involving aggregations and joins across multiple tables in the database. They are sometimes long-running and tend not to have a strict service level expectation.

Strategic queries are often ad hoc. They may require significant database resources to execute, they are often submitted from third-party tools, and they can take advantage of session pooling.

Tactical Queries

Tactical queries are short, highly tuned queries that facilitate action-taking or decision-making in a time-sensitive environment. They usually come with a clear service level expectation and consume a very small percentage of the overall system resources.

Tactical queries are usually repetitively executed and take advantage of techniques such as request (query plan) caching and session-pooling.

Teradata Active Solutions

In an active data warehouse, Teradata provides both strategic intelligence and operational intelligence.

- Strategic intelligence entails delivering intelligence through tools and utilities and query mechanisms that support strategic decision-making.

  This includes, for example, providing users with simple as well as complex reports throughout the day which indicate the business trends that have occurred and that are occurring, which show why such trends occurred, and which predict if they will continue to occur.
- Operational intelligence entails delivering intelligence through tools and utilities and query mechanisms that support front-line or operational decision-making.

  This includes, for example, ensuring aggressive Service Level Goals (SLGs) with respect to high performance, data freshness, and system availability.

Active Load

Teradata is able to load data actively and in a non-disruptive manner and, at the same time, process other workloads.

Teradata delivers Active Load through methods that support continuous data loading. These include streaming from a queue, more frequent batch updates, and moving changed data from another database platform to Teradata.

These methods exercise such Teradata Database features as queue tables and triggers, and use FastLoad, MultiLoad, TPump, standalone utilities, and Teradata Parallel Transporter.

Teradata can effectively manage a complex workload environment on a “single version of the business.”

Active Access

Teradata is able to access analytical intelligence quickly and consistently in support of operational business processes.

But the benefit of Active Access entails more than just speeding up user and customer requests. Active Access provides intelligence for operational and customer interactions consistently.

Active Access queries, also referred to as tactical queries, support tactical decision-making at the front-line. Such queries can be informational, such as simply retrieving a customer record or transaction, or they may include complex analytics.

Active Events

Teradata is able to detect a business event automatically, apply business rules against current and historical data, and initiate operational actions when appropriate. This enables enterprises to reduce the latency between the identification of an event and taking action with respect to it. Active Events entails more than event detection.

When notified of something important, Teradata presents users with recommendations for appropriate action. The analysis done for users may prescribe the best course of action or give them alternatives from which to choose.

Active Workload Management

Teradata is able to manage mixed workloads dynamically and to optimize system resource utilization to meet business goals.

Teradata Active System Management (TASM) is a portfolio of products that enables real-time system management.

TASM assists the database administrator in analyzing and establishing workloads and resource allocation to meet business needs. TASM facilitates monitoring workload requests to ensure that resources are used efficiently and that dynamic workloads are prioritized automatically.

TASM also provides state-of-the-art techniques to visualize the current operational environment and to analyze long-term trends. TASM enables database administrators to set SLGs, to monitor adherence to them, and to take any necessary steps to reallocate resources to meet business objectives.

Active Enterprise Integration

Teradata is able to integrate itself into enterprise business and technical architectures, especially those that support business users, partners, and customers. This simplifies the task of coordinating enterprise applications and business processes.

For example, a Teradata event, generated from a database trigger, calls a stored procedure, which inserts a row into a queue table and publishes a message via the Teradata JMS Provider. The message is delivered to a JMS queue on a WebLogic, SAP NetWeaver, or other JMS-compatible application server. SAP Customer Relationship Management receives the message, notifies the user, and takes an action.

Active Availability

Teradata is able to meet business objectives for its own availability. Moreover, it assists customers in identifying application-specific availability, recoverability, and performance requirements based on the impact of enterprise downtime. Teradata can also recommend strategies for achieving system availability goals.

CHAPTER 2

Teradata Database and Tools

Teradata Database Design and Architecture

Teradata Database is an information repository supported by tools and utilities that make it a complete and active relational database management system.

Teradata developers designed Teradata Database from mostly off-the-shelf hardware components. The result was an inexpensive, high-quality system that exceeded the performance of conventional relational database management systems. The hardware components of Teradata Database evolved from those of a simple parallel database computer into those of a general-purpose, massively parallel computer running the database.

The architecture supports both single-node, Symmetric Multiprocessing (SMP) systems and multinode, Massively Parallel Processing (MPP) systems in which the distributed functions communicate by means of a fast interconnect structure. The interconnect structure is the BYNET for MPP systems and the boardless BYNET for SMP systems.

Attachment Methods

To support its role in the active warehouse environment, Teradata Database can use either of two attachment methods to connect to other operational computer systems as illustrated in the following table.

This attachment method…	Allows the system to be attached…
workstation	to intelligent workstations through a TCP/IP-based network.
mainframe	to an IBM mainframe computer.

Character Support

Teradata Database has an international customer base. To accommodate communications in different languages, Teradata Database supports non-Latin character sets, including KANJI, KANJISJIS, and UNICODE.

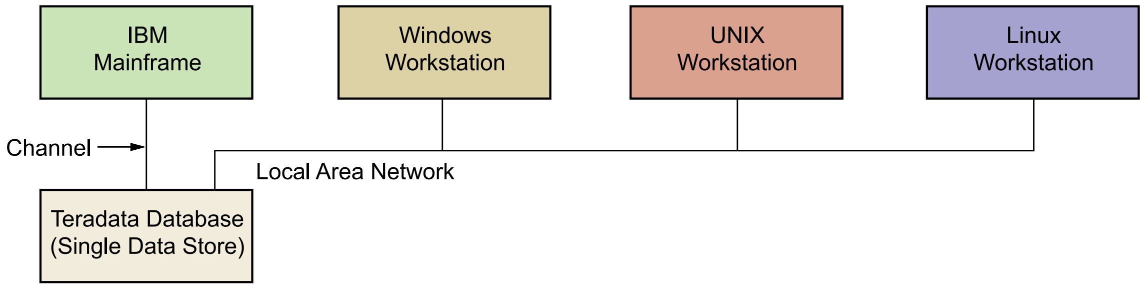
For detailed information about international character set support, see Chapter 14: “International Language Support.”

Single Data Store

A design goal of Teradata Database was to provide a *single data store* for a variety of client architectures. This approach greatly reduces data duplication and inaccuracies that can creep into data that is maintained in multiple stores.

This approach to data storage is known as the *single version of the business*, and Teradata Database implements this through heterogeneous client access. Clients can access a single copy of enterprise data and Teradata Database takes care of such things as data type translation, connections, concurrency, and workload management.

The following figure illustrates the idea of heterogeneous client access, where mainframe-attached and workstation-attached systems can access and manipulate the same database simultaneously. In this figure, the mainframe is attached via channel connections and other systems are attached via network connections.



Teradata Database Capabilities

Teradata Database allows users to view and manage large amounts of data as a collection of related tables. Some of the capabilities of Teradata Database are listed in the following table.

Teradata Database provides…	That…
capacity	includes: • Scaling from gigabytes to terabytes to petabytes of detailed data stored in billions of rows. • Scaling to millions of millions of instructions per second (MIPS) to process data.
parallel processing	makes Teradata Database faster than other relational systems.
single data store	• can be accessed by workstation-attached and mainframe-attached systems. • supports the requirements of many diverse clients.
fault tolerance	automatically detects and recovers from hardware failures.
data integrity	ensures that transactions either complete or rollback to a consistent state if a fault occurs.

Teradata Database provides…	That…
scalable growth	allows expansion without sacrificing performance.
SQL	serves as a standard access language for relational database communication. SQL enables users to control data.

Teradata Database Software

Teradata Database software resides on the platform and implements the relational database environment. The platform software includes the following functional modules.

This module…	Provides…
Database Window	a tool that you can use to control the operation of Teradata Database.
Teradata Gateway	communications support. The Gateway software validates messages from clients that generate sessions over the network and it controls encryption.
Parallel Database Extensions (PDE)	a software interface layer that lies between the operating system and database. PDE enables the database to operate in a parallel environment. For more information about PDE, see "Parallel Database Extensions".
Teradata Database management software	the following modules: • Parsing Engine (PE) • Access module processor (AMP) • Teradata Database file system For more information about Teradata Database file system, see "Teradata Database File System".

Software Installation

The Parallel Upgrade Tool (PUT) automates much of the installation process for Teradata Database software. There are two major operational modes for PUT.

The operational mode…	Does the following…
Major upgrade	upgrades one or more software products to the next version.
Patch upgrade	applies patch packages to one or more software products.

Teradata Tools and Utilities

Teradata Tools and Utilities is a comprehensive suite of tools and utilities designed to operate in the client environment used to access Teradata Database.

Supported Platforms

Teradata Tools and Utilities applications are supported on a wide variety of operating platforms. For a list of supported platforms and product versions, see *Teradata Tools and Utilities ##.# Supported Platforms and Product Versions*, which is available from .

Installation Guides for Teradata Tools and Utilities

To learn how to install Teradata Tools and Utilities, see the following installation guides:

- *Teradata Tools and Utilities for IBM z/OS Installation Guide*
- *Teradata Tools and Utilities Installation Guide for UNIX® and Linux®*
- *Teradata Tools and Utilities for Microsoft Windows Installation Guide*

Application Programming Interfaces

Teradata provides a number of standardized interfaces to facilitate easy development of applications that access Teradata Database. The following table describes Teradata Application Programming Interfaces (APIs) and what each provides.

This utility...	Provides…
.NET Data Provider for Teradata	access to Teradata Database for .NET Framework applications. The Data Provider conforms to the ADO.NET specification. The ADO.NET specification is available from the Microsoft Developer Network (MSDN).
ODBC Driver for Teradata	access to Teradata Database from various tools, increasing the portability of access.
Teradata Call-Level Interface version 2 (CLIv2)	a Teradata proprietary API and library used by applications to communicate with Teradata Database.
Teradata Data Connector	a block-level I/O interface to one or more access modules that interface to hardware storage devices, software messaging systems.
Teradata JDBC Driver	platform-independent, Java-application access to Teradata Database from various tools increasing portability of data.

Mainframe-Attached Connectivity Tools

The following table describes tools and utilities on channel-attached clients and what each tool or utility provides.

This utility for mainframe-attached clients…	Provides…
IBM® Customer Information Control System (CICS) Interface for Teradata	an interface that enables CICS macro or command-level application programs to access Teradata Database resources.
IBM IMS Interface for Teradata	an Information Management System (IMS) interface to Teradata Database.
Teradata CLIv2	a Teradata proprietary API and library used by applications to communicate with Teradata Database.
Teradata Director Program (TDP)	a high-performance interface for messages sent between the client and Teradata Database.

Language Preprocessors

Language preprocessors enable applications to access Teradata Database by interpreting SQL statements in C, COBOL, or Programming Language 1 (PL/I) programs. The following table describes the available Teradata preprocessors.

This preprocessor…	Provides a mechanism for…	For…
Teradata C Preprocessor	embedding SQL in C programs	mainframe-attached and workstation-attached clients.
Teradata COBOL Preprocessor	embedding SQL in COBOL programs	mainframe-attached and some workstation-attached clients.
Teradata PL/I Preprocessor	embedding SQL in PL/I programs	mainframe-attached clients.

Load and Unload Utilities

Teradata load and unload utilities offer a powerful solution for managing all your data load requirements from batch to real-time. Teradata load and unload utilities are fully parallel to provide optimal and scalable performance for getting data in and out of your Teradata Database. In addition, you can import and export data to and from host-based and client-resident data sources, including mainframe host databases, enterprise server databases or departmental data marts. The following table describes Teradata load and unload utilities.

This utility…	Provides…
Basic Teradata Query (BTEQ)	a general-purpose, command-based tool that enables you to communicate with a Teradata Database. BTEQ provides an interactive or batch interface that allows you to submit SQL

This utility…	Provides…
	statements, import and export data, and generate reports.
Teradata FastExport	a means of reading large volumes of data from Teradata Database.
Teradata FastLoad	high-performance data loading from client files into empty tables.
Teradata MultiLoad	high-performance data maintenance, including inserts, updates, and deletions to existing tables.
Teradata Parallel Data Pump (TPump)	continuous update of tables; performs insert, update, and delete operations or a combination of those operations on multiple tables using the same source feed.
Teradata Parallel Transporter (Teradata PT)	a means to load data into and unload data from any accessible table in Teradata Database or from any other data stores for which an access operator or an access module exists.
Teradata Parallel Transporter Application Programming Interface (Teradata PT API)	a set of programming interfaces for loading and unloading data to and from Teradata Database systems. Teradata PT API enables an application to access the Teradata Database using proprietary Teradata load and export protocols. Because Teradata PT API is a functional library that is part of your client applications, it provides more control during the load and unload processes.

Teradata Access Modules

Access modules are adapters that allow all Teradata load and unload utilities to interface with a variety of data sources through standards-based interfaces. These standards-based modules let you read from a given data source as if you were reading from a flat file. The following table describes Teradata access modules.

This utility…	Provides…
Named Pipes Access Module	an inter-process communication link between a writer process, such as Teradata FastExport, and a reader process, such as Teradata FastLoad.
Teradata Access Module for JMS	a fast, asynchronous method to import and export data between Teradata Database and any JMS-enabled messaging system. Teradata Access Module for JMS can be invoked by Teradata load and unload utilities.
Teradata OLE DB Access Module	a dynamic link library (DLL) that acts as an interface between Teradata load and export utilities and data sources for which an OLE DB provider is available. The access module quickly

This utility…	Provides…
	moves data between an OLE DB data source and Teradata Database without requiring intermediate storage.
Teradata Data Connector	a software layer between a client utility and an access module. It establishes, maintains, and monitors the connection between a client utility and an access module by being statically linked to the client and dynamically linked to the access module.
Teradata WebSphere MQ Access Module	Teradata utilities to import data using IBM WebSphere MQ message queuing middleware.

Teradata Active System Management

Teradata Active System Management (TASM) is a portfolio of products designed for the analysis, organization and control of workloads inside the Teradata solution. These products help you keep pace with your increasingly complex production workloads, especially those dealing with critical business intelligence, analytics, or tactical queries. The following table describes Teradata Tools and Utilities products that are part of TASM and what each tool provides.

This tool for workstation-attached clients…	Provides…
Teradata Viewpoint	a means to set up rules (including workload limits on accounts, users, and objects, such as databases and tables) that manage database access, increase database efficiency, and enhance workload capacity.
Teradata Workload Analyzer (Teradata WA)	recommendations on workload definitions and operating rules to ensure that database performance meets Service Level Goals (SLGs).

Teradata Analyst Pack

As application environments expand to include mixed workloads for both decision support and near-real-time analytic processing, maximizing the performance of the Teradata Database becomes more challenging. Teradata provides the Teradata Analyst Pack to enable you to analyze and tune your database queries for better performance. The Teradata Analyst Pack simplifies your DBA and query planner's jobs by automating the steps required to analyze and optimize your Teradata Databases. The following table describes the Teradata Analyst Pack tools and what each tool provides.

This tool for workstation-attached clients…	Provides…
Teradata Index Wizard	analyses of various SQL query workloads and suggests candidate indexes to enhance performance of those queries.

This tool for workstation-attached clients…	Provides…
Teradata System Emulation Tool (Teradata SET)	the capability to examine the query plans generated by the test system Optimizer as if the queries were processed on the production system.
Teradata Visual Explain (Teradata VE)	provides a graphical depiction of the execution plan of complex SQL statements.

Teradata Database Management Tools

You must ensure that your database can keep pace with changing requirements and the addition of new users and business applications. Teradata Database has a rich collection of tools and facilities to control database operation, administration, and maintenance.

The following table describes database management tools and what each tool provides.

This tool for workstation-attached clients…	Provides…
Unity Director	a program that routes sessions for high availability purposes.
Teradata Query Scheduler (QS)	a means to manage requests input to Teradata Database and keep the database running at optimum performance levels. The product consists of client, platform, and administrator components, plus a separate database within Teradata Database.
Teradata SQL Assistant	a means of information storage, manipulation, and discovery in Teradata Database (or other database that provides an ODBC interface). Teradata SQL Assistant also supports connectivity using .NET Data Provider for Teradata. Teradata SQL Assistant Java Edition runs on any platform that supports Java. It can connect to any JDBC-compliant database.
Teradata Studio	a GUI for administration, query development, and management tasks on Teradata Databases, Teradata Aster Databases, and Hadoop systems.

Teradata Viewpoint

In addition to the Teradata Tools and Utilities database management tools, Teradata Viewpoint provides a suite of web-based applications that enable you to monitor and manage Teradata Database system performance from anywhere using a standard browser. For more information, see “Teradata Viewpoint”.

Storage Management Utilities

The following table describes storage management utilities and what each utility provides.

This utility for mainframe- and workstation-attached clients…	Provides…
Archive/Recovery (Teradata ARC)	a means of archiving data to tape and disk and restoring data to Teradata Database.
Teradata Data Stream Architecture (DSA)	similar functionality to Teradata ARC, but generally provides improved performance.
Backup Application Software includes the following: • NetVault® Plug-in for Teradata • NetBackup™ Teradata Extension • Tivoli® Storage Manager Teradata Extension	open architecture products for backup and restore functions for Windows and Linux clients.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database and Teradata Tools and Utilities books.

IF you want to learn more about…	See…
Archive/Recovery utility	*Teradata Archive/Recovery Utility Reference*
Data Stream Architecture	Data Stream Architecture documentation
Backup/Archive/Recovery (BAR), including BAR framework products	*Teradata BAR Solutions Guide*
Teradata BTEQ	*Basic Teradata Query Reference*
CICS Interface for Teradata	*IBM CICS Interface for Teradata Reference*
Teradata CLIv2	• *Teradata Call-Level Interface Version 2 Reference for Mainframe-Attached Systems* • *Teradata Call-Level Interface Version 2 Reference for Workstation-Attached Systems*
IMS/DC Interface for Teradata	*IBM IMS/DC Interface for Teradata Reference*
Teradata JDBC Driver	*Teradata JDBC Driver User Guide*
Load and export utilities	• *Teradata FastExport Reference* • *Teradata FastLoad Referencee* • *Teradata MultiLoad Reference* • *Teradata Parallel Data Pump Reference*

IF you want to learn more about…	**See…**
Teradata ODBC Driver	*ODBC Driver for Teradata User Guide*
SQL	*SQL Fundamentals*
Teradata Database architecture	*Database Design*
Teradata Data Connector	*Teradata Tools and Utilities Access Module Reference*
Teradata Director Program	*Teradata Director Program Reference*
Teradata Index Wizard	*Teradata Index Wizard User Guide*
.NET Data Provider for Teradata	.NET Data Provider for Teradata Developers Guide and Reference, which is available from (Teradata Downloads).
Teradata Parallel Transporter, including Teradata Parallel Transport Application Programming Interface	• *Teradata Parallel Transporter Application Programming Interface Programmer Guide* • *Teradata Parallel Transporter Operator Programmer Guide* • *Teradata Parallel Transporter Reference* • *Teradata Parallel Transporter User Guide*
Teradata Preprocessor2 for Embedded SQL	• *Teradata Preprocessor2 for Embedded SQL Programmer Guide* • *SQL Stored Procedures and Embedded SQL*
Unity Director	*Teradata Unity User Guide*
Teradata Query Scheduler	• *Teradata Query Scheduler Administrator Guide* • *Teradata Query Scheduler User Guide*
Teradata SQL Assistant	• *Teradata SQL Assistant for Microsoft Windows User Guide* • *Teradata SQL Assistant Java Edition Release Definition*
Teradata Studio	Teradata Studio documentation
Teradata System Emulation Tool	• Database Administration • *Teradata System Emulation Tool User Guide*
Teradata Access Modules	• *Teradata Tools and Utilities Access Module Programmer Guide* • *Teradata Tools and Utilities Access Module Reference*
Teradata Visual Explain	*Teradata Visual Explain User Guide*
Teradata Workload Analyzer	*Teradata Workload Analyzer User Guide*
Teradata Studio products	Teradata Studio documentation

CHAPTER 3

Teradata Database Model

Overview

This chapter describes the concepts on which relational databases are modeled, and discusses some of the objects that are part of a relational database.

Relational Model

The relational model for database management is based on concepts derived from the mathematical theory of sets. Basically, set theory defines a table as a relation. The number of rows is the *cardinality* of the relation, and the number of columns is the *degree*. Any manipulation of a table in a relational database has a consistent, predictable outcome, because the mathematical operations on relations are well-defined.

By way of comparison, database management products based on hierarchical, network, or object-oriented architectures are not built on rigorous theoretical foundations. Therefore, the behavior of such products is not as predictable or as flexible as that of relational products.

The SQL Optimizer in the database builds the most efficient access path to requested data. The Optimizer can readily adapt to changes in system variables by rebuilding access paths without programmer intervention. This adaptability is necessary because database definitions and data demographics can change.

Relational Database

Users perceive a relational database as a collection of objects, that is, as tables, views, macros, stored procedures, and triggers that are easily manipulated using SQL directly or specifically developed applications.

Set Theory and Relational Database Terminology

Relational databases are a generalization of the mathematics of set theory relations. Thus, the correspondences between set theory and relational databases are not always direct. The following table notes the correspondence between set theory and relational database terms.

Set Theory Term	Relational Database Term
Relation	Table
Tuple	Row
Attribute	Column

Tables, Rows, and Columns

Tables are two-dimensional objects consisting of rows and columns. Data is organized in tabular format and presented to the users of a relational database. References between tables define the relationships and constraints of data inside the tables themselves. For more information on different types of tables, see “Tables”.

Table Constraints

You can define conditions that must be met before Teradata Database writes a given value to a column in a table. These conditions are called *constraints*. Constraints can include value ranges, equality or inequality conditions, and intercolumn dependencies. Teradata Database supports constraints at both the column and table levels.

During table creation and modification, you can specify constraints on single-column values as part of a column definition or on multiple columns using the CREATE and ALTER TABLE statements.

Rows and Columns

A column always contains the same kind of information. For example, a table that has information about employees would have a column for last name and nothing other than the employee last names should be placed in that column.

A row is one instance of all the columns in a table. For example, each row in the employee table would contain, among other things, the first name and the last name for that employee. The columns in a table represent entities, relationships, or attributes.

An *entity* is a person, place, or thing about which the table contains information. The table mentioned in the previous paragraphs contains information about the employee entity. Each table holds only one kind of row. The relational model requires that each row in a table be uniquely identified. To accomplish this, you define a uniqueness constraint to identify each row in the table. For more information about primary keys, see “Relationships Between Primary Indexes and Primary Keys”.

For More Information

For more information on the topics presented in this chapter, see *Database Design*.

CHAPTER 4
Teradata Database Hardware and Software Architecture

Overview

This chapter briefly describes Teradata Database hardware components and software architecture.

SMP and MPP Platforms

The hardware that supports Teradata Database software is based on Symmetric Multiprocessing (SMP) technology. The hardware can be combined with a communications network that connects the SMP systems to form Massively Parallel Processing (MPP) systems. The components of the SMP and MPP hardware platforms include the following.

Component	Description	Function
Processor Node	A hardware assembly containing several, tightly coupled, central processing units (CPUs) in an SMP configuration. An SMP node is connected to one or more disk arrays with the following installed on the node: • Teradata Database software • Client interface software • Operating system • Multiple processors with shared-memory • Failsafe power provisions An MPP configuration is a configuration of two or more loosely coupled SMP nodes.	Serves as the hardware platform upon which the database software operates.
BYNET	Hardware interprocessor network to link nodes on an MPP system. **Note:** Single-node SMP systems use a software-configured virtual BYNET driver to implement BYNET services.	Implements broadcast, multicast, or point-to-point communication between processors, depending on the situation.

These platforms use virtual processors (vprocs) that run a set of software processes on a node under the Parallel Database Extensions (PDE). For information about PDE, see “Parallel Database Extensions”.

Vprocs provide the parallel environment that enables Teradata Database to run on SMP and MPP systems. For more information on vprocs, see “Virtual Processors”.

The BYNET

At the most elementary level, you can look at the BYNET as a switched fabric that loosely couples all the SMP nodes in a multinode system. But the BYNET has capabilities that range far beyond those of a simple system bus.

The BYNET possesses high-speed logic that provides bi-directional broadcast, multicast, and point-to-point communication and merge functions.

A multinode system has at least two BYNETs. This creates a fault-tolerant environment and enhances interprocessor communication. Load-balancing software optimizes transmission of messages over the BYNETs. If one BYNET should fail, the second can handle the traffic.

Boardless BYNET

Single-node SMP systems use Boardless BYNET (or virtual BYNET) software to provide the same functions without the presence of BYNET hardware.

Disk Arrays

Teradata Database employs Redundant Array of Independent Disks (RAID) storage technology to provide data protection at the disk level. You use the RAID management software to group disk drives into RAID LUNS (Logical Units) to ensure that data is available in the event of a disk failure. Redundant implies that either data, functions, or components are duplicated in the architecture of the array.

Logical Units

The RAID Manager uses drive groups. A drive group is a set of drives that have been configured into one or more LUNs. Each LUN is uniquely identified.

The operating system recognizes a LUN as a disk and is not aware that it is actually writing to spaces on multiple disk drives. This technique allows RAID technology to provide data availability without affecting the operating system.

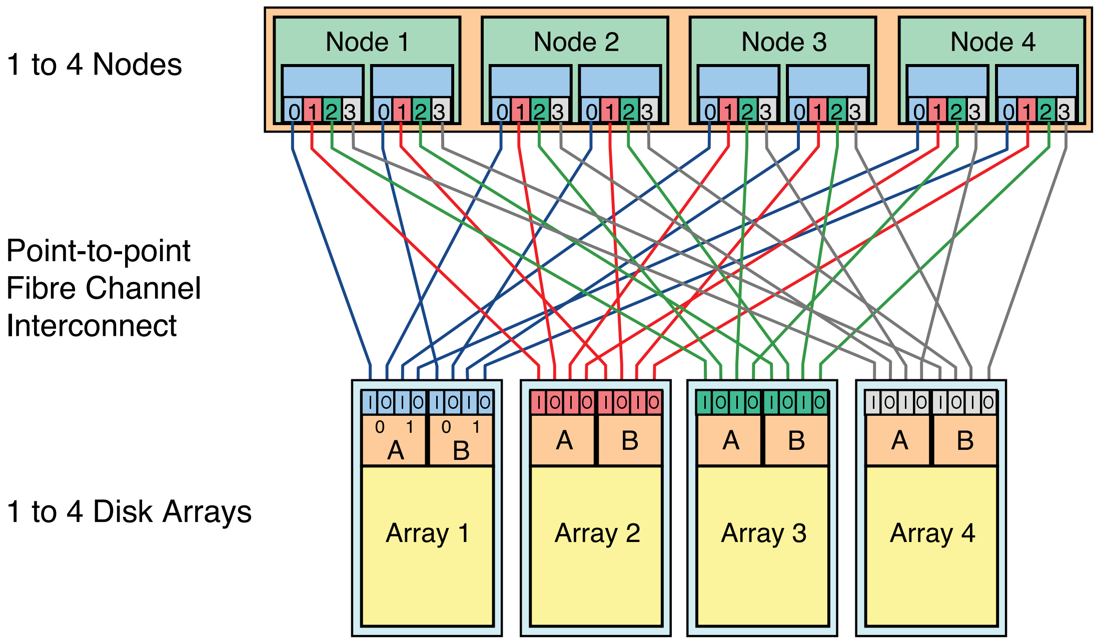
Vdisks

The group of cylinders currently assigned to an AMP is referred to as a vdisk, although the actual physical storage may derive from several different storage devices. For information about the role that AMPs play in Teradata Database architecture, see “Virtual Processors”.

Cliques

The clique is a feature of some MPP systems that physically group nodes together by multiported access to common disk array units. Inter-node disk array connections are made using FibreChannel (FC) buses.

FC paths enable redundancy to ensure that loss of a processor node or disk controller does not limit data availability.The nodes do not share data. They only share access to the disk arrays. The following figure illustrates a four-node clique:



A clique is the mechanism that supports the migration of vprocs under PDE following a node failure. If a node in a clique fails, then vprocs migrate to other nodes in the clique and continue to operate while recovery occurs on their home node. For more detailed information on vprocs see “Virtual Processors”.

PEs that manage TPA-hosted physical channel connections *cannot* migrate because they are dependent on the hardware that is physically attached to the node to which they are assigned.

PEs for workstation-attached TCP/IP connections *do* migrate when a node failure occurs, as do all AMPs.

Hot Standby Nodes

Hot standby nodes allow spare nodes to be incorporated into the production environment. Teradata Database can use spare nodes to improve availability and maintain performance levels in the event of a node failure. A hot standby node is a node that:

- Is a member of a clique.
- Does not normally participate in Teradata Database operations.
- Can be brought in to participate in Teradata Database operations to compensate for the loss of a node in the clique.

Configuring a hot standby node can eliminate the system-wide performance degradation associated with the loss of a node. A hot standby node is added to each clique in the system. When a node fails, all AMPs and all LAN-attached PEs on the failed node migrate to the node designated as the hot standby. The hot standby node becomes a production node. When the failed node returns to service, it becomes the new hot standby node.

Configuring hot standby nodes eliminates:

- Restarts that are required to bring a failed node back into service.
- Degraded service when vprocs have migrated to other nodes in a clique.

Virtual Processors

The versatility of Teradata Database is based on virtual processors (vprocs) that eliminate dependency on specialized physical processors. Vprocs are a set of software processes that run on a node under Teradata Parallel Database Extensions (PDE) within the multitasking environment of the operating system.

The following table contains information about the different types of vprocs:

Vproc Type	Description
AMP	Access module processors perform database functions, such as executing database queries. Each AMP owns a portion of the overall database storage.
GTW	Gateway vprocs provide a socket interface to Teradata Database.
Node	The node vproc handles PDE and operating system functions not directly related to AMP and PE work. Node vprocs cannot be externally manipulated, and do not appear in the output of the Vproc Manager utility.
PE	Parsing engines perform session control, query parsing, security validation, query optimization, and query dispatch.
RSG	Relay Services Gateway provides a socket interface for relaying dictionary changes to the Teradata Meta Data Services utility.
TVS	Manages Teradata Database storage. AMPs acquire their portions of database storage through the TVS vproc.

A single system can support a maximum of 16,384 vprocs. The maximum number of vprocs per node can be as high as 128, but is typically between 6 and 12.

Each vproc is a separate, independent copy of the processor software, isolated from other vprocs, but sharing some of the physical resources of the node, such as memory and CPUs. Multiple vprocs can run on an SMP platform or a node.

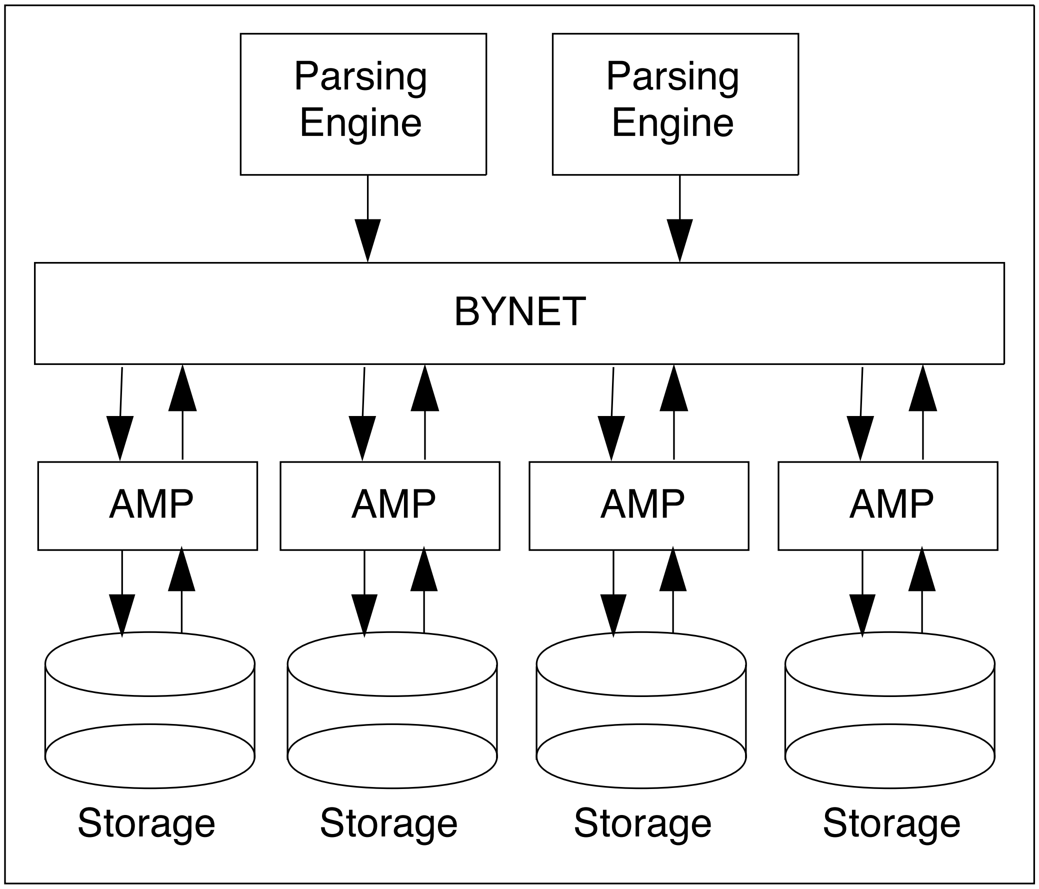
Vprocs and the tasks running under them communicate using unique-address messaging, as if they were physically isolated from one another. This message communication is done using the Boardless BYNET Driver software on single-node platforms or BYNET hardware and BYNET Driver software on multinode platforms.

Access Module Processor

The AMP vproc manages Teradata Database interactions with the disk subsystem. Each AMP manages a share of the disk storage.

AMP functions include…	For example…
database management tasks	• Accounting • Journaling • Locking tables, rows, and databases • Output data conversion
	During query processing: • Sorting • Joining data rows • Aggregation
file system management	disk space management.

Each AMP, as represented in the following figure, manages a portion of the physical disk space. Each AMP stores its portion of each database table within that space.



AMP Clusters

AMPs are grouped into logical clusters to enhance the fault-tolerant capabilities of Teradata Database. For more information on this method of creating additional fault tolerance in a system see Chapter 5: "Teradata Database RASUI."

Parsing Engine

The PE is the vproc that communicates with the client system on one side and with the AMPs (via the BYNET) on the other side.

Each PE executes the database software that manages sessions, decomposes SQL statements into steps, possibly in parallel, and returns the answer rows to the requesting client.

The PE software consists of the following elements.

Parsing Engine Elements	Process
Parser	Decomposes SQL into relational data management processing steps.
Optimizer	Determines the most efficient path to access data.
Generator	Generates and packages steps.
Dispatcher	Receives processing steps from the parser and sends them to the appropriate AMPs via the BYNET.
	Monitors the completion of steps and handles errors encountered during processing.
Session Control	• Manages session activities, such as logon, password validation, and logoff. • Recovers sessions following client or server failures.

Request Processing

SQL is the language that you use to make requests of Teradata Database, that is, you use SQL to query Teradata Database.

The SQL parser handles all incoming SQL requests in the following sequence:

1. The Parser looks in the Request cache to determine if the request is already there.

IF the request is…	THEN the Parser…
in the Request cache	reuses the plastic steps found in the cache and passes them to gncApply. Go to step 8 after checking privileges (step 4). Plastic steps are directives to the database management system that do not contain data values.
not in the Request cache	begins processing the request with the Syntaxer.

2. The Syntaxer checks the syntax of an incoming request.

IF there are…	THEN the Syntaxer…
no errors	converts the request to a parse tree and passes it to the Resolver.
errors	passes an error message back to the requestor and stops.

3. The Resolver adds information from the Data Dictionary (or cached copy of the information) to convert database, table, view, stored procedure, and macro names to internal identifiers.
4. The security module checks privileges in the Data Dictionary.

IF the privileges are…	THEN the Security module…
valid	passes the request to the Optimizer.
not valid	aborts the request and passes an error message and stops.

5. The Optimizer determines the most effective way to implement the SQL request.
6. The Optimizer scans the request to determine where to place locks, then passes the optimized parse tree to the Generator.
7. The Generator transforms the optimized parse tree into plastic steps, caches the steps if appropriate, and passes them to gncApply.
8. gncApply takes the plastic steps produced by the Generator, binds in parameterized data if it exists, and transforms the plastic steps into concrete steps.

   Concrete steps are directives to the AMPs that contain any needed user- or session-specific values and any needed data parcels.
9. gncApply passes the concrete steps to the Dispatcher.

The Dispatcher

The Dispatcher controls the sequence in which steps are executed. It also passes the steps to the BYNET to be distributed to the AMP database management software as follows:

1. The Dispatcher receives concrete steps from gncApply.
2. The Dispatcher places the first step on the BYNET; tells the BYNET whether the step is for one AMP, several AMPS, or all AMPs; and waits for a completion response.

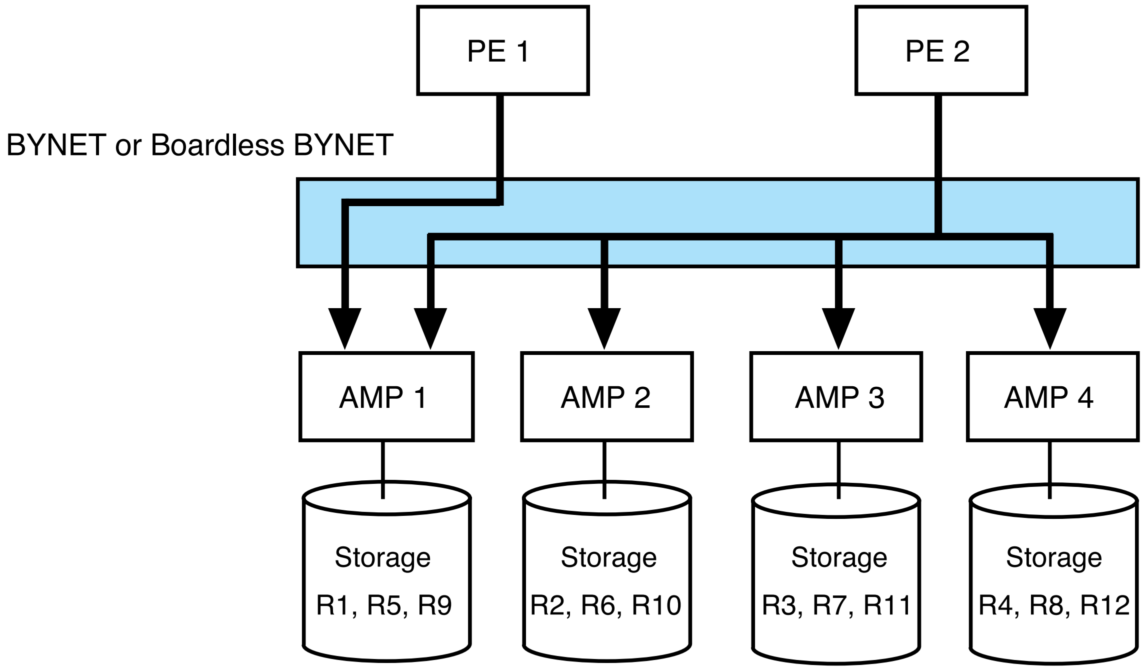
   Whenever possible, Teradata Database performs steps in parallel to enhance performance. If there are no dependencies between a step and the following step, the following step can be dispatched before the first step completes, and the two execute in parallel. If there is a dependency, for example, the following step requires as input the data produced by the first step, then the following step cannot be dispatched until the first step completes.
3. The Dispatcher receives a completion response from all expected AMPs and places the next step on the BYNET. It continues to do this until all the AMP steps associated with a request are done.

The AMPs

AMPs obtain the rows required to process the requests (assuming that the AMPs are processing a SELECT statement). The BYNET transmits messages to and from the AMPS and PEs. An AMP step can be sent to one of the following:

- One AMP
- A selected set of AMPs, called a dynamic BYNET group
- All AMPs in the system

The following figure is based on the example in the next section. If access is through a primary index and a request is for a single row, the PE transmits steps to a single AMP, as shown at PE1. If the request is for many rows (an all-AMP request), the PE makes the BYNET broadcast the steps to all AMPs, as shown in PE2. To minimize system overhead, the PE can send a step to a subset of AMPs, when appropriate.



Example: SQL Request

As an example, consider the following Teradata SQL requests using a table containing checking account information. The example assumes that AcctNo column is the unique primary index for Table_01. For information about the types of indexes used by Teradata Database, see Chapter 11: “Data Distribution and Data Access Methods.”

```
1. SELECT * FROM Table_01 WHERE AcctNo = 129317 ;
2. SELECT * FROM Table_01 WHERE AcctBal > 1000 ;
```

In this example:

- PEs 1 and 2 receive requests 1 and 2.
- The data for account 129317 is contained in table row R9 and stored on AMP1.
- Information about all account balances is distributed evenly among the disks of all four AMPs.

The sample Teradata SQL statement is processed in the following sequence:

1. PE 1 determines that the request is a primary index retrieval, which calls for the access and return of one specific row.
2. The Dispatcher in PE 1 issues a message to the BYNET containing an appropriate read step and R9/AMP 1 routing information. After AMP 1 returns the desired row, PE 1 transmits the data to the client.

3. The PE 2 Parser determines that this is an all-AMPs request, then issues a message to the BYNET containing the appropriate read step to be broadcast to all four AMPs.
4. After the AMPs return the results, PE 2 transmits the data to the client.

AMP steps are processed in the following sequence:

1. Lock—Serializes access in situations where concurrent access would compromise data consistency.

   For some simple requests using Unique Primary Index (UPI), Nonunique Primary Index (NUPI), or Unique Secondary Index (USI) access, the lock step will be incorporated into step 2. For information about indexes and their uses, see Chapter 11: "Data Distribution and Data Access Methods."
2. Operation—Performs the requested task. For complicated queries, there may be hundreds of operation steps.
3. End transaction—Causes the locks acquired in step 1 or 2 to be released.

   The end transaction step tells all AMPs that worked on the request that processing is complete.

Parallel Database Extensions

Parallel Database Extensions (PDE) is a software interface layer that lies between the operating system and Teradata Database. PDE supports the parallelism that gives Teradata Database its speed and linear scalability. PDE provides Teradata Database with the ability to:

- Run in a parallel environment
- Execute vprocs
- Apply a flexible priority scheduler to Teradata Database sessions
- Consistently manage memory, I/O, and messaging system interfaces across multiple OS platforms

PDE provides a series of parallel operating system services, which include:

- Facilities to manage parallel execution of database operations on multiple nodes.
- Dynamic distribution of database tasks.
- Coordination of task execution within and between nodes.

PDE enables MPP systems to take advantage of hardware features such as the BYNET and shared disk arrays.

Teradata Database File System

The file system is a layer of software between Teradata Database and PDE. File system service calls allow Teradata Database to store and retrieve data efficiently and with integrity without being concerned about the specific low-level operating system interfaces.

Teradata Database Window

Teradata DBW allows database administrators, system operators, and support personnel to control the operation of Teradata Database.

DBW is also the primary vehicle for starting and controlling the operation of Teradata Database utilities.

How Database Window Communicates with Teradata Database

DBW provides a graphical user interface to the Teradata Console Subsystem (CNS). Use DBW to issue database commands and run many of the database utilities. CNS is a part of the Parallel Database Extensions (PDE) software upon which the database runs.

Running DBW

You can run DBW from the following locations:

- System Console
- Remote workstation or computer

Note:

In order to use DBW, you must either be a member of the tdtrusted user group, or have been explicitly granted the right to run console utilities with the CNS GRANT command.

To learn more about the DBW interface, see "Database Window (xdbw)" in Utilities.

Teradata Generic Security Service

Network security for Teradata is provided by Teradata Generic Security Service (TDGSS) software. It provides for secure communication between a workstation client and Teradata Database.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database books.

IF you want to learn more about…	See…
Disk Arrays	*Database Administration*
Cliques	*Database Administration*
Hot Standby Nodes	*Database Administration*
Virtual Processors	• *Database Design* • *Database Administration* • *SQL Request and Transaction Processing*
Access Module Processor	• *Database Administration* • *Support Utilities*
Parsing Engine	*Database Administration*
Request Processing	*SQL Request and Transaction Processing*
Teradata Database File System	*Utilities*
Teradata Database Window	*Utilities*

IF you want to learn more about…	See…
Teradata General Security Service	*Security Administration*

CHAPTER 5

Teradata Database RASUI

Overview of RASUI

Teradata Database addresses the critical requirements of reliability, availability, serviceability, usability, and installability (RASUI) by combining the following elements:

- Multiple microprocessors in a Symmetric Multi-Processing, (SMP) arrangement.
- RAID disk storage technology.
- Protection of Teradata Database from operating anomalies of the client platform.

  Both hardware and software provide fault tolerance, some of which is mandatory and some of which is optional.

Software Fault Tolerance

This section explains the following Teradata Database facilities for software fault tolerance:

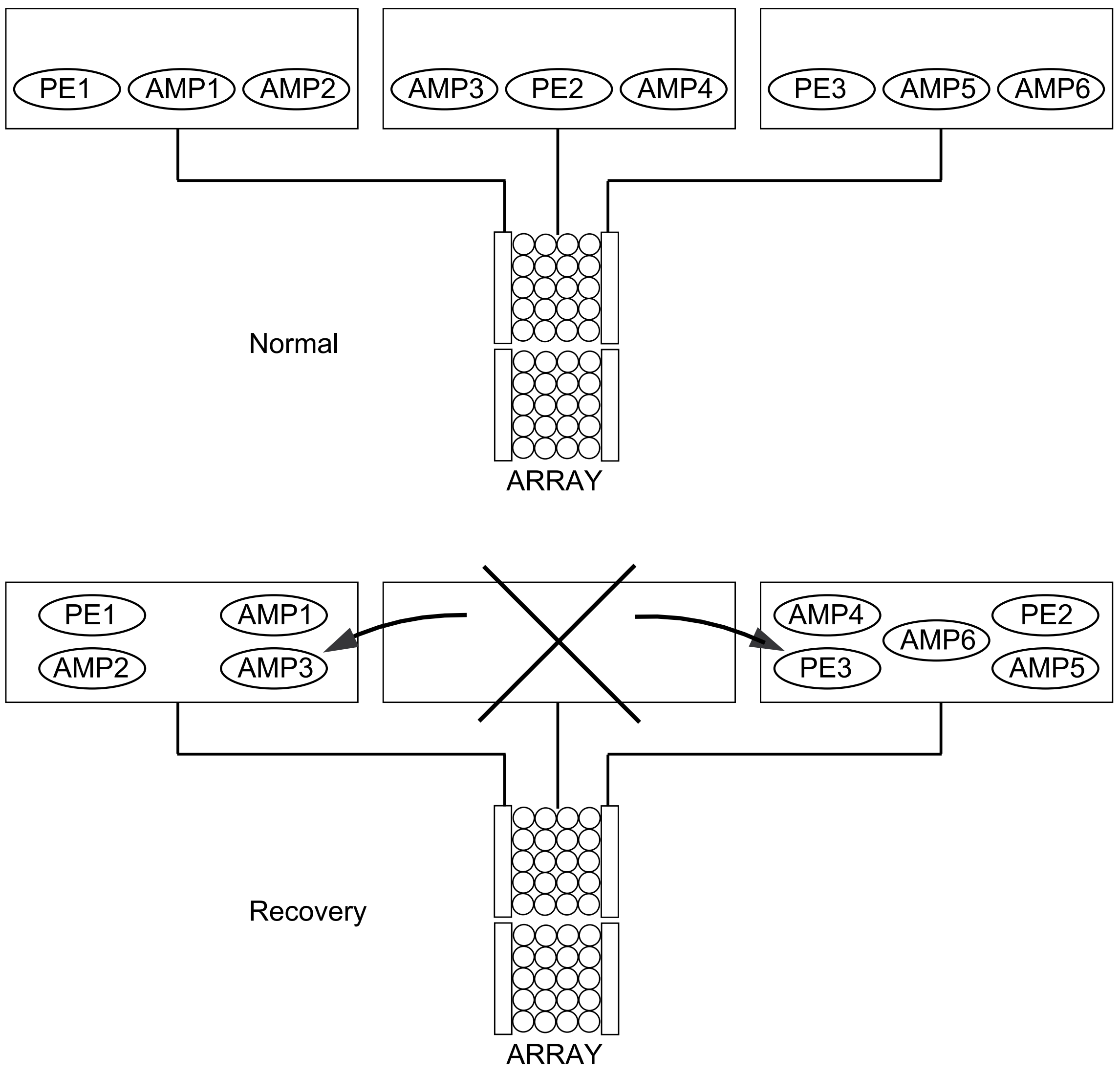
- Vproc migration
- Fallback tables
- AMP clusters
- Journaling
- Backup/Archive/Recovery
- Table Rebuild utility

Vproc Migration

Because the Parsing Engine (PE) and Access Module Processor (AMP) are vprocs and, therefore, software entities, they can migrate from their home node to another node within the same hardware clique if the home node fails for any reason. Although the system normally determines which vprocs migrate to which nodes, a user can configure preferred migratory destinations.

Vproc migration permits the system to function completely during a node failure, with some degradation of performance due to the non-functional hardware.

The following figure illustrates vproc migration, where the large X indicates a failed node, and arrows pointing to nodes still running indicate the migration of AMP3, AMP4, and PE2.



Note:
PEs that manage TPA-hosted physical channel connections cannot migrate during a node failure because they are dependent on the hardware that is physically attached to their assigned node.

Fallback Tables

A fallback table is a duplicate copy of a primary table. Each fallback row in a fallback table is stored on an AMP different from the one to which the primary row hashes. This storage technique maintains availability should the system lose an AMP and its associated disk storage in a cluster. In that event, the system would access data in the fallback rows.

The disadvantage of fallback is that this method doubles the storage space and the I/O (on INSERT, UPDATE, and DELETE statements) for tables. One advantage is that data is almost never unavailable because of one down AMP. Data is fully available during an AMP or disk outage. Another advantage is that if there is a data read error, Teradata Database can repair the primary copy of the data using the fallback copy.

Teradata Database permits the definition of fallback for individual tables. As a general rule, you should run all tables critical to your enterprise in fallback mode. You can run other, non-critical tables in nonfallback mode in order to maximize resource usage.

Even though RAID disk array technology may provide data access even when you have not specified fallback, neither RAID1 nor RAID5 provides the same level of protection as fallback.

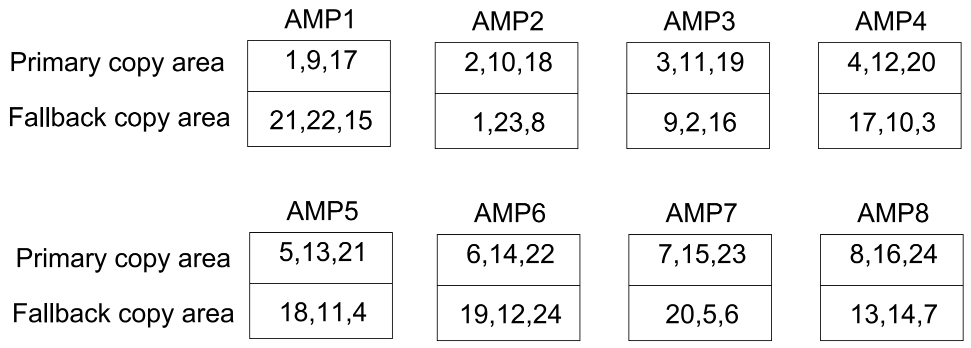
You specify whether a table is fallback or not using the CREATE TABLE (or ALTER TABLE) statement. The default is *not* to create tables with fallback.

AMP Clusters

A cluster is a group of 2-8 AMPs that provide fallback capability for each other. A copy of each row is stored on a separate AMP *in the same* cluster. In a large system, you would probably create many AMP clusters. However, whether large or small, the concept of a cluster exists even if all the AMPs are in one cluster.

One-Cluster Configuration

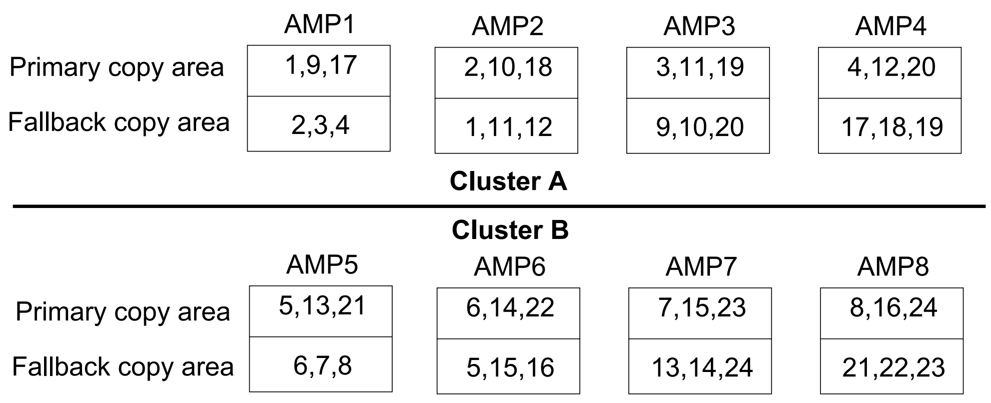
Pictures best explain AMP clustering. The following figure illustrates a situation in which fallback is present with one cluster, which is essentially an unclustered system.



Note that the fallback copy of any row is always located on an AMP different from the AMP which holds the primary copy. This is an entry-level fault tolerance strategy. In this example which shows only a few rows, the data on AMP3 is fallback protected on AMPs 4, 5, and 6. However, in practice, some of the data on AMP3 would be fallback protected on each of the other AMPs in the system. The system becomes unavailable if two AMPs in a cluster go down.

Smaller Cluster Configuration

The following figure illustrates smaller clusters. Decreasing cluster size reduces the likelihood that two AMP failures will occur in the same cluster. The illustration shows the same 8-AMP configuration now partitioned into 2 AMP clusters of 4 AMPs each.



Compare this clustered configuration with the earlier illustration of an unclustered AMP configuration. In the example, the (primary) data on AMP3 is backed up on AMPs 1, 2, and 4 and the data on AMP6 is backed up on AMPs 5, 7, and 8.

If AMPs 3 and 6 fail at the same time, the system continues to function normally. Only if two failures occur within the same cluster does the system halt.

Subpools are logical groupings of AMPs and disks for fault-tolerance. In a single-clique system to ensure that a disk failure will not bring down both AMPs in a cluster, disks and AMPs are divided into two subpools, and clustering is done across the subpools.

Journaling

Teradata Database supports tables that are devoted to journaling. A journal is a record of some kind of activity. Teradata Database supports several kinds of journaling. The system does some journaling on its own, while you can specify whether to perform other journaling.

The following table explains the capabilities of the different Teradata Database journals.

This type of journal…	Does the following…	And Occurs …
Down AMP recovery	• Is active during an AMP failure only • Journals fallback tables only • Is used to recover the AMP after the AMP is repaired, then is discarded	always.
Transient	• Logs BEFORE images for transactions • Is used by system to roll back failed transactions aborted either by the user or by the system • Captures: ◦ Begin/End Transaction indicators ◦ Before row images for UPDATE and DELETE statements ◦ Row IDs for INSERT statements ◦ Control records for CREATE, DROP, DELETE, and ALTER statements	always.

This type of journal…	Does the following…	And Occurs …
	• Keeps each image on the same AMP as the row it describes • Discards images when the transaction or rollback completes	
Permanent	• Is available for tables or databases • Can contain before images, which permit rollback, or after images, which permit rollforward, or both before and after images • Provides rollforward recovery • Provides rollback recovery • Provides full recovery of nonfallback tables • Reduces need for frequent, full-table archives	as specified by the user.

Backup Archive and Recovery

To archive, restore, and recover data in Teradata Database, you can use either:

- Teradata Data Stream Architecture, which is accessible via the Viewpoint BAR Operations portlet
- Teradata Archive and Recovery utility (ARC)

These programs can co-exist at a customer site; however, only the program that created the archive can read it and restore it. Teradata Data Stream Architecture cannot restore an archive created by ARC and vice versa.

For more information, see the documentation for Teradata Data Stream Architecture.

Table Rebuild Utility

Use the Table Rebuild utility to recreate a table, database, or entire disk on a single AMP under the following conditions:

- The table structure or data is damaged because of a software problem, head crash, power failure, or other malfunction.
- The affected tables are enabled for fallback protection.

Table rebuild can create all of the following on an AMP-by-AMP basis:

- Primary or fallback portions of a table.
- An entire table (both primary and fallback portions).
- All tables in a database.
- All tables on an individual AMP.

The Table Rebuild utility can also remove inconsistencies in stored procedure tables in a database. A Teradata Database system engineer, field engineer, or system support representative usually runs the Table Rebuild utility.

Hardware Fault Tolerance

Teradata Database provides the following facilities for hardware fault tolerance.

Facility	Description
Multiple BYNETs	Multinode Teradata Database servers are equipped with at least two BYNETs. Interprocessor traffic is never stopped unless all BYNETs fail. Within a BYNET, traffic can often be rerouted around failed components.
RAID disk units	• Teradata Database servers use Redundant Arrays of Independent Disks (RAIDs) configured for use as RAID1, RAID5, or RAIDS. Non-array storage cannot use RAID technology. • RAID1 arrays offer mirroring, the method of maintaining identical copies of data. • RAID5 or RAIDS protects data from single-disk failures with a 25% increase in disk storage to provide parity. • RAID1 provides better performance and data protection than RAID5/RAIDS, but is more expensive.
Multiple client-server connections	In a client-server environment, multiple connections between mainframe and workstation-based clients ensure that most processing continues even if one or several connections between the clients and server are not working. Vproc migration is a software feature supporting this hardware issue.
Isolation from client hardware defects	In a client-server environment, a server is isolated from many client hardware defects and can continue processing in spite of such defects.
Power supplies and fans	Each cabinet in a configuration has redundant power supplies and fans to ensure fail-safe operation.
Hot swap capability for node components	Teradata Database can allow some components to be removed and replaced while the system is running. This process is known as hot swap. Teradata Database offers hot swap capability for the following: • Disks within RAID arrays • Fans • Power supplies
Cliques	• A clique is a group of nodes sharing access to the same disk arrays. The nodes and disks are interconnected through FC buses and each node can communicate directly to all disks. This architecture provides and balances data availability in the case of a node failure. • A clique supports the migration of vprocs following a node failure. If a node in a clique fails, then its vprocs migrate to another node in the clique and continue to operate while recovery occurs on their home node. Migration minimizes the performance impact on the system. • PEs that manage TPA-hosted physical channel connections *cannot* migrate because they depend on the hardware that is physically attached to the assigned node.

Facility	Description
	• PEs for workstation-attached connections *do* migrate when a node failure occurs, as do all AMP vprocs. • To ensure maximum fault tolerance, no more than one node in a clique is placed in the same cabinet. Usually the battery backup feature makes this precaution unnecessary, but if you want maximum fault tolerance, then plan your cliques so the nodes are never in the same cabinet.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database and Teradata Tools and Utilities books.

IF you want to learn more about…	See…
Software Fault Tolerance, including: • Vproc Migration and Fallback Tables • Clusters (AMP clusters, one-cluster and small cluster configurations) • Journaling and Backup/Archive/Recovery (online archiving) • Table Rebuild Utility	• *Database Administration* • *Teradata Archive/Recovery Utility Reference* • *SQL Data Definition Language* • *Utilities*
Hardware Fault Tolerance	*Database Design*

CHAPTER 6

Client Communication with Teradata Database

Overview

Client applications can connect to Teradata Database using one of the following methods:

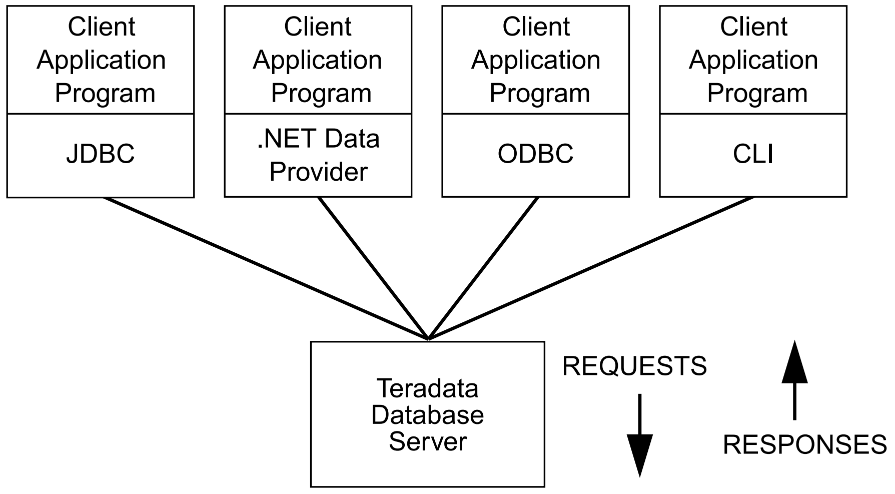
- Network-attached through a Local Area Network (LAN)
- Channel-attached through an IBM mainframe

Workstation Attachment Methods

Workstation-attached methods include:

- .NET Data Provider for Teradata
- Java Database Connectivity (JDBC)
- Microsoft OLE DB Provider for ODBC
- Open Database Connectivity (ODBC)
- Teradata CLIv2 for workstation-attached systems

The following figure illustrates the transparent connection between client applications and Teradata Database.


Client Application Program
JDBC
Client Application Program
.NET Data Provider
Client Application Program
ODBC
Client Application Program
CLI
Teradata Database Server
REQUESTS
RESPONSES

.NET Data Provider for Teradata

The .NET Data Provider for Teradata conforms to the ADO.NET specifications. ADO.NET provides a rich set of data access services to .NET Framework applications. The Data Provider allows .NET applications to access Teradata Database from all .NET Framework languages including C#, VB, F# and PowerShell.

Java Database Connectivity

JDBC is a specification for an API. The API allows platform-independent Java applications to access Teradata Database using SQL and external stored procedures.

The JDBC API provides a standard set of interfaces for:

- Opening connections to databases
- Executing SQL statements
- Processing results

Teradata JDBC Driver provides access to Teradata Database using the Java language. Teradata JDBC Driver is a type 4 (pure Java) JDBC Driver. It is a set of Java classes that use TCP/IP to communicate directly with Teradata Database.

Open Database Connectivity

ODBC Driver for Teradata provides an interface to Teradata Databases using the industry standard ODBC API. ODBC Driver for Teradata provides Core-level SQL and Extension-level 1 (with some Extension-level 2) function call capability using the Windows Sockets (WinSock) Transmission Control Protocol/Internet Protocol (TCP/IP) communications software interface. ODBC operates independently of CLI.

Teradata CLIv2 for Workstation-Attached Systems

Teradata CLIv2 for workstation-attached systems is a Teradata proprietary API and library providing an interface between applications on a TCP/IP-connected client and Teradata Database server. Teradata CLIv2 for workstation-attached systems can:

- Build parcels which are packaged by Micro Teradata Director Program (MTDP) and sent to Teradata Database using the Micro Operating System Interface (MOSI).
- Manage all interactions between the application and Teradata Database.
- Provide an application with a pointer to data rows returned from Teradata Database.

MTDP

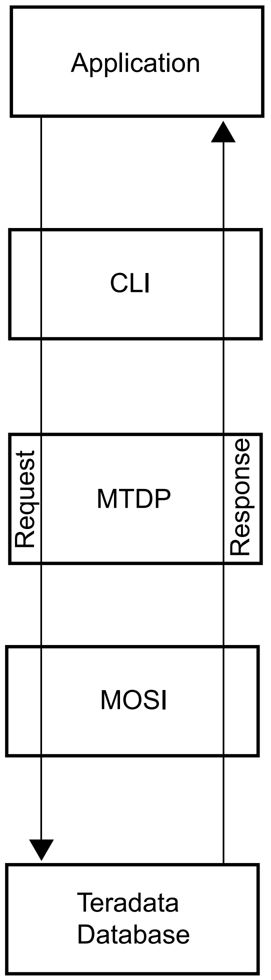
MTDP is the interface between Teradata CLIv2 for workstation-attached systems and MOSI. Functions of MTDP include:

- Session initiation and termination
- Logging, verification, recovery, and restart
- Physical input to and output from the server

Note:
MTDP does not control session balancing; session balancing on workstation-attached systems is controlled by Teradata Database Gateway on the server.

MOSI

MOSI is the interface between MTDP and Teradata Database. MOSI is a library of service routines providing operating system independence among clients that access Teradata Database. With MOSI, only one version of MTDP is required to run on all workstation-attached platforms.



Mainframe Attachment Method

Mainframe attachment uses Teradata CLIv2 for mainframe-attached systems.

Teradata CLIv2 for Mainframe-Attached Systems

Teradata CLIv2 for mainframe-attached systems is a collection of callable service routines providing the interface between applications and the Teradata Director Program (TDP) on an IBM mainframe client. Teradata CLIv2 for mainframe-attached systems can operate with all versions of IBM operating systems, including Customer Information Control System (CICS), Information Management System (IMS), and IBM System z Operating System.

By way of TDP, Teradata CLIv2 foraminifera-attached systems sends requests to the server and provides client applications with responses from the server. Teradata CLIv2 for mainframe-attached systems provides support for:

- Managing multiple serially executed requests in a session
- Managing multiple simultaneous sessions to the same or different servers
- Using cooperative processing so an application can perform operations on the client and the server at the same time
- Generally insulating the application from the details of communicating with a server

Teradata Director Program

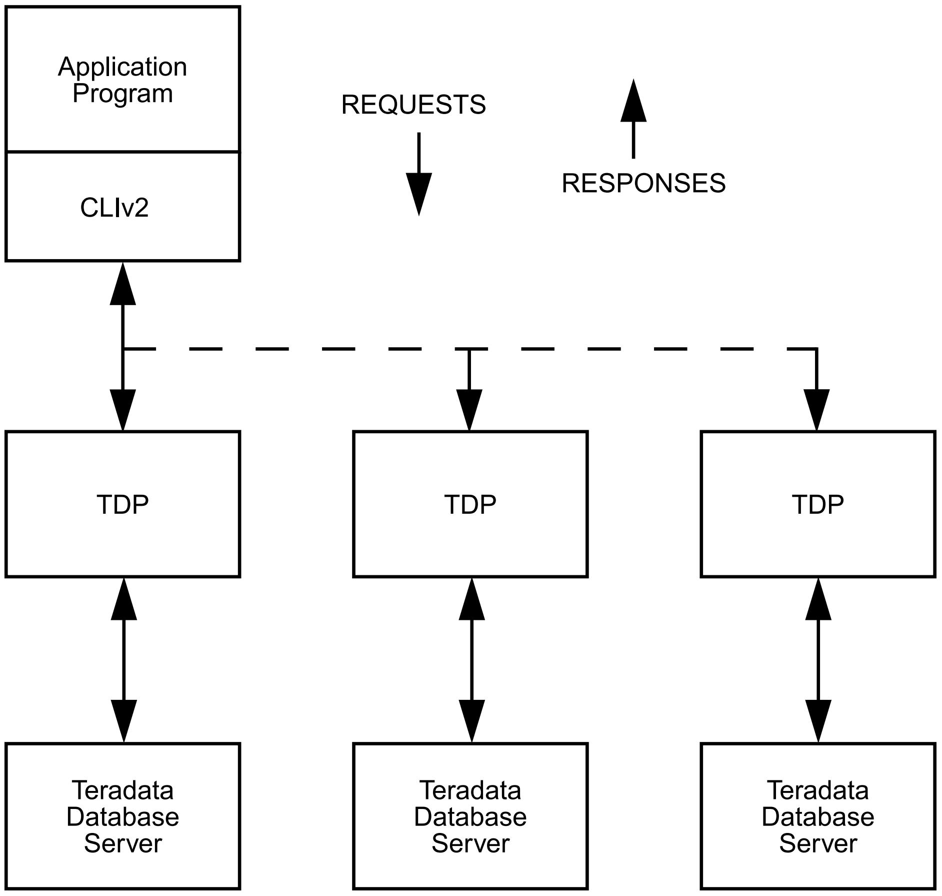
TDP manages communications between Teradata CLIv2 for mainframe-attached systems and the Teradata Database server. TDP executes on the same mainframe as Teradata CLIv2 for mainframe-attached systems, but runs as a different job or virtual machine. Although an individual TDP is associated with one logical server, any number of TDPs may operate and be simultaneously accessed by Teradata CLIv2 for mainframe-attached systems on the same mainframe. Each TDP is referred to by an application using an identifier called the TDPid that is unique in a mainframe; for example, TDP2.

Functions of TDP include:

- Session initiation and termination
- Logging, verification, recovery, and restart
- Physical input to and output from the server, including session balancing and queue maintenance
- Security

Teradata Database Server

A server implements the actual relational database that processes requests received from Teradata CLIv2 for mainframe-attached systems by way of TDP. The following figure illustrates the logical structure of the client-server interface on mainframe-attached systems:



For More Information

For more information on the topics presented in this chapter, see the following Teradata Tools and Utilities books.

IF you want to learn more about…	See…
Workstation Attachment Methods: • .NET Data Provider for Teradata • Teradata CLIv2 for workstation-attached systems • Java Database Connectivity, including Java language external stored procedures that use JDBC • Microsoft OLE DB Provider for ODBC • Open Database Connectivity	• *.NET Data Provider for Teradata Developers Guide and Reference*, which is available from (Teradata Downloads). • *Teradata Call-Level Interface Version 2 Reference for Workstation-Attached Systems* • *Teradata JDBC Driver User Guide* • *The Microsoft Developer Network (MSDN)* • *ODBC Driver for Teradata User Guide*
Mainframe Attachment Methods:	• *Teradata Call-Level Interface Version 2 Reference for Mainframe-Attached Systems* • *Teradata Director Program Reference*

IF you want to learn more about…	See…
• Teradata CLIv2 for mainframe-attached systems • Teradata Director Program	

CHAPTER 7

Database Objects, Databases, and Users

Tables

Tables are two-dimensional objects consisting of rows and columns. Data is organized in table format and presented to the users of a relational database. The following table describes basic table types:

Table Type	Description
ANSI Temporal	ANSI-compliant support for temporal tables. Using temporal tables, Teradata Database can process statements and queries that include time-based reasoning. Temporal tables record both system time (the time period when the information was recorded in the database) and valid time (the time period when the information is in effect or true in a real-world application).
Derived	A derived table: • Is a type of temporary table obtained from one or more other tables as the result of a subquery. • Is specified in an SQL SELECT statement. • Avoids the need to use the CREATE and DROP TABLE statements for storing retrieved information. • Is useful when you are coding more sophisticated, complex queries.
Error Logging	Error logging tables: • Store information about errors on an associated permanent table. • Log information about insert and update errors.
Global Temporary	Global temporary tables: • Are private to the session. • Are dropped automatically at the end of a session. • Have a persistent table definition stored in the Data Dictionary. The saved definition may be shared by multiple users and sessions with each session getting its own instance of the table.
Global Temporary Trace	Global temporary trace tables: • Store trace output for the length of the session. • Have a persistent table definition stored in the Data Dictionary. • Are useful for debugging SQL stored procedures (via a call to an external stored procedure written to the trace output) and external routines (UDFs, UDMs, and external stored procedures).
NoPI	NoPI tables are permanent tables that do not have primary indexes defined on them.

Table Type	Description
	They provide a performance advantage when used as staging tables to load data from FastLoad or TPump Array INSERT. They can have secondary indexes defined on them to avoid full-table scans during row access.
Permanent	Permanent tables allow different sessions and users to share table content.
Queue	Queue tables: • Are permanent tables with a timestamp column. The timestamp indicates when each row was inserted into the table. • Establish first-in first-out (FIFO) ordering of table contents, which is needed for customer applications requiring event processing.
Volatile	Volatile tables are used when: • Only one session needs the table. • Only the creator needs to access the table. • You want better performance than a global temporary table. • You do not need the table definition after the session ends. Note: The definition of a volatile table can survive across a system restart if it is contained in a macro.

For more information about table types, see *SQL Data Definition Language - Syntax and Examples* and *Database Design*.

Views

Database views are actually virtual tables that you can use *as if* they were physical tables to retrieve data defining columns from underlying views or tables, or from both.

A view does not contain data and is not materialized until a DML statement references it. View definitions are stored in the Data Dictionary.

Creating Views

A view is created from one or more base tables or from other views.

In fact, you can create hierarchies of views in which views are created from other views. This can be useful, but be aware that deleting any of the lower-level views invalidates dependencies of higher-level views in the hierarchy.

A view usually presents only a subset of the columns and rows in the base table or tables.

Moreover, some view columns do not exist in the underlying base tables. For example, it is possible to present data summaries in a view (for example, an average), which you cannot directly obtain from a base table.

Benefits of Using Views

There are at least four reasons to use views. Views provide:

- A user view of data in the database.
- Security for restricting table access and updates.
- Well-defined, well-tested, high-performance access to data.
- Logical data independence.

Restrictions on Using Views

You can use views as if they were tables in SELECT statements. Views are subject to some restrictions regarding INSERT, UPDATE, MERGE, and DELETE statements. For more information, see “SQL Access to the Data Dictionary”.

SQL Stored Procedures

SQL stored procedures are executed on Teradata Database server space. It is a combination of procedural control statements, SQL statements, and control declarations that provide a procedural interface to Teradata Database.

Using SQL Stored Procedures

Using SQL stored procedures, you can build large and complex database applications. In addition to a set of SQL control statements and condition handling statements, an SQL stored procedure can contain the following:

- Multiple input and output parameters.
- Local variables and cursors.
- SQL DDL, DCL, and DML statements, including dynamic SQL, with a few exceptions.

  Dynamic SQL is a method of invoking an SQL statement by creating and submitting it at runtime from within a stored procedure.

Applications based on SQL stored procedures provide the following benefits. They:

- Reduce network traffic in the client-server environment because stored procedures reside and execute on the server.
- Allow encapsulation and enforcement of business rules on the server, contributing to improved application maintenance.
- Provide better transaction control.
- Provide better security. The data access clause of an SQL stored procedure can restrict access to the database.
- Provide better security by granting the user access to the procedures rather than to the data tables.
- Provide an exception handling mechanism to handle the runtime conditions generated by the application.
- Allow all the SQL and SQL control statements embedded in an SQL stored procedure to be executed by submitting one CALL statement. Nested CALL statements further extend the versatility.

Elements of an SQL Stored Procedure

An SQL stored procedure contains some or all of the following elements.

This element…	Includes…
SQL control statements	nested or non-nested compound statements.
Control declarations	• Condition handlers in DECLARE HANDLER statements for completion and exception conditions. Conditional handlers can be: ◦ CONTINUE or EXIT type. ◦ Defined for a specific SQLSTATE code, the generic exception condition SQLEXCEPTION, or generic completion conditions NOT FOUND and SQLWARNING. • Cursor declarations in DECLARE CURSOR statements or in FOR iteration statements. Cursors can be either updatable or read only type. Cursors can also be result set cursors for returning the result of a SELECT statement executed in the stored procedure to the caller or client applications • Local variable declarations in DECLARE statements.
SQL transaction statements	DDL, DCL, DML, and SELECT statements, including dynamic SQL statements, with a few exceptions.
LOCKING modifiers	with all supported SQL statements except CALL.
Comments	bracketed and simple comments. **Note:** Nested bracketed comments are not allowed.

For more information, see "SQL Stored Procedures as SQL Applications".

External Stored Procedures

External stored procedures are written in the C, C++, or Java programming language, installed on the database, and then executed like stored procedures.

Usage

Here is a synopsis of the steps you take to develop, compile, install, and use external stored procedures:

1. Write, test, and debug the C, C++, or Java code for the procedure.
2. If you are using Java, place the class or classes for the external stored procedure in an archive file (JAR or ZIP) and call the SQLJ.INSTALL_JAR external stored procedure to register the archive file with the database.

3. Use CREATE PROCEDURE or REPLACE PROCEDURE for external stored procedures to create a database object for the external stored procedure.
4. Use GRANT to grant privileges to users who are authorized to use the external stored procedure.
5. Invoke the procedure using the CALL statement.

Macros

The macro database object consists of one or more SQL statements that can be executed by performing a single request. Each time the macro is performed, one or more rows of data may be returned.

SQL Statements Related to Macros

The following table lists the basic SQL statements that you can use with macros.

Use this statement…	To…
CREATE MACRO	incorporate a frequently used SQL statement or series of statements into a macro.
EXECUTE	run a macro. **Note:** A macro can also contain an EXECUTE statement that executes another macro.
DROP MACRO	delete a macro.

Single-User and Multi-User Macros

You can create a macro for your own use, or grant execution authorization to others. For example, your macro might enable a user in another department to perform operations on the data in Teradata Database. When executing the macro, the user need not be aware of the database access, the tables affected, or even the results.

Macro Processing

Regardless of the number of statements in a macro, Teradata Database treats it as a single request. When you execute a macro, the system processes either all of the SQL statements, or processes none of the statements. If a macro fails, the system aborts it, backs out any updates, and returns the database to its original state.

Triggers

The trigger defines events that happen when some other event, called a triggering event, occurs. This database object is essentially a stored SQL statement associated with a table called a *subject* table.

Teradata Database implementation of triggers complies with ANSI SQL specifications and provides extensions.

Triggers execute when any of the following modifies a specified column or columns in the subject table:

- DELETE
- INSERT
- UPDATE

Typically, the stored SQL statements perform a DELETE, INSERT, UPDATE, or MERGE on a table different from the subject table.

Types of Triggers

Teradata Database supports two types of triggers.

This type of trigger…	Fires for each…
statement	statement that modifies the subject table.
row	row modified in the subject table.

When to Fire Triggers

You can specify when triggers fire.

WHEN you specify…	THEN the triggered action…
BEFORE	executes before the completion of the triggering event. As specified in the ANSI/ISO SQL standard, a BEFORE trigger cannot have data changing statements in the triggered action.
AFTER	executes after completion of the triggering event. **Note:** To support stored procedures the CALL statement is supported in the body of an AFTER trigger. Both row and statement triggers can call a stored procedure.

Sometimes a request fires a trigger, which in turn, fires another trigger. Thus the outcome of one triggering event can itself become another trigger. Teradata Database processes and optimizes the triggered and triggering statements in parallel to maximize system performance.

ANSI/ISO-Specified Order

When you specify multiple triggers on a subject table, both BEFORE and AFTER triggers execute in the order in which they were created as determined by the timestamp of each trigger.

Triggers are sorted according to the preceding ANSI/ISO rule, unless you use the Teradata Database extension, ORDER. This extension allows you to specify the order in which the triggers execute, regardless of creation time stamp.

Using Triggers

You can use triggers to do various things:

- Define a trigger on the subject table to ensure that the UPDATE, INSERT, MERGE, and DELETE statements performed to the subject table are propagated to another table.
- Use triggers for auditing. For example, you can define a trigger which causes the INSERT statements in a log table when an employee receives a raise higher than 10%.
- Use a trigger to disallow massive UPDATE, INSERT, MERGE, or DELETE during business hours.
- Use a trigger to set a threshold. For example, you can use triggers to set thresholds for inventory of each item by store, to create a purchase order when the inventory drops below a threshold, or to change a price if the daily volume does not meet expectations.
- Use a trigger to call SQL stored procedures and external stored procedures.

User-Defined Functions

SQL provides a set of useful functions, but they might not satisfy all of the particular requirements you have to process your data.

Teradata Database supports two types of user-defined functions (UDFs) that allow you to extend SQL by writing your own functions:

- SQL UDFs
- External UDFs

SQL UDFs

SQL UDFs allow you to encapsulate regular SQL expressions in functions and then use them like standard SQL functions.

Rather than coding commonly used SQL expressions repeatedly in queries, you can objectize the SQL expressions through SQL UDFs.

Moving complex SQL expressions from queries to SQL UDFs makes the queries more readable and can reduce the client/server network traffic.

External UDFs

External UDFs allow you to write your own functions in the C, C++, or Java programming language, install them on the database, and then use them like standard SQL functions.

You can also install external UDF objects or packages from third-party vendors.

Teradata Database supports three types of external UDFs.

UDF Type	Description
Aggregate	Aggregate functions produce summary results. They differ from scalar functions in that they take grouped sets of relational data, make a pass over each group, and return one result for the group. Some examples of standard SQL aggregate functions are AVG, SUM, MAX, and MIN.
Scalar	Scalar functions take input parameters and return a single value result. Examples of standard SQL scalar functions are CHARACTER_LENGTH, POSITION, and TRIM.
Table	A table function is invoked in the FROM clause of a SELECT statement and returns a table to the statement.

Usage

To create and use an SQL UDF, follow these steps:

1. Use CREATE FUNCTION or REPLACE FUNCTION to define the UDF.
2. Use GRANT to grant privileges to users who are authorized to use the UDF.
3. Call the function.

Here is a synopsis of the steps you take to develop, compile, install, and use an external UDF:

1. Write, test, and debug the C, C++, or Java code for the UDF.
2. If you are using Java, place the class or classes for the UDF in an archive file (JAR or ZIP) and call the SQLJ.INSTALL_JAR external stored procedure to register the archive file with the database.
3. Use CREATE FUNCTION or REPLACE FUNCTION to create a database object for the UDF.
4. Use GRANT to grant privileges to users who are authorized to use the UDF.
5. Call the function.

Related Topics

For more information on …	See …
writing, testing, and debugging source code for an external UDF	*SQL External Routine Programming*
data definition statements related to UDFs, including CREATE FUNCTION and REPLACE FUNCTION	*SQL Data Definition Language*
invoking a table function in the FROM clause of a SELECT statement	*SQL Data Manipulation Language*
archiving and restoring UDFs	*Teradata Archive/Recovery Utility Reference*

User-Defined Methods

A User-Defined Method (UDM) is a special kind of UDF that is associated with a UDT. The term *method* and the acronym UDM are interchangeable.

Teradata Database supports two types of UDMs:

- Instance
- Constructor

Instance Methods

An instance method operates on a specific instance of a distinct or structured UDT. For example, an instance method named *area* might calculate and return the area of an instance of a structured UDT named *circle* that contains attributes *x*, *y*, and *radius*.

Instance methods can also provide transform, ordering, and cast functionality for a distinct or structured UDT. Teradata Database uses this functionality during certain operations involving the UDT.

Constructor Methods

A constructor method initializes an instance of a structured UDT.

A structured UDT can have more than one constructor method, each one providing different initialization options.

User-Defined Types

SQL provides a set of predefined data types, such as INTEGER and VARCHAR, that you can use to store the data that your application uses, but they might not satisfy all of the requirements you have to model your data.

User-defined types (UDTs) allow you to extend SQL by creating your own data types and then using them like predefined data types.

UDT Types

Teradata Database supports distinct and structured UDTs.

UDT Type	Description	Example
Distinct	A UDT that is based on a single predefined data type, such as INTEGER or VARCHAR.	A distinct UDT named euro that is based on a DECIMAL(8,2) data type can store monetary data.
Structure	A UDT that is a collection of one or more fields called attributes, each of which is defined as a predefined data type or other UDT (which allows nesting).	A structured UDT named circle can consist of x-coordinate, y-coordinate, and radius attributes.

Distinct and structured UDTs can define methods that operate on the UDT. For example, a distinct UDT named euro can define a method that converts the value to a US dollar amount. Similarly, a structured UDT named circle can define a method that computes the area of the circle using the radius attribute.

Teradata Database also supports a form of structured UDT called dynamic UDT. Instead of using a CREATE TYPE statement to define the UDT, like you use to define a distinct or structured type, you use the NEW VARIANT_TYPE expression to construct an instance of a dynamic UDT and define the attributes of the UDT at run time.

Unlike distinct and structured UDTs, which can appear almost anywhere that you can specify predefined types, you can only specify a dynamic UDT as the data type of (up to eight) input parameters to external UDFs. The benefit of dynamic UDTs is that they significantly increase the number of input arguments that you can pass in to external UDFs.

Databases and Users

While Teradata Database is a collection of related tables, views, stored procedures, macros, and so on, it also contains databases and users.

A database and a user are *almost* identical in Teradata Database. The major difference is that a user can log on to the system whereas the database cannot.

Creating Databases and Users

When Teradata Database is first installed on a server, one user exists on the system, that is, *User DBC* exists. *User DBC* owns all other databases and users in the system, and initially owns all the space in the entire system.

When Teradata Database is first installed on a server, User DBC is created. User DBC owns (with some exceptions):

- All free space on the system
- All databases and users created after installation

The database administrator manages this user and assigns space from *User DBC* to all other objects (or database and users).

To protect the security of system tables within Teradata Database, the database administrator typically creates *User System Administrator* from *User DBC*. The usual procedure is to assign all database disk space that system tables do not require to *User System Administrator*.

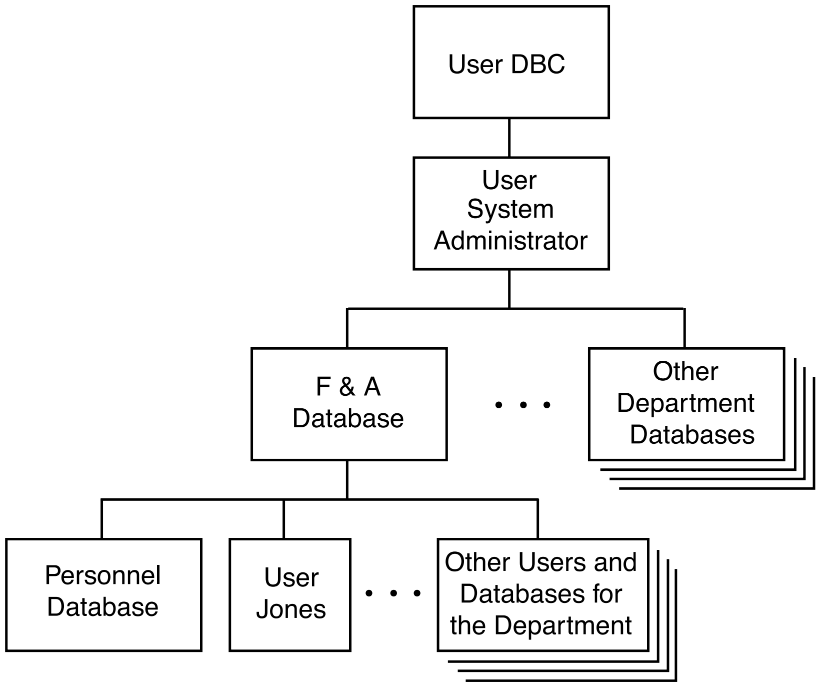
The database administrator then uses this user as a resource from which to allocate space to other databases and users of the system.

For information on how to create databases and users, see *Database Administration*.

Example: Creating a Finance and Administration Database

Consider the following example: the database administrator needs to create a Finance and Administration (F&A) department database with *User Jones* as a supervisory user, that is, as database administrator within the F&A department.

The database administrator first creates *F&A Database* and then allocates space from it to *User Jones* to act as the F&A database administrator. The database administrator also allocates space from F&A to Jones for his personal use and for creating a *Personnel Database*, as well as other databases and other user space allocations. The following figure illustrates the hierarchy implicit in this relationship:


User DBC
User System Administrator
F & A Database
• • •
Other Department Databases
Personnel Database
User Jones
• • •
Other Users and Databases for the Department

F&A Database owns *Personnel Database* and all the other department databases.

F&A Database also owns *User Jones* and all other users within the department.

Because *User DBC* ultimately owns all databases and users, it is the final owner of all the databases and user space belonging to the organization.

This hierarchical ownership structure provides an owner of a database or user space with complete control over the security of owned data. An owner can archive the database or control access to it by granting or revoking privileges on it.

Note:
There can only one immediate owner of a database or user.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database books.

If you want to learn more about…	See…
Tables	• *SQL Data Definition Language* • *SQL Data Manipulation Language*
Queue tables	• *SQL Data Definition Language* • *SQL Data Manipulation Language*

If you want to learn more about…	See…
Views	• *SQL Data Definition Language* • *Data Dictionary*
SQL stored procedures	• *SQL Data Definition Language* • *SQL Stored Procedures and Embedded SQL*
SQL external stored procedures	• *SQL Data Definition Language* • *SQL Stored Procedures and Embedded SQL*
Macros	*SQL Data Definition Language*
Triggers	*SQL Data Definition Language*
User-Defined Functions	• *SQL Data Definition Language* • *SQL External Routine Programming*
User-Defined Methods	• *SQL Data Definition Language* • *SQL External Routine Programming*
User-Defined Types	• *SQL Data Definition Language* • *SQL External Routine Programming*
Databases and Users	*Database Administration*

CHAPTER 8

SQL

SQL and Teradata SQL

SQL is the American National Standards Institute (ANSI) and International Organization for Standardization (ISO) language for relational database management. All application programming facilities ultimately make queries against Teradata Database using SQL.

Teradata Database conforms closely to the ANSI/ISO SQL standard while also supporting unique extensions that enable users to take full advantage of the efficiency of Teradata parallelism. This comprehensive language is called Teradata SQL. You can run transactions in either Teradata or ANSI mode. For more information about Teradata SQL and how Teradata implements ANSI/ISO SQL, see *SQL Fundamentals*.

Using SQL

SQL has the advantage of being the most commonly used language for relational database management systems. Because of this, both the data structures in Teradata Database and the commands for manipulating those structures are controlled using SQL. In addition, all applications, including those written in a client language with embedded SQL, macros, and ad hoc SQL queries, are written and executed using the same set of instructions and syntax.

Other database management systems use different languages for data definition and data manipulation and may not permit ad-hoc queries of the database. Teradata Database lets you use one language to define, query, and update your data.

Types of SQL Statements

The SQL language allows you, using SQL statements, to define database objects, to define user access to those objects, and to manipulate the data stored.

These functions form the principal functional families of SQL statements:

- Data Definition Language (DDL) statements
- Data Control Language (DCL) statements
- Data Manipulation Language (DML) statements

In addition, SQL provides HELP and SHOW statements that provide help about database object definitions, sessions and statistics, the EXPLAIN request modifier, SQL statement syntax, as well as displaying the SQL used to create tables.

The following sections contain information about the functional families of Teradata SQL.

Data Definition Language Statements

You use DDL statements to define the structure and instances of a database. DDL provides statements for the definition and description of database objects.

The following table lists some *basic* DDL statements. The list is not exhaustive.

Statement	Action
CREATE	Defines a new database object, such as a database, user, table, view, trigger, index, macro, stored procedure, user-defined type, user-defined function, or user-defined macro, depending on the object of the CREATE request.
DROP	Removes a database object, such as a database, user, table, view, trigger, index, macro, stored procedure, user-defined type, user-defined function, user-defined method, depending on the object of the DROP request.
ALTER	Changes, for example, a table, column, referential constraint, trigger, or index.
ALTER PROCEDURE	Recompiles an external stored procedure.
MODIFY	Changes a database or user definition.
RENAME	Changes, for example, the names of tables, triggers, views, stored procedures, and macros.
REPLACE	Replaces, for example, macros, triggers, stored procedures, and views
SET	Specifies, for example, time zones, the collation or character set for a session.
COLLECT	Collects optimizer or QCD statistics on, for example, a column, group of columns, index.
DATABASE	Specifies a default database.
COMMENT	Inserts or retrieves a text comment for a database object.

Successful execution of a DDL statement automatically creates, updates, or removes entries in the Data Dictionary. For information about the contents of the Data Dictionary, see Chapter 13: "The Data Dictionary."

Data Control Language Statements

You use DCL statements to grant and revoke access to database objects and change ownership of those objects from one user or database to another. The results of DCL statement processing also are recorded in the Data Dictionary.

The following table lists some *basic* DCL statements. The list is not exhaustive.

Statement	Action
GRANT/REVOKE	Controls privileges of the users on an object.

Statement	Action
GRANT LOGON/ REVOKE LOGON	Controls logon privileges to a host (client) or host group (if the special security user is enabled).
GIVE	Gives a database object to another database object.

Data Manipulation Language Statements

You use DML statements to manipulate and process database values. You can insert new rows into a table, update one or more values in stored rows, or delete a row.

The following table list some *basic* DML statements. The list is not exhaustive.

Statement	Action
CHECKPOINT	Checkpoints a journal. CHECKPOINT is a statement that defines a recovery point in the journal that can later be used to restore the table contents to its state at a point in time. This can be useful if, for example, the table contents become incorrect due to hardware failure or an operational error.
DELETE	Removes a row (or rows) from a table.
ECHO	Echoes a string or command to a client.
INSERT	Inserts new rows into a table. For more information about a special case of INSERT, see Atomic Upsert later in this table.
MERGE	Combines both UPDATE and INSERT in a single SQL statement. Supports primary index operations only, similar to Atomic Upsert but with fewer constraints.
These statements: • ABORT • ROLLBACK • COMMIT • BEGIN TRANSACTION • END TRANSACTION	Allows you to manage transactions.
SELECT	Returns specific row data in the form of a result table.
UPDATE	Modifies data in one or more rows of a table. For more information about a special case of UPDATE, see Atomic Upsert later in this table.
	Atomic Upsert The upsert form of the UPDATE DML statement is a Teradata Database extension of the ANSI/ISO SQL standard designed to enhance the performance of TPump utility by allowing the

Statement	Action
	statement to support atomic upsert. For more information about how TPump operates, see Teradata Parallel Data Pump. This feature allows Teradata TPump and all other CLIv2-, ODBC-, and JDBC-based applications to perform single-row upsert operations using an optimally efficient single-pass strategy. This single-pass upsert is called atomic to emphasize that its component UPDATE and INSERT SQL statements are grouped together and performed as a single, or atomic, SQL statement.

SQL Statement Syntax

A typical SQL request consists of the following:

- A statement keyword
- One or more column names
- A database name
- A table name
- One or more optional clauses introduced by keywords

For example, in the following single-statement request, the statement keyword is SELECT:

```
SELECT deptno, name, salary
FROM personnel.employee
WHERE deptno IN(100, 500)
ORDER BY deptno, name
;
```

The select list and FROM clause for this statement is made up of the following names:

- Deptno, name, and salary (the column names)
- Personnel (the database name)
- Employee (the table name)

The search condition, or WHERE clause, is introduced by the keyword WHERE, as in:

```
WHERE deptno IN(100, 500)
```

The sort ordering, or ORDER BY clause, is introduced by the keywords ORDER BY, as in:

```
ORDER BY deptno, name
```

Statement Execution

Teradata Database offers the following ways to invoke an executable statement:

- Interactively from a terminal
- Embedded within an application program

- Dynamically created within an embedded application
- Embedded within a stored procedure or external stored procedure
- Dynamically created within an SQL stored procedure
- Via a trigger
- Embedded within a macro

Statement Punctuation

You can use punctuation to separate or identify the parts of an SQL statement.

This syntax element…	Named…	Performs this function in a SQL statement…
.	period	separates database names from table names and table names from a particular column name (for example, personnel.employee.deptno).
,	comma	separates and distinguishes column names in the select list, or column names or parameters in an optional clause.
'	apostrophe	delimits the boundaries of character string constants.
( )	left and right parentheses	groups expressions or defines the limits of a phrase.
;	semicolon	separates statements in multi-statement requests and terminates requests submitted via certain utilities such as BTEQ.
"	double quotation marks	identifies user names that might otherwise conflict with SQL reserved words or that would not be valid names in the absence of the double quotation marks.
:	colon	prefixes reference parameters or client system variables.

To include an apostrophe or show possession in a title, double the apostrophes.

The SELECT Statement

SELECT is probably the most frequently used SQL statement. It specifies the table columns from which to obtain the data you want, the corresponding database (if different from the current default database), and the table or tables that you need to reference within that database.

The SELECT statement further specifies how, in what format, and in what order the system returns the set of result data.

You can use the following options, lists, and clauses with the SELECT statement to request data from Teradata Database. The list is not exhaustive.

- DISTINCT option

- FROM clause
- WHERE clause, including subqueries
- GROUP BY clause
- HAVING clause
- QUALIFY clause
- ORDER BY clause
    - CASESPECIFIC option
    - International sort orders
- WITH clause
- Query expressions and set operators

Another variation is the SELECT INTO statement, which is used in embedded SQL and stored procedures. This statement selects at most one row from a table and assigns the values in that row to host variables in embedded SQL or to local variables or parameters in Teradata Database stored procedures.

SELECT Statement and Set Operators

The SELECT statement can use the set operators UNION, INTERSECT, and MINUS/EXCEPT. These set operators allow you to manipulate the answers to two or more queries by combining the results of each query into a single result set.

You can use the set operators within, for example, the following operations:

- View definitions
- Derived tables
- Subqueries

SELECT Statement and Joins

A SELECT statement can reference data in two or more tables and the relational join combines the data from the referenced tables.

In this way, the SELECT statement defines a join of specified tables to retrieve data more efficiently than without defining a join of tables.

You can specify both inner joins and outer joins:

- An inner join selects data from two or more tables that meet specific join conditions. Each source must be named and the join condition, that is the common relationship among the tables to be joined, can be on an ON clause or a WHERE clause.
- The outer join is an extension of the inner join that includes rows that qualify for a simple inner join, as well as a specified set of rows that do not match the join conditions expressed by the query.

SQL Data Types

Data Types

Every data value belongs to an SQL data type. For example, when you define a column in a CREATE TABLE statement, you must specify the data type of the column. Teradata Database supports the following categories of data types. For a complete list of supported data types and detailed information about each data type, see *SQL Data Types and Literals*.

Data Type Category	Description	Data Type Examples
ARRAY/ VARRAY	An ARRAY data type is used for storing and accessing multidimensional data. The ARRAY data type can store many values of the same specific data type in a sequential or matrix-like format.	• One-dimensional (1-D) ARRAY • Multidimensional (n-D) ARRAY
Byte	Byte data types store raw data as logical bit streams. These data types are stored in the client system format and are not translated by Teradata Database. The data is transmitted directly from the memory of the client system.	• BYTE • VARBYTE • BLOB (Binary Large Object)
Character	Character data types represent characters that belong to a given character set. These data types represent character data.	• CHAR • VARCHAR • CLOB (Character Large Object)
DATASET	A complex data type that represents self-describing data stored in a format conforming to some schema.	AVRO
DateTime	DateTime data types represent date, time, and timestamp values.	• DATE • TIME • TIMESTAMP • TIME WITH TIME ZONE • TIMESTAMP WITH TIME ZONE
Geospatial	Geospatial data types represent geographic information and provides a way for applications that manage, analyze, and display geographic information to interface with Teradata Database.	• ST_Geometry • MBR
Interval	Interval data types represent a span of time. For example, an interval value can represent a time span that includes a number of years, months, days, hours, minutes, or seconds.	• INTERVAL YEAR • INTERVAL YEAR TO MONTH • INTERVAL MONTH • INTERVAL DAY • INTERVAL DAY TO HOUR

Data Type Category	Description	Data Type Examples
		• INTERVAL DAY TO MINUTE • INTERVAL DAY TO SECOND • INTERVAL HOUR • INTERVAL HOUR TO MINUTE • INTERVAL HOUR TO SECOND • INTERVAL MINUTE • INTERVAL MINUTE TO SECOND • INTERVAL SECOND
JSON	The JSON data type represents data that is in JSON (JavaScript Object Notation) format.	JSON
Numeric	Numeric data types represent a numeric value that is an exact numeric number (integer or decimal) or an approximate numeric number (floating point).	• BYTEINT • SMALLINT • INTEGER • BIGINT • DECIMAL/NUMERIC • FLOAT/REAL/DOUBLE PRECISION • NUMBER
Parameter	Parameter data types are data types that can be used only with input or result parameters in a function, method, stored procedure, or external stored procedure.	• TD_ANYTYPE • VARIANT_TYPE
Period	A Period data type represents a time period, where a period is a set of contiguous time granules that extends from a beginning bound up to but not including an ending bound.	• PERIOD(DATE) • PERIOD(TIME) • PERIOD(TIME WITH TIME ZONE) • PERIOD(TIMESTAMP) • PERIOD(TIMESTAMP WITH TIME ZONE)
UDT	UDT (User-Defined Type) data types are custom data types that you define to model the structure and behavior of the data used by your applications.	• Distinct • Structured
XML	The XML data type represents XML content. The data is stored in a compact binary form that preserves the information set of the XML document, including the hierarchy information and type information derived from XML validation.	XML

For detailed information on data types and the related functions, procedures, methods, and operators that operate on these types, see the following books:

SQL Data Types and Literals

SQL Geospatial Types

Teradata JSON

Teradata XML

SQL Functions, Operators, Expressions, and Predicates

Data Type Phrase

A data type phrase does the following:

- Determines how data is stored on Teradata Database.
- Specifies how data is presented to the user.

Data Type Attributes

You can define the attributes of a data value.

Core data type attributes that Teradata Database supports include the following.

Data Type Attribute	Description
NOT NULL	Specifies that the fields of a column must contain a value; they cannot be null.
UPPERCASE	Specifies that character data for a column is stored as uppercase.
[NOT] CASESPECIFIC	Specifies case for character data comparisons and collations.
FORMAT	Controls the display of expressions, column data, and conversions between data types.
TITLE	Defines a heading for displayed or printed results that is different from the column name, which is used by default.
AS	Assigns a temporary name to an expression.
NAMED	Assigns a temporary name to an expression. NAMED is a Teradata extension to the ANSI standard. For ANSI compliance, use AS instead of NAMED.
DEFAULT	Specifies that a user-defined default value is to be inserted in the field when a value is not specified for a column in an INSERT statement.
WITH DEFAULT	Specifies that a system-defined default value is to be inserted in the field when a value is not specified for a column in an INSERT statement.

Data Type Attribute	Description
WITH TIME ZONE	Used with the TIME or TIMESTAMP data type to specify a TIME or TIMESTAMP value with a displacement from UTC as defined for the system.
CHARACTER SET	Specifies the server character set for a character column.

For a complete list of supported data type attributes and detailed information about each attribute, see *SQL Data Types and Literals*.

Teradata Database Recursive Query

A recursive query is a named query expression that references itself in its definition. The self-referencing capability gives the user a simple way to search a table using iterative self-join and set operations.

The recursive query feature benefits the user by reducing the complexity of the queries and allowing a certain class of queries to execute more efficiently.

Recursive queries are implemented using the WITH RECURSIVE clause in the statement and the RECURSIVE clause in the CREATE VIEW statement.

SQL Functions

The control statements of SQL stored procedures make SQL a computationally complete (that is, procedural) language. You can write your own UDFs and external stored procedures in C, C++, or Java to define what you want.

Procedural languages contain functions that perform complex operations. The usual SQL statements do not support many functions. However, to reduce the reliance on ancillary application code, SQL does support the following standard functions:

- Scalar
- Aggregate
- Ordered analytical

In addition, you can create scalar, aggregate, and table functions to meet specific needs.

Scalar Functions

A scalar function works on input parameters to create a result.

When it is part of an expression, the function is invoked as needed whenever expressions are evaluated for an SQL statement. When a function completes, its result is used by the expression in which the function was referenced.

For example, the following request returns the current date plus 13 years.

```
SELECT ADD_MONTHS (CURRENT_DATE, 12*13);
```

The following request returns the date 6 months ago.

```
SELECT ADD_MONTHS (CURRENT_DATE, -6);
```

Aggregate Functions

Sometimes the information you want can only be derived from data in a set of rows, instead of individual rows.

Aggregate functions produce results from sets of relational data that you have grouped (optionally) using a GROUP BY or ORDER BY clause. Aggregate functions process each set and produce one result for each set.

The following table lists a few examples of aggregate functions.

The function…	Returns the…
AVG	arithmetic average of the values in a specified column.
COUNT	number of qualified rows.
MAX	maximum column value for the specified column.
MIN	minimum column value for the specified column.
SUM	arithmetic sum of a specified column.

Ordered Analytical Functions

Ordered analytical functions work over a range of data for a particular set of rows in some specific order to produce a result for each row in the set.

Like aggregate functions, ordered analytical functions are called for each item in a set. But unlike an aggregate function, an ordered analytical function produces a result for each detail item.

Ordered analytical functions allow you to perform sophisticated data mining on the information in your databases to get the answers to questions that SQL otherwise cannot provide.

The following table lists two examples of ordered analytical functions.

The following function…	Returns the…
AVG	arithmetic average of all values in the specified expression for each row in the group. The OVER() phrase must be specified to make AVG an ordered analytical function.
RANK	ordered ranking of rows based on the value of the column being ranked.

Cursors

Traditional application development languages cannot deal with result tables without some kind of intermediary mechanism because SQL is a set-oriented language. The intermediary mechanism is the cursor.

A cursor is a pointer that the application program uses to move through a result table.

You declare a cursor for a SELECT request, and then open the named cursor. The act of opening the cursor executes the SQL request.

You use the FETCH... INTO... statement to individually fetch and write the rows into host variables. The application can then use the host variables to do computations.

Teradata Preprocessor2 uses cursors to mark or tag the first row accessed by an SQL query. Preprocessor2 then increments the cursor as needed.

SQL stored procedures use:

- Cursors to fetch one result row at a time and then to execute SQL and SQL control statements as required for each row. Local variables or parameters from the stored procedure can be used for computations.
- Result set cursors to return the result of a SELECT statement executed in the stored procedure to the caller of the stored procedure or the client application.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database books.

If you want to learn more about…	See…
SQL	*SQL Fundamentals*
Types of SQL Statements	• *SQL Data Control Language* • *SQL Data Definition Language* • *SQL Data Manipulation Language*
The SELECT Statement	• *SQL Data Manipulation Language* • *SQL Functions, Operators, Expressions, and Predicates* • *SQL Fundamentals*
SQL Data Types	*SQL Data Types and Literals*
Teradata Database Recursive Query	*SQL Fundamentals*
SQL Functions	*SQL Functions, Operators, Expressions, and Predicates*
Cursors	• *SQL Data Definition Language* • *SQL Stored Procedures and Embedded SQL*
Session Modes	• *SQL Fundamentals* • *SQL Request and Transaction Processing*

CHAPTER 9
SQL Application Development

SQL Application Development Overview

This chapter describes the tools used to develop applications for Teradata Database and the interfaces used to establish communications between the applications and Teradata Database.

Client Applications

Client applications can use the following APIs to communicate with Teradata Database:

- .NET Data Provider for Teradata
- Java Database Connectivity (JDBC)
- Open database Connectivity (ODBC)

For information on these APIs, see "Workstation Attachment Methods".

Embedded SQL Applications

When you write applications using embedded SQL, you insert SQL requests into your application program, which must be written in one of the supported programming languages shown in "Supported Languages and Platforms".

Because third-generation application development languages do not have facilities for dealing with results sets, embedded SQL contains extensions to executable SQL that permit declarations.

Embedded SQL declarations include:

- Code to encapsulate the SQL from the application language
- Cursor definition and manipulation

A cursor is a pointer device you use to read through a results table one record/row at a time. For more information about cursors, see "Cursors".

Using Embedded SQL

The client application languages that support embedded SQL are all compiled languages. SQL is not defined for any of them. For this reason, you must precompile your embedded SQL code to translate the SQL into native code before you can compile the source using a native compiler.

The precompiler tool is called Preprocessor2 (PP2), and you use it to:

- Read your application source code to look for the defined SQL code fragments.
- Interpret the intent of the code after it isolates all the SQL code in the application and translates it into Call-Level Interface (CLI) calls.

- Comment out all the SQL source.

The output of the precompiler is native language source code with CLI calls substituted for the SQL source. After the precompiler generates the output, you can process the converted source code with the native language compiler. For information about Call-Level Interface communications interface, see Chapter 6: “Client Communication with Teradata Database.”

Supported Languages and Platforms

Preprocessor2 supports the following application development languages on the specified platforms.

Application Development Language	Platform
C	• IBM mainframe clients • UNIX® operating system clients and some other workstation clients
COBOL	• IBM mainframe clients • Some workstation clients
PL/I	IBM mainframes

Macros as SQL Applications

Teradata Database macros are SQL statements that the server stores and executes. Macros provide an easy way to execute frequently used SQL operations. Macros are particularly useful for enforcing data integrity rules, providing data security, and improving performance.

SQL Used to Create a Macro

You use the CREATE MACRO statement to create Teradata Database macros.

For example, suppose you want to define a macro for adding new employees to the Employee table and incrementing the EmpCount field in the Department table. The CREATE MACRO statement looks like this:

```
CREATE MACRO NewEmp (name VARCHAR(12),
                     number INTEGER NOT NULL,
                     dept INTEGER DEFAULT 100
                    )
AS (INSERT INTO Employee (Name,
                          EmpNo,
                          DeptNo
                         )
    VALUES (name,
           number,
            dept
           )
    ;
    UPDATE Department
    SET EmpCount=EmpCount+1
```

```
        WHERE DeptNo=dept
        ;
       )
    ;
```

This macro defines parameters that users must fill in each time they execute the macro. A leading colon (:) indicates a reference to a parameter within the macro.

Macro Usage

The following example shows how to use the NewEmp macro to insert data into the Employee and Department tables.

The information to be inserted is the name, employee number, and department number for employee H. Goldsmith. The EXECUTE macro statement looks like this:

```
      EXECUTE NewEmp (‘Goldsmith H’, 10015, 600);
```

SQL Used to Modify a Macro

The following example shows how to modify a macro. Suppose you want to change the NewEmp macro so that the default department number is 300 instead of 100. The REPLACE MACRO statement looks like this:

```
    REPLACE MACRO NewEmp (name VARCHAR(12),
                          number INTEGER NOT NULL,
                          dept INTEGER DEFAULT 300)
    AS (INSERT INTO Employee (Name,
                              EmpNo,
                              DeptNo)
        VALUES (name,
                number,
                dept)
        ;
        UPDATE Department
        SET EmpCount=EmpCount+1
        WHERE DeptNo=dept
        ;
       )
    ;
```

SQL Used to Delete a Macro

The example that follows shows how to delete a macro. Suppose you want to drop the NewEmp macro from the database. The DROP MACRO statement looks like this:

```
   DROP MACRO NewEmp;
```

SQL Stored Procedures as SQL Applications

SQL stored procedures are database applications created by combining SQL control statements with other SQL elements and condition handlers. They provide a procedural interface to Teradata Database and many of the same benefits as embedded SQL.

SQL stored procedures conform to the ANSI/ISO SQL standard with some exceptions.

SQL Used to Create Stored Procedures

Teradata SQL supports creating, modifying, dropping, renaming, and controlling privileges of stored procedures through DDL and DCL statements.

You can create or replace an external stored procedure through the COMPILE command in Basic Teradata Query (BTEQ), BTEQ for Microsoft Windows systems (BTEQWIN), Teradata Studio, and SQL Assistant. You must specify a source file as input for the COMPILE command.

Stored procedures do not need to be compiled, but external stored procedures do.

You can also create or modify a stored procedures using the CREATE PROCEDURE or REPLACE PROCEDURE statement from CLIv2, ODBC, and JDBC applications.

SQL Stored Procedure Example

Assume you want to create an SQL stored procedure named NewProc that you can use to add new employees to the Employee table and retrieve the department name of the department to which the employee belongs. You can also report an error, in case the row that you are trying to insert already exists, and handle that error condition.

The following SQL stored procedure definition includes nested, labeled compound statements. The compound statement labeled L3 is nested within the outer compound statement L1. Note that the compound statement labeled L2 is the handler action clause of the condition handler.

This SQL stored procedure defines parameters that must be filled in each time it is called (executed).

```
CREATE PROCEDURE NewProc (IN name CHAR(12),
              IN num INTEGER,
              IN dept INTEGER,
              OUT dname CHAR(10),
              INOUT p1 VARCHAR(30))
L1: BEGIN
   DECLARE CONTINUE HANDLER FOR SQLSTATE value '23505'
    L2: BEGIN
        SET p1='Duplicate Row'
        ;
    END L2;
        L3: BEGIN
        INSERT INTO Employee (EmpName, EmpNo, DeptNo)
        VALUES (name, num, dept)
    ;

        SELECT DeptName
        INTO dname FROM Department
        WHERE DeptNo = dept;
        IF SQLCODE <> 0 THEN LEAVE L3;
        ...
        END L3
         ;
END L1
;
```

SQL Used to Execute a Stored Procedure

After compiling an external stored procedure, procedures are stored as objects in Teradata Database. You can execute stored procedures from Teradata Database client products using the SQL CALL statement. Arguments for all input (IN or INOUT) parameters of the stored procedure must be submitted with the CALL statement.

BTEQ and other Teradata Database client products support stored procedure execution and DDL. These include:

- JDBC
- ODBC
- CLIv2
- PP2
- Teradata SQL Assistant
- Teradata Studio
- BTEQWIN (BTEQ for Windows)

DDL Statements with Stored Procedures

You can use the following DDL statements with stored procedures. The list is not exhaustive.

Use This Statement...	To...
CREATE PROCEDURE	direct the stored procedure compiler to create a procedure from the SQL statements in the remainder of the statement text.
ALTER PROCEDURE	direct the stored procedure compiler to recompile a stored procedure created in an earlier version of Teradata Database without executing SHOW PROCEDURE and REPLACE PROCEDURE statements.
DROP PROCEDURE	drop a stored procedure.
RENAME PROCEDURE	rename a procedure.
REPLACE PROCEDURE	direct the stored procedure compiler to replace the definition of an existing stored procedure. If the specified stored procedure does not exist, create a new procedure by that name from the SQL statements in the remainder of the source text.
HELP PROCEDURE ... ATTRIBUTES	view all the parameters and parameter attributes of a procedure, or the creation time attributes of a procedure.

Use This Statement…	To…
HELP 'SPL'	display a list of all DDL and control statements associated with stored procedures.
HELP 'SPL' *command_name*	display help about the command you have named.
SHOW PROCEDURE	view the current definition (source text) of a procedure. The text is returned in the same format as defined by the creator.

The EXPLAIN Request Modifier

Teradata SQL supplies a very powerful EXPLAIN request modifier that allows you to see the execution plan of a query.

The EXPLAIN request modifier not only explains how a request will be processed, but provides an estimate of the number of rows involved as well as the performance impact of the request. Teradata Database supports EXPLAIN request modifiers with detailed Optimizer information including, for example, cost estimates for Insert, Update, Upsert, Merge, and Delete steps, as well as spool size estimates.

How EXPLAIN Works

The EXPLAIN request modifier that precedes any SQL request causes Teradata Database to display the execution plan for that request. The request itself is not submitted for execution.

When you perform an EXPLAIN against any SQL request, that request is parsed and optimized. The access and join plans generated by the Optimizer are returned in the form of a text file that explains the (possibly parallel) steps used in the execution of the request. Also included is the relative cost required to complete the request given the statistics with which the Optimizer had to work. If the statistics are not reasonably accurate, the cost estimate may not be accurate.

Benefits of Using EXPLAIN

EXPLAIN helps you to evaluate complex queries and to develop alternative, more efficient, processing strategies. You may be able to get a better plan by collecting more statistics on more columns, or by defining additional indexes. Your knowledge of the actual demographics information may allow you to identify row count estimates that seem badly wrong, and help to pinpoint areas where additional statistics would be helpful.

Simple EXPLAIN Example

The EXPLAIN example shown below results from joining tables with the following table definitions:

```
CREATE TABLE customer
(c_custkey INTEGER,
c_name CHAR(26),
c_address VARCHAR(41),
c_nationkey INTEGER,
c_phone CHAR(16),
c_acctbal DECIMAL(13,2),
c_mktsegment CHAR(21),
c_comment VARCHAR(127))
UNIQUE PRIMARY INDEX( c_custkey );

 *** Table has been created.
 *** Total elapsed time was 1 second.
+---------+---------+---------+---------+---------+---------+---------
CREATE TABLE orders
(o_orderkey INTEGER NOT NULL,
o_custkey INTEGER,
o_orderstatus CHAR(1),
o_totalprice DECIMAL(13,2) NOT NULL,
o_orderdate DATE FORMAT 'yyyy-mm-dd' NOT NULL,
o_orderpriority CHAR(21),
o_clerk CHAR(16),
o_shippriority INTEGER,
o_commment VARCHAR(79))
UNIQUE PRIMARY INDEX(o_orderkey);

 *** Table has been created.
 *** Total elapsed time was 1 second.
+---------+---------+---------+---------+---------+---------+---------
CREATE TABLE lineitem
(l_orderkey INTEGER NOT NULL,
l_partkey INTEGER NOT NULL,
l_suppkey INTEGER,
l_linenumber INTEGER,
l_quantity INTEGER NOT NULL,
l_extendedprice DECIMAL(13,2) NOT NULL,
l_discount DECIMAL(13,2),
l_tax DECIMAL(13,2),
l_returnflag CHAR(1),
l_linestatus CHAR(1),
l_shipdate DATE FORMAT 'yyyy-mm-dd',
l_commitdate DATE FORMAT 'yyyy-mm-dd',
l_receiptdate DATE FORMAT 'yyyy-mm-dd',
l_shipinstruct VARCHAR(25),
l_shipmode VARCHAR(10),
l_comment VARCHAR(44))
PRIMARY INDEX( l_orderkey );

 *** Table has been created.
 *** Total elapsed time was 1 second.
+---------+---------+---------+---------+---------+---------+---------
collect stats orders index (o_orderkey) values (0,0,1,10,1,1000000,1000000)
;

 *** Update completed. One row changed.
 *** Total elapsed time was 1 second.
+---------+---------+---------+---------+---------+---------+---------
collect stats lineitem index (l_orderkey) values (0,0,1,10,1,500000,1000000
);

 *** Update completed. One row changed.
 *** Total elapsed time was 1 second.
```

The following statement defines a join index on these tables:

```
CREATE JOIN INDEX order_join_line AS
SELECT ( l_orderkey, o_orderdate, o_custkey, o_totalprice ),
( l_partkey, l_quantity, l_extendedprice, l_shipdate )
FROM lineitem
LEFT JOIN orders ON l_orderkey = o_orderkey
ORDER BY o_orderdate
PRIMARY INDEX (l_orderkey);

 *** Index has been created.
 *** Total elapsed time was 1 second
```

The following EXPLAIN shows that the Optimizer used the newly created join index, order_join_line:

```
EXPLAIN
SELECT o_orderdate, o_custkey, l_partkey, l_quantity,
l_extendedprice
FROM lineitem , orders
WHERE l_orderkey = o_orderkey;

 *** Help information returned. 14 rows.
 *** Total elapsed time was 1 second.

Explanation
-----------------------------------------------------------------------
  1) First, we lock a distinct EXPLAINSAMPLE."pseudo table" for read on
     a RowHash to prevent global deadlock for
     EXPLAINSAMPLE.ORDER_JOIN_LINE.
  2) Next, we lock EXPLAINSAMPLE.ORDER_JOIN_LINE for read.
  3) We do an all-AMPs RETRIEVE step from EXPLAINSAMPLE.ORDER_JOIN_LINE
     by way of an all-rows scan with a condition of ("NOT
     (EXPLAINSAMPLE.ORDER_JOIN_LINE.o_orderdate IS NULL)") into Spool 1
     (group_amps), which is built locally on the AMPs.  The size of
     Spool 1 is estimated with high confidence to be 36 rows.  The
     estimated time for this step is 0.01 seconds.
  4) Finally, we send out an END TRANSACTION step to all AMPs involved
     in processing the request.
  -> The contents of Spool 1 are sent back to the user as the result of
     statement 1.  The total estimated time is 0.01 seconds.
```

The following statement drops the join index named order_join_line:

```
drop join index order_join_line;

 *** Index has been dropped.
 *** Total elapsed time was 1 second.
```

Third-Party Development

Teradata Database supports many third-party software products. The two general components of supported products include those of the transparency series and the native interface products.

Compatible Third-Party Software Products

Many third-party, interactive query products operate in conjunction with Teradata Database, permitting queries formulated in a native query language to access Teradata Database.

The list of supported third-party products changes frequently. For a current list, contact your Teradata sales office.

Workload Management Application Programming Interface

Workload management API consists of interfaces to PM/APIs and open APIs. You can use these interfaces to:

- Monitor system and session-level activities.
- Monitor Teradata Active System Management (ASM) activity.
- Track system usage and manage task priorities.
- Retrieve data from the Priority Scheduler.

For more information about the APIs, see *Application Programming Reference*.

PM/API

PM/APIs provide access to PMPC routines resident in Teradata Database. The PMPC subsystem is available through a logon partition called MONITOR, using a specialized PM/API subset of CLIv2 or Teradata JDBC Driver. PM/APIs have the following features:

- CLIv2 or Teradata JDBC Driver data is acquired in near real time, with less overhead and minimal possibility of being blocked. These capabilities allow frequent in-process performance analysis.
- CLIv2 request saves the raw data in an in-memory buffer where a client application program can easily retrieve the data for real-time analysis or importing into custom reports. The Teradata JDBC Driver returns the data as a JDBC ResultSet where a client application program can easily retrieve the data.
- CLIv2 or Teradata JDBC Driver request provides access to data that the resource usage does not. For example, session-level resource usage data, and data on application locks and which application is being blocked.

Note:
Using PM/APIs may not be the right choice for all performance monitoring requirements. Standard performance monitoring tools and reports, such as resource usage reports, may be sufficient for your needs.

Open API

The workload management open API provides an SQL interface to the PMPC subsystem and Teradata Database system through user-defined functions, embedded services functions, and external stored procedures. Most of the SQL interfaces available to the PMPC subsystem provide similar functionality to the CLIv2 or Teradata JDBC Driver requests.

Note:

Most open APIs do not follow transaction rules. If a transaction calls a UDF or external stored procedure and the transaction rolls back, the action of the UDF or external stored procedure is not rolled back. However, the external stored procedures that update the TDWM database must follow the transaction rules. If a transaction calls one of these external stored procedures and the transaction is aborted, the update will be rolled back.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database and Teradata Tools and Utilities books.

IF you want to learn more about…	See…
SQL Applications, including: • Client Applications • Embedded SQL Applications	• *SQL Stored Procedures and Embedded SQL* • *Teradata Preprocessor2 for Embedded SQL Programmer Guide*
Macros as SQL Applications	• *SQL Fundamentals* • *SQL Data Definition Language* • *SQL Data Manipulation Language*
Teradata Database SQL stored procedures as SQL applications	• *SQL Fundamentals* • *SQL Stored Procedures and Embedded SQL*
The EXPLAIN Request Modifier	• *SQL Data Manipulation Language* • *SQL Request and Transaction Processing*
Third-Party Development, including: • Workload Management APIs	• *Application Programming Reference*

CHAPTER 10

Scripting and Language Support

About Scripting Capabilities

Application developers can write functions and scripts in almost any programming language, install and execute them inside Teradata Database, and run them in parallel for efficient analysis of large data sets.

Customers can:

- Write scripts in languages such as Ruby, Python, and Perl
- Install the scripts, database server configuration files, or flat files using a Teradata-supplied external stored procedure
- Use the SCRIPT table operator to execute the scripts on all AMPs
- Use Teradata Database rows as input to the scripts

External stored procedures INSTALL_FILE, REPLACE_FILE, REMOVE_FILE, and REDISTRIBUTE_FILE manage the user-installed files on all nodes.

Benefits

- Faster prototype creation.
- Enables customers to use the fastest, simplest solution for any task they need to perform.
- Eliminates unnecessary data movement and improves performance.
- Application developers can decide where in the architecture different parts of an application run.

For More Information

For more information on running scripts in Teradata Database, see *SQL Functions, Operators, Expressions, and Predicates*.

About Using R with Teradata Database

Data scientists use statistical models to predict future events based on current and historical data. R is an open source programming language for statistical modeling and graphics. Users can write scripts and table operators in R to run inside Teradata Database. Both methods access data stored in Teradata Database and process the data using R data analysis tools. For more information on R table operators, see *SQL External Routine Programming*. For more information on R scripts, see *SQL Functions, Operators, Expressions, and Predicates*.

CHAPTER 11

Data Distribution and Data Access Methods

Overview

This chapter describes how Teradata Database distributes and organizes data. It discusses data access methods, data normalization, and referential integrity.

Teradata Database Indexes

An index is a physical mechanism used to store and access the rows of a table. Indexes on tables in a relational database function much like indexes in books, they speed up information retrieval.

In general, Teradata Database uses indexes to:

- Distribute data rows.
- Locate data rows.
- Improve performance.

  Indexed access is usually more efficient than searching all rows of a table.
- Ensure uniqueness of the index values.

  Only one row of a table can have a particular value in the column or columns defined as a unique index.

Teradata Database supports the following types of indexes:

- Primary
- Partitioned Primary
- Secondary
- Join
- Hash
- Special indexes for referential integrity

These indexes are discussed in the following sections.

Primary Indexes

You can create a table with a Unique Primary Index (UPI), a Non-Unique Primary Index (NUPI), or No Primary Index (NoPI).

IF you create a table with...	THEN...
a NUPI	the PI is a column, or columns, that may have duplicate values.

IF you create a table with...	THEN...
a UPI	the PI is a column, or columns, that has no duplicate values.
NoPI	there is no PI column and rows are not hashed based on any column values.

Primary Indexes and Data Distribution

Unique Primary Indexes (UPIs) guarantee uniform distribution of table rows.

Nonunique Primary Indexes (NUPIs) can cause skewed data. While not a guarantor of uniform row distribution, the degree of uniqueness of the index will determine the degree of uniformity of the distribution. Because all rows with the same PI value are distributed to the same AMP, columns with a small number of distinct values that are repeated frequently do not make good PI candidates.

The most efficient access methods to get data in a table is through the PI. For this reason, choosing a PI should take the following design goal into consideration: choosing a PI that gives good distribution of data across the AMPs must be balanced against choosing a PI that reflects the most common usage pattern of the table.

If you do not explicitly specify a primary index when a table is created, Teradata Database uses the first column as the nonunique primary index by default.

User tables can also be created explicitly without a primary index. These NoPI tables are used for special purposes. NoPI tables are typically used as staging tables to allow faster data loading.

Primary Key

A Primary Key (PK), a term that comes from data modeling, defines a column, or columns, that uniquely identify a row in a table. Because it is used for identification, a PK cannot be null. There must be something in that column, or columns, that uniquely identify it. Moreover, PK values should not be changed. Historical information, as well as relationships with others tables, may be lost if a PK is changed or re-used.

A PK is a logical relational database concept. It may or may not be the best column, or columns, to choose as a PI for a table.

Foreign Key

A Foreign Key (FK) identifies table relationships. They model the relationship between data values across tables. Relational databases, like Teradata Database, permit data values to associate across more than one table.

Thus each FK a table may have must exist somewhere as a PK. That is, there must be referential integrity between FKs and PKs.

Relationships Between Primary Indexes and Primary Keys

The following table describes some of the relationships between PKs and PIs.

Primary Key	Primary Index
Identifies a row uniquely.	Distributes rows.

Primary Key	Primary Index
Does not imply access path.	Defines most common access path.
Must be unique.	May be unique or nonunique.
May not be null.	May be null.
Causes a Unique Primary Index (UPI) or Unique Secondary Index (USI) to be created.	N/A
Constraint used to ensure referential integrity.	Physical access mechanism.
Required by Teradata Database only if referential integrity checks are to be performed.	Defined for most production tables. Some staging tables may not have a primary index (NoPI table).
• If Teradata Database performs referential integrity checks, then the column limit is 64. • If Teradata Database performs no referential integrity checks, then there is no arbitrary column limit.	64-column limit.
Values should not be changed if you want to maintain data integrity and preserve historical relations among tables.	Values can be changed.

The columns chosen for the UPI of a table are frequently the same columns identified as the PK during the data modeling process, but no hard-and-fast rule makes this so. In fact, physical database design considerations often lead to a choice of columns other than those of the primary key for the PI of a table.

Secondary Indexes

Secondary Indexes (SIs) allow access to information in a table by alternate paths, and can improve performance by avoiding full table scans.

Although SIs add to table overhead, in terms of disk space and maintenance, you can drop and recreate SIs as needed.

SIs:

- Do not affect the distribution of rows across AMPs.
- Can be unique or nonunique.
- Can be created for complex data types, such as geospatial data.
- Are used by the Optimizer when the indexes can improve query performance.
- Can be useful for NoPI tables.

Comparison of Primary and Secondary Indexes

The following table provides a brief comparison of PI and SI features.

Feature	Primary	Secondary
Can be unique or nonunique	Both	Both
Affects row distribution	Yes	No
Create and drop dynamically	No	Yes
Improves access	Yes	Yes
Create using multiple data types	Yes	Yes
Requires separate physical structure	No	Yes, a subtable
Requires extra processing overhead	No	Yes

Join Indexes

A Join Index (JI) is an indexing structure containing columns from one or more base tables.

Some queries can be satisfied by examining only the JI when all referenced columns are stored in the index. Such queries are said to be *covered* by the JI. Other queries may use the JI to qualify a few rows, then refer to the base tables to obtain requested columns that are not stored in the JI. Such queries are said to be *partially-covered* by the index.

Because Teradata Database supports multitable, partially-covering JIs, all types of JIs, except the aggregate JI, can be joined to their base tables to retrieve columns that are referenced by a query but are not stored in the JI. Aggregate JIs can be defined for commonly-used aggregation queries.

Much like SIs, JIs impose additional processing on insert and delete operations and update operations which change the value of columns stored in the JI. The performance trade-off considerations are similar to those for SIs.

Single-table Join Indexes

Join indexes are similar to base tables in that they support a primary index, which can be used for direct access to one or a few rows.

A single-table JI is a an index structure that contains rows from only a single-table. This type of structure has been found to be very useful by Teradata Database users because it provides an alternative approach (primary index) to directly accessing data.

Multitable Join Indexes

When queries frequently request a particular join, it may be beneficial to predefine the join with a multitable JI. The Optimizer can use the predefined join instead of performing the same join repetitively.

Aggregate Join Indexes

Aggregate operations calculate a single value from individual column values in several rows of a table. Sums and averages calculated from sets of column values are examples of aggregate operations. If the same aggregate operation is frequently performed on the same columns, an aggregate JI can provide improved query performance because Teradata Database can use the aggregate index to satisfy queries, rather than repeating the aggregate calculations for every query. You can define aggregate JIs on one or more tables.

Sparse Join Indexes

Indexes include a subset of the columns of one or more tables. Typically, indexes include the column values from all rows in the table. Sparse join indexes further limit the index to include only a subset of the table rows, in addition to a subset of table columns. Sparse JIs can include one or more tables, and can also be aggregate JIs.

If queries frequently are limited to a subset of the rows of a table, for example, rows with a specific value in one column, a sparse JI that includes only those rows can improve performance by providing a more limited data set to be queried.

Comparison of Index Types

Teradata Database does not require or allow users to explicitly dictate how indexes should be used for a particular query. The Optimizer costs all of the reasonable alternatives and selects the one that is estimated to be the least expensive.

The object of any query plan is to return accurate results as quickly as possible. Therefore, the Optimizer uses an index or indexes only if the index speeds up query processing. In some cases, the Optimizer processes the query without using any index.

Optimizer index selection for a query plan:

- Can have a direct impact on overall Teradata Database performance.
- Is not always a straightforward process.
- Is based partly on usage expectations.

The following table assumes execution of a simple SELECT statement and explains the strengths and weaknesses of some of the various indexing methods.

This access method…	Has the following strengths…	And the following possible drawbacks…
Unique Primary Index (UPI)	• is the most efficient access method when the SQL statement contains the PI value • involves one AMP and one row • requires no spool file (for a simple SELECT) • can obtain the most granular locks	none, in the context of a SELECT statement specifying a PI value. However, a poorly chosen PI can cause poor overall performance in a large workload.
Nonunique Primary Index (NUPI)	• provides efficient access when the SQL statement contains the PI value • involves one AMP • can obtain granular locks	• may slow down INSERTs for a SET table with no USIs. • may decrease the efficiency of SELECTs containing the PI value when some values are repeated in many rows.

This access method…	Has the following strengths…	And the following possible drawbacks…
	• may not require a spool file as long as the number of rows returned is small	
Unique Secondary Index (USI)	• provides efficient access when the SQL statement contains the USI values, and you do not specify PI values • involves two AMPs and one row • requires no spool file (for a simple SELECT)	requires additional overhead for INSERT, UPDATE, MERGE, and DELETE statements.
Nonunique Secondary Index (NUSI)	• provides efficient access when the number of rows per value in the table is relatively small • involves all AMPS and probably multiple rows • provides access using information that may be more readily available than a UPI value, such as employee last name, compared to an employee number • may require a spool file	• requires additional overhead for INSERT, UPDATE, MERGE, and DELETE statements. • will not be used by the Optimizer if the number of data blocks accessed is a significant percentage of the data blocks in the table because the Optimizer will determine that a full table scan is cheaper.
Full table scan	• accesses each row only once • provides access using any arbitrary set of column conditions	• examines every row. • usually requires a spool file possibly as large as the base table.
Multitable join index (JI)	• can eliminate the need to perform certain joins and aggregates repetitively • may be able to satisfy a query without referencing the base tables • can have a different PI from that of the base table • can replace an NUSI or a USI	• requires additional overhead for INSERT, UPDATE, MERGE, and DELETE statements for any of the base tables that contribute to the multitable JI. • usually is not suitable for data in tables subjected to a large number of daily INSERT, UPDATE, MERGE, and DELETE statements. • imposes some restrictions on operations performed on the base table.
Single-table join index (JI) or hash index	• can isolate frequently used columns (or their aggregates for	• requires additional overhead for INSERT, UPDATE, MERGE, and DELETE statements.

This access method…	Has the following strengths…	And the following possible drawbacks…
	JIs only) from those that are seldom used • can reduce number of physical I/Os when only commonly used columns are referenced • can have a different PI from that of the base table	• imposes some restrictions on operations performed on the base table.
Sparse join index (JI)	• can be stored in less space than an ordinary JI • reduces the additional overhead associated with INSERT, UPDATE, MERGE, and DELETE statements to the base table when compared with an ordinary JI • can exclude common values that occur in many rows to help ensure that the Optimizer chooses to use the JI to access less common values	• requires additional overhead for INSERT, UPDATE, MERGE, and DELETE statements to the base table. • imposes some restrictions on operations performed on the base table.

For More Information

For more information about indexing, types of indexes, and creating indexes, see *Database Design*, *SQL Data Definition Language*, and *SQL Geospatial Types*.

Partitioned Tables

Partitioning stores related groups of data in physical proximity to improve the performance of queries that are likely to require that data. To satisfy a query, Teradata Database can employ an optimization called “partition elimination” to limit data searches to only those partitions containing data relevant to the query.

A row-partitioned table assigns rows to a particular partition within an AMP based on a user-defined partitioning expression that defines how the rows should be grouped for storage (horizontal partitioning). The partitioning expression is defined when a table is created or altered.

The rows of a row-partitioned table are assigned to an appropriate partition based on the value of the partitioning expression.

Tables without primary indexes can also be partitioned by column (vertical partitioning). Whereas row partitioning allows sets of rows to be stored in separate partitions based on a partitioning expression, column partitioning allows sets of columns (including just a single column) to be stored in separate partitions. Like row partitioning, column partitioning can improve performance for some types of queries by allowing for partition elimination, whereby only the column data relevant to a particular query is searched during the processing of that query.

Multilevel Partitioned Tables

A table or join index may be column partitioned, row partitioned, or both, by using multilevel partitioning. Multilevel partitioning allows each partition to be subpartitioned. Each level must define at least two partitions.

For More Information

For more information about row and column partitioning, see *Database Design* and *SQL Data Definition Language*.

Hashing

Teradata Database uses hashing to distribute data for tables with a PI to disk storage and uses indexes to access the data.

Because the architecture of Teradata Database is massively parallel, it requires an efficient means of distributing and retrieving its data. That efficient method is hashing. Virtually all Teradata Database indexes are based on (or partially based on) row hash values rather than table column values.

For PIs, Teradata Database obtains a row hash by hashing the values of the PI columns. The row hash and a sequence number, which is assigned to distinguish between rows with the same row hash within a table, are collectively called a row identifier and uniquely identify each row in a table. A partition identifier is also part of the row identifier in the case of partitioned tables. For more information on partitioned tables, see “Partitioned Tables”.

For SIs, Teradata Database computes a hash value using the hash of the values of the SI columns. This value is used for access when an SI value is specified in the SQL. The SI subtable records the hash value for the SI, the actual value of the index columns (for synonym resolution), and a list of primary index row identifiers for the table being indexed.

Identity Columns

Identity columns are columns that have unique values for every row in the table. Because those values confer uniqueness on every row, the system can use the values to identify each row in a table. When a column is defined as an identity column, Teradata Database automatically generates a unique numeric value for the column in every row that is added to the table.

Identity columns can be used to generate unique values for UPI, USI, and primary key columns. However, creating a UPI from combinations of frequently queried columns is preferable to adding identity columns to tables to serve this function.

For more information about indexes, see “Teradata Database Indexes”.

Normalization

Normalization is the process of reducing a complex database schema into a simple, stable one. Generally this process involves removing redundant attributes, keys, and relationships from the conceptual data model.

Normal Forms

Normalization theory is constructed around the concept of *normal forms* that define a system of constraints. If a relation meets the constraints of a particular normal form, we say that relation is in normal form.

By definition, a relational database is always normalized to first normal form, because the column values are always *atomic*. That is, a column can contain one and only one value or null.

But to simply leave it at that invites a number of problems including redundancy and potential update anomalies. The higher normal forms were developed to correct those problems.

First, Second, and Third Normal Forms

First, second, and third normal forms are stepping stones to the Boyce-Codd normal form and, when appropriate, the higher normal forms.

First Normal Form

First normal form (1NF) is definitive of a relational database. If we are to consider a database relational, then all relations in the database are in 1NF.

We say a relation is in 1NF if all fields within that relation are atomic. We sometimes refer to this concept as the elimination of repeating groups from a relation. Furthermore, first normal form allows no hierarchies of data values.

Second Normal Form

Second normal form (2NF) deals with the elimination of circular dependencies from a relation. We say a relation is in 2NF if it is in 1NF and if every non-key attribute is fully dependent on the entire Primary Key.

A non-key attribute is any attribute that is not part of the Primary Key for the relation.

Third Normal Form

Third normal form (3NF) deals with the elimination of non-key attributes that do not describe the Primary Key.

For a relation to be in 3NF, the relationship between any two non-Primary Key columns, or groups of columns, in a relation must *not* be one-to-one in either direction.

We say attributes are mutually independent if none of them is functionally dependent on any combination of the others. This mutual independence ensures that we can update individual attributes without any danger of affecting any other attribute in a row.

The following list of benefits summarizes the advantages of implementing a normalized logical model in 3NF.

- Greater number of relations
- More PI choices
- Optimal distribution of data
- Fewer full table scans

Referential Integrity

Traditional referential integrity is the concept of relationships between tables, based on the definition of a primary key and a foreign key. The concept states that a row cannot exist in a table with a value (not null) for a *referencing* column if an equal value does not exist in a *referenced* column.

Using referential integrity, you can specify columns within a *referencing* table that are foreign keys for columns in some other *referenced* table. You must define referenced columns as either primary key columns or unique columns.

Referential integrity is a reliable mechanism that prevents accidental database inconsistencies when you perform inserts, merges, updates, and deletes.

Referential Integrity Terminology

We use the following terms to explain the referential integrity concept.

Term	Definition
Parent Table	The table referred to by a Child table. Also called the "referenced table."
Child Table	A table in which the referential constraints are defined. Also called the "referencing table."
Parent Key	A candidate key in the parent table.
Primary Key	With respect to referential integrity, a primary key is a parent table column set that is referred to by a foreign key column set in a child table.
Foreign Key	With respect to referential integrity, a foreign key is a child table column set that refers to a primary key column set in a parent table.

Referencing (Child) Table

We call the referencing table the Child table, and we call the specified Child table columns the referencing columns. Referencing columns should be of the same number and have the same data type as the referenced table key.

Referenced (Parent) Table

A Child table must have a parent table, and the referenced table is referred to as the Parent table. The parent key columns are the referenced columns.

Importance of Referential Integrity

Referential integrity is important, because it keeps you from introducing errors into your database. Suppose you have an Order Parts table like the following.

Order Number	Part Number	Quantity
PK		Not Null

Order Number	Part Number	Quantity
FK	FK	
1	1	110
1	2	275
2	1	152

Part number and order number, each foreign keys in this relation, also form the composite primary key.

Suppose you were to delete the row defined by the primary key value 1 in the PART NUMBER table. The foreign key for the first and third rows in the ORDER PART table would now be inconsistent, because there would be no row in the PART NUMBER table with a primary key of 1 to support it. Such a situation shows a loss of referential integrity.

Teradata Database provides referential integrity to prevent this from happening. If you try to delete a row from the PART NUMBER table for which you have specified referential integrity, the database management system will not allow you to remove the row if the part number is referenced in child tables.

Besides data integrity and data consistency, referential integrity provides these benefits.

Benefit	Description
Increases development productivity	You do not need to code SQL statements to enforce referential integrity constraints because Teradata Database automatically enforces referential integrity.
Requires fewer written programs	All update activities are programmed to ensure that referential integrity constraints are not violated, because Teradata Database enforces referential integrity in all environments. Additional programs are not required.
Allows optimizations	Referential integrity allows optimizations to occur, such as join elimination.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database books:

IF you want to learn more about…	See…
Teradata Database Indexes	• *Database Design* • *SQL Fundamentals*
Primary Indexes	• *Database Design* • *Database Administration* • *SQL Data Definition Language - Syntax and Examples* • *SQL Fundamentals*

IF you want to learn more about…	See…
No Primary Index (NoPI) Tables	• *Database Design* • *SQL Data Definition Language - Syntax and Examples* • *SQL Data Manipulation Language* • *SQL Request and Transaction Processing*
Row and Column Partitioning	• *Database Design* • *SQL Data Definition Language - Syntax and Examples*
Secondary Indexes Join Indexes	• *Database Design* • *Database Administration* • *SQL Data Definition Language - Syntax and Examples* • *SQL Fundamentals* • *SQL Request and Transaction Processing*
Hash Indexes	• *Database Design* • *SQL Data Definition Language - Syntax and Examples*
Index Specification	*SQL Data Manipulation Language*
Hashing	• *Database Design* • *SQL Data Definition Language - Syntax and Examples* • *SQL Request and Transaction Processing*
Identity column	*SQL Data Definition Language - Syntax and Examples*
Normalization Referential integrity	*Database Design*

CHAPTER 12

Concurrency Control and Transaction Recovery

Overview

This chapter describes the concurrency control in relational database management systems and how to use transaction journaling to recover lost data, or to restore an inconsistent database to a consistent state.

About Concurrency Control

Concurrency control prevents concurrently running processes from improperly inserting, deleting, or updating the same data. A system maintains concurrency control through two mechanisms:

- Transactions
- Locks

Transactions

Transactions are a mandatory facility for maintaining the integrity of a database while running multiple, concurrent operations.

Definition of a Transaction

A transaction is a logical unit of work and the unit of recovery. The requests nested within a transaction must either all happen or not happen at all. Transactions are atomic. A partial transaction cannot exist.

Definition of Serializability

A set of transactions is serializable if the set produces the same result as some arbitrary serial execution of those same transactions for arbitrary input.

A set of transactions is correct only if it is serializable. The Two-Phase Locking (2PL) protocol ensures the serializability of transactions.

The phases of the Two-Phase Locking protocol are described in the following table:

In the…	A transaction must…
growing phase	acquire a lock on an object before operating on it.

In the…	A transaction must…
	The Teradata Optimizer requests locks as close to the beginning of the transaction as possible.
shrinking phase	release the previously acquired lock and never acquire any more locks after it has released a lock. Lock release is an all-or-none operation. Once acquired, locks are not released until the transaction has committed or is completely rolled back.

ANSI Mode Transactions

All ANSI transactions are implicitly opened. Either of the following events opens an ANSI transaction:

- Execution of the first SQL request in a session.
- Execution of the first request following the close of a previous transaction.

Transactions close when the application performs a COMMIT, ROLLBACK, or ABORT request.

When the transaction contains a DDL statement, including DATABASE and SET SESSION, which are considered DDL statements in this context, the statement must be the last request in the transaction other than the transaction closing statement.

A session executing under ANSI transaction semantics allows neither the BEGIN TRANSACTION statement, the END TRANSACTION statement, nor the two-phase commit protocol. When an application submits these statements in ANSI mode, the database software generates an error.

In ANSI mode, the system rolls back the entire transaction if the current request:

- Results in a deadlock.
- Performs a DDL statement that aborts.
- Executes an explicit ROLLBACK or ABORT statement.

Teradata Database accepts the ABORT and ROLLBACK statements in ANSI mode, including conditional forms of those statements. If the system detects an error for either a single or multistatement request, it only rolls back that request, and the transaction remains open, except in special circumstances.

Application-initiated, asynchronous aborts also cause full transaction rollback in ANSI mode.

Teradata Mode Transactions

Teradata mode transactions can be either implicit or explicit. An explicit, or user-generated, transaction is a single set of BEGIN TRANSACTION/END TRANSACTION statements surrounding one or more requests. All other requests are implicit transactions.

Consider the following transaction:

```
        BEGIN TRANSACTION;
        DELETE FROM Employee
        WHERE Name = ‘Smith T’;
        UPDATE Department
        SET EmpCount=EmpCount-1
        WHERE DeptNo=500;
        END TRANSACTION;
```

If an error occurs during the processing of either the DELETE or UPDATE statement within the BEGIN TRANSACTION and END TRANSACTION statements, the system restores both Employee and Department tables to the states at which they were before the transaction began. If an error occurs during a Teradata transaction, then the system rolls back the *entire* transaction.

Locks

A lock is a means of controlling access to some resource. Teradata Database locks different types of resources in several different ways.

Overview of Teradata Database Locking

Most locks used on Teradata Database resources are obtained automatically. Users can upgrade the severity of a lock with the LOCKING request modifier but can not downgrade the severity of a lock. The data integrity requirement of a request determines the type of lock that the system uses.

A request for a resource that is locked by another user is queued (in the case of a conflicting lock level) until the process using the resource releases its lock on that resource. A user can specify that the request be aborted if the lock cannot be obtained immediately.

Recovery and Transactions

Recovery is a process by which an inconsistent database is brought back to a consistent state.

Transactions play the critical role in this process because they are used to “play back” (using the term in its most general sense) a series of updates to the database, either taking it back to some earlier state or bringing it forward to a current state.

System and Media Recovery

The following sections describe the behavior of Teradata Database when it encounters different types of errors or failures.

System Restarts

Unscheduled restarts occur for one of the following reasons:

- AMP or disk failure
- Software failure
- Disk parity error

Failures and errors affect all software recovery in the same way. Hardware failures take the affected component offline and it remains offline until repaired or replaced.

Transaction Recovery

Two types of automatic transaction recovery can occur:

- Single transaction recovery
- Database recovery

The following table details what happens when the two automatic recovery mechanisms take place.

This recovery type…	Happens when Teradata Database…
single transaction	aborts a single transaction because of: • Transaction deadlock • User error • User-initiated abort command • An inconsistent data table Single transaction recovery uses the transient journal to effect its data restoration.
database	performs a restart for one of the following reasons: • Hardware failure • Software failure • User command

Down AMP Recovery

When an AMP fails to come online during system recovery, which is done using the down AMP Recovery Journal, Teradata Database continues to process requests using fallback data. When the down AMP comes back online, down AMP recovery procedures begin to bring the data for the AMP up-to-date as follows.

IF there are…	THEN the AMP recovers…
a large number of rows to be processed	offline.
only a few rows to be processed	online.

After all updates are made, we consider the AMP to be fully recovered.

Down Subtable Recovery

Teradata Database can isolate some file system errors to a specific data or index subtable, or to a contiguous range of rows ("region") in a data or index subtable. In these cases, Teradata Database marks only the affected subtable or region down. This improves system performance and availability by allowing transactions that do not require access to the down subtable or rows to proceed, without causing a database crash that would require a system restart.

In-progress transactions that require the down subtable or region are aborted. Subsequent transactions that require access to the down subtable or region are not allowed until the problem is fixed.

Two-Phase Commit Protocol

Two-phase commit (2PC) is a protocol for assuring update consistency across distributed databases in which each participant in the transaction commit operation votes to either commit or abort the changes. Participants wait before committing a change until they know that all participants can commit.

A participant is a "database manager" that performs some work on behalf of the transaction and that commits or aborts changes to the database. A participant can also be a coordinator of participants at a lower level.

By voting to commit, a participant guarantees that it can either commit or roll back its part of the transaction, even if it crashes before receiving the result of the vote.

The 2PC protocol allows the development of Customer Information Control System (CICS) and Information Management System (IMS) applications that can update one or more Teradata Database databases or databases, or both under some other DBMS in a synchronized manner. The result is that all updates requested in a defined unit of work will either succeed or fail.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database and Teradata Tools and Utilities books.

IF you want to learn more about…	See…
Host Utility Locks	• *SQL Request and Transaction Processing* • *Teradata Archive/Recovery Utility Reference* • *Database Administration*
Locks	• *SQL Request and Transaction Processing* • *Database Administration*
System and Media Recovery	• *Database Administration* • *Utilities*
Transactions	*SQL Request and Transaction Processing*
ANSI Mode Transactions	
Teradata Mode Transactions	
Two-phase Commit Protocol	• *Teradata Director Program Reference* • *IBM CICS Interface for Teradata Reference* • *IBM IMS/DC Interface for Teradata Reference*

CHAPTER 13
The Data Dictionary

Overview

The Teradata Database Data Dictionary is composed of tables and views that reside in the system user named DBC. The tables are reserved for use by the system and contain metadata about the objects in the system, privileges, system events, and system usage. The views provide access to the information in the tables. The tables contain current definitions, control information, and general information about the following:

- Authorization
- Accounts
- Character sets
- Columns
- Constraints
- Databases
- Disk Space
- End Users
- Events
- External stored procedures
- Indexes
- JAR and ZIP archive files
- Logs
- Macros
- Privileges
- Profiles
- Resource usage
- Roles
- Rules
- Sessions and session attributes
- Statistics
- Stored Procedures
- Tables
- Translations
- Triggers
- User-defined functions
- User-defined methods
- User-defined types
- Views

The Data Dictionary stores object definitions as well as details about objects.

Object	Details Stored
Table	• Location, identification, version • Database name, table name, creator name, and user names of all owners in the hierarchy • Each column in the table, including column name, data type, length, and phrases • User/creator access privileges • Indexes • Constraints • Table backup and protection (including fallback and permanent journaling status) • Date and time the object was created
Database	• Database name, creator name, owner name, and account name • Space allocation (if any) including: ◦ Permanent

Object	Details Stored
	◦ Spool ◦ Temporary • Number of fallback tables • Collation type • Creation timestamp • Date and time the database was last altered and the name that altered it • Role and profile names • Revision numbers for the UDF library and any XSP libraries by Application Category
View or macro	• View or macro text • Creation time attributes • User and creator access privileges
Stored procedure	• Creation time attributes • Parameters, including parameter name, parameter type, data type, and default format • User and creator access privileges
External stored procedure	• C/C++ source code and object code if its language is not Java • External stored procedure name • External name • Data types of the parameters • Source file language • Data accessing characteristic • Parameter passing convention • Execution protection mode • Character type • Platform type
JAR	• Java object code for the JAR • JAR name • External name • Platform type • Revision number
Java external stored procedure	• Java external stored procedure name • External file reference • Data types of the parameters • Source file language • Data accessing characteristic • Parameter passing convention • Execution protection mode • Character type

Object	Details Stored
	• Platform type
Java user-defined function (UDF)	• Function call name • Specific name • External name • Data types of the parameters • Function class • Source file language • Data accessing characteristic • Parameter passing convention • Deterministic characteristic • Null-call characteristic • Execution protection mode • Character type • Platform type
Trigger	• IDs of the: ◦ Table ◦ Trigger ◦ Database and subject table database ◦ User who created the trigger ◦ User who last updated the trigger • Timestamp for the last update • Indexes • Trigger name and: ◦ Whether the trigger is enabled ◦ The event that fires the trigger ◦ The order in which triggers fire • Default character set • Creation text and time stamp • Overflow text, that is, trigger text that exceeds a specified limit • Fallback tables
UDF	• C source code and object code if its language is not Java • Function call name • Specific name • External name • Data types of the parameters • Function class • Source file language • Data accessing characteristic • Parameter passing convention • Deterministic characteristic

Object	Details Stored
	• Null-call characteristic • Execution protection mode • Character type • Platform type
User-defined method (UDM)	• C source code and object code • Function call name • Specific name • External name • Data types of the parameters • Function class • Source file language • Data accessing characteristic • Parameter passing convention • Deterministic characteristic • Null-call characteristic • Execution protection mode • Character type • Platform type
User-defined type (UDT)	DBC.UDTInfo - one entry per UDT • Type name • Type kind (Distinct or Structured) • Whether the type is instantiable • Default transform group (name) • Ordering form (full ordering or equals only - distinct and structured types are always full) • Ordering category (map or relative - distinct and structured types are always map) • Ordering routine ID • Cast count DBC.UDTCast - one entry per cast for a UDT • Whether cast is implicit assignment • Cast routine ID DBC.UDFInfo - one entry for the UDT's auto-generated default constructor; entries are the same as for a regular (C/C++) UDF DBC.UDTTransform - one entry for the UDT's transform • Default transform group name • ToSQL routine ID • FromSQL routine ID
User	• User name, creator name, and owner name

Object	Details Stored
	• Password string and password change date • Space allocation, including: ◦ Permanent ◦ Spool ◦ Temporary • Default account, database, collation, character type, and date form • Creation timestamp • Name and time stamp of the last alteration made to the user • Role and profile name

Data Dictionary Views

You can examine the information about the system tables in database DBC directly or through a series of views. Typically, you use views to obtain information on the objects in the Data Dictionary rather than querying the actual tables, which can be very large. The database administrator controls who has access to views.

Users of Data Dictionary Views

Some Data Dictionary views may be restricted to special types of users, while others are accessible by all users. The database administrator controls access to views by granting privileges. The following table defines the information needs of various types of users.

This type of user…	Needs to know…
End	• Objects to which the user has access • Types of access available to the user • The privileges the user has granted to other users
Supervisory	• How to create and organize databases • How to monitor space usage • How to define new users • How to allocate access privileges • How to create indexes • How to perform archiving operations
Database administrator	• Performance • Status and statistics • Errors • Accounting

This type of user…	Needs to know…
Security administrator	• Access logging rules generated by the execution of BEGIN LOGGING statements • Results of access checking events, logged as specified by the access logging rules
Operations and Recovery Control	Archive and recovery activities

SQL Access to the Data Dictionary

Every time you log on to Teradata Database, perform an SQL query, or type a password, you are using the Data Dictionary.

For security and data integrity reasons, the only SQL DML command you can use on the Data Dictionary is the SELECT statement. You cannot use the INSERT, UPDATE, MERGE, or DELETE SQL statements to alter the Data Dictionary, except for some Data Dictionary tables, such as the AccLogTbl table or the EventLog table.

The Data Dictionary contains Unicode system views. However, to maintain backwards compatibility, Teradata provides compatibility system views to translate object names to the Kanji1/Latin. For more information on Unicode and compatibility system views, see *Data Dictionary* and *Database Administration*.

You can use SELECT to examine any view in the Data Dictionary to which your database administrator has granted you access. For example, if you need to access information in the Personnel database, then you can query the DBC.DatabasesV view as shown:

```
SELECT Databasename,
         Creatorname,
         Ownername,
         Permspace
  FROM DBC.DatabasesV
  WHERE Databasename=’Personnel’
  ;
```

The query above produces a report like this:

```
Databasename        Creatorname        Ownername        Permspace
Personnel           Jones              Jones            1,000,000
```

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database book.

IF you want to learn more about…	See…
Data Dictionary	*Data Dictionary*
Data Dictionary Views	

IF you want to learn more about…	See…
SQL Access to the Data Dictionary	

CHAPTER 14

International Language Support

International Language Support Overview

A character set (sometimes called a code page) is simply a way of representing characters on a computer. There are many ways to represent characters on a computer, so there are many character sets in use today.

Because different characters are needed for different languages, character sets are often designed to support a particular language. Even for the same language, many different character sets may exist.

When computers or computer applications exchange character data, it is important that they either use the same character set or properly convert the data from one character set to the other during the transfer process. Otherwise, the data received by one machine may no longer have the same meaning as it had before the transfer. The same issue exists for numeric data, but there are fewer ways used to represent numbers, so it is not as big a problem.

A character set has a repertoire of characters it supports, a representation for character strings, and an implied collation based on this representation.

Character Representation

Representing strings of characters is essentially a two-step process:

- Creating a mapping between each character required and an integer.
- Devising an encoding scheme for placing a sequence of numbers into memory.

The simplest systems map the required characters to small integers between 0 and 255, and encode sequences of characters as sequences of bytes with the appropriate numeric values.

Representing characters for repertoires that require more than 256 characters, such as Japanese, Chinese, and Korean, requires more complex schemes.

External and Internal Character Sets

Client systems communicate with Teradata Database using their own *external* format for numbers and character strings.

Teradata Database converts numbers and strings to its own *internal* format when importing the data, and converts numbers and strings back to the appropriate form for the client when exporting the data.

This allows data to be exchanged between mutually incompatible client data formats. Take for example, mainframe-attached clients using EBCDIC-based character sets and workstation-attached clients using ASCII-based character sets. Both clients can access and modify the same data in Teradata Database.

Character Data Translation

Teradata Database translates the characters:

- Received from a client system into a form suitable for storage and processing on the server.
- Returned to a client into a form suitable for storage, display, printing, and processing on that client.

Thus, the server communicates with each client in the language of that client without the need for additional software. It is essential, for this process to work properly, for the database to be informed of the correct character set used by each client.

What Teradata Database Supports

Teradata Database supports many external *client* character sets and allows each application to choose the internal *server* character set best suited to each column of character data in Teradata Database. Because of automatic translation, it is normally the repertoire of characters required that determines the server character set to use.

No matter which server character set you chose, communication with the client is always in the client character set (also known as the session charset).

Teradata Database Character Data Storage

Teradata Database uses internal server character sets to represent user data and data in the Data Dictionary within the system.

Internal Server Character Sets

Server character sets include:

- LATIN
- UNICODE
- KANJISJIS
- GRAPHIC
- KANJI1

Notice:

KANJI1 support is deprecated. KANJI1 is not allowed as a default character set. The system changes the KANJI1 default character set to the UNICODE character set. Creation of new KANJI1 objects is highly restricted. Although many KANJI1 queries and applications may continue to operate, sites using KANJI1 should convert to another character set as soon as possible.

User Data

User data refers to character data that you store in a character data type column on Teradata Database.

Object Names in the Data Dictionary

Teradata Database stores a variety of character data in the Data Dictionary, including the names of the following objects:

• Tables	• Triggers	• User-defined functions
• Databases	• JAR and ZIP archive files	• User-defined types
• Users	• Join indexes	• User-defined methods
• Columns	• Hash indexes	• Authorizations
• Views	• Stored procedures	
• Macros	• External stored procedures	

Language Support Modes

During system initialization (sysinit) the database administrator can optimize Teradata Database for one of two language support modes:

- Standard
- Japanese

The language support mode determines the:

- Character set that Teradata Database uses to store system dictionary data.
- Default character set for user data.

IF you enable this language support mode …	THEN Teradata Database stores object names using this character set …	AND stores character data other than object names using this character set …
Standard	UNICODE For backward compatibility, object names are processed internally as LATIN	LATIN.
Japanese	UNICODE For backward compatibility, object names are processed internally as KANJI1	UNICODE.

Overriding the Default Character Set for User Data

The language support mode sets the default server character set for a user if the DEFAULT CHARACTER SET clause does not appear in the CREATE USER or MODIFY USER statement.

To override the default character set for a user, you can use the DEFAULT CHARACTER SET clause in a CREATE USER statement. You can also specify a server character set for a character column when you created the table.

Standard Language Support Mode

Although the Data Dictionary uses UNICODE to store object names in standard language support mode and Japanese language support mode, the permissible range of characters in object names depends on the language support mode.

For user data, the default server character set is LATIN.

LATIN Character Set

Standard language support provides Teradata Database internal coding for the entire set of printable characters from the ISO 8859-1 (Latin1) and ISO 8859-15 (Latin9) standard, including diacritical marks such as ä, ñ, Ÿ, Œ, and œ, though the Z with caron in Latin9 is not supported. ASCII control characters are also supported for the standard language set.

Note:

ASCII, as used here, represents the characters that can be stored as the LATIN server character set, referred to as Teradata LATIN. EBCDIC is Teradata Database extended ASCII mapped to the corresponding EBCDIC code points.

Compatible Languages

The LATIN server character set that Teradata Database uses in Standard language support mode is sufficient for you to use client character sets that support the international languages listed in the following table.

International Languages That are Compatible with Standard Language Support			
Albanian	English	Germanic	Portuguese
Basque	Estonian	Greenlandic	Rhaeto-Romantic
Breton	Faroese	Icelandic	Romance
Catalonian	Finnish	Irish Gaelic (new orthography)	Samoan
Celtic	French	Italian	Scottish Gaelic
Cornish	Frisian	Latin	Spanish
Danish	Galician	Luxemburgish	Swahili
Dutch	German	Norwegian	Swedish

Japanese Language Support Mode

Although the Data Dictionary uses UNICODE to store object names in Japanese language support mode and standard language support mode, the permissible range of characters in object names depends on the language support mode.

For user data, the default server character set is UNICODE.

Advantages of Storing User Data Using UNICODE

Unicode is a 16-bit encoding of virtually all characters in all current languages in the world. Teradata Database UNICODE server character set supports Unicode 4.1 and is designed eventually to store all character data on the server.

UNICODE may be used to store all characters from all single-byte and multibyte client character sets. User data stored as UNICODE can be shared among heterogeneous clients.

User DBC Default Character Set

With Japanese language mode support, the default character set for user DBC is UNICODE.

Extended Support

Extended support allows you to customize Teradata Database to provide additional support for local character set usage.

A sufficiently privileged user can create single-byte and multibyte client character sets that support, with certain constraints, any subset of the Unicode repertoire. Moreover, such a user can customize a collation for the entire Unicode repertoire.

Extended support is available on systems that have been enabled with standard language support or Japanese language support.

For More Information

For more information on the topics presented in this chapter, see *International Character Set Support*.

CHAPTER 15
Query and Database Analysis Tools

Query and Database Analysis Tools Overview

The Teradata Database Optimizer analyzes steps required to execute each query, then chooses the most efficient path that returns answers in the minimal amount of time. The EXPLAIN statement in SQL shows the details of the path chosen by the Optimizer, and can be used as an aid to help improve query performance further. Analyzing complex query plans can be difficult, so Teradata provides tools to help.

This chapter discusses:

- Tools found in Teradata Analyst Pack, a set of tools designed to help automate and simplify query plan analysis. These include:
    - Teradata Visual Explain (VE)
    - Teradata System Emulation Tool (SET)
    - Teradata Index Wizard
- Teradata Database Query Analysis Tools (DBQAT), tools designed to improve the overall performance analysis capabilities of Teradata Database. These include:
    - Query Capture Facility
    - Database Query Log
    - Target Level Emulation (TLE)
    - Database Object Use Count

Teradata Visual Explain

Teradata Visual Explain (VE) is a tool that visually depicts the execution plan of complex SQL requests in a graphical manner.

Teradata VE presents a graphical view of the request broken down into discrete steps showing the flow of data during execution. It has the ability to compare multiple execution plans side by side.

Because comparing optimized queries is easier with Teradata VE, application developers and database administrators can fine-tune the SQL statements so that Teradata Database can access data in the most effective manner.

In order to view an execution plan using Teradata VE, the execution plan information must first be captured in the Query Capture Database (QCD) by means of the Query Capture Facility (QCF).

Teradata VE reads the execution plan, which has been stored in a QCD, and turns it into a series of icons.

Teradata System Emulation Tool

When Target Level Emulation (TLE) information is stored on a test system, Teradata System Emulation Tool (SET) enables you to generate and examine the query plans using the test system Optimizer as if the plans were processed on the production system.

Using Teradata SET you can:

- Change system configuration details, including DBS Control fields, and table demographics and model the impact of various changes on SQL statement performance.
- Determine the source of various Optimizer-based production problems.
- Provide an environment in which Teradata Index Wizard can produce recommendations for a production system workload.

Teradata Index Wizard

Teradata Index Wizard analyzes SQL queries and suggests candidate indexes to enhance their performance.

The workload definitions, supporting statistical and demographic data, and index recommendations are stored in various QCD tables.

Using data from a QCD or the Database Query Log (DBQL), Teradata Index Wizard:

- Recommends, using an INITIATE PARTITION ANALYSIS statement, the potential performance benefits from adding a partitioning expression to one or more tables in a given workload.

  The statement does not recommend the complete removal of any defined partitioning expressions. It considers, however, the alteration of an existing partitioning expression if a Partitioned Primary Index (PPI) table is explicitly included in the table_list.
- Recommends secondary indexes for the tables based on workload details, including data demographics, that are captured using the QCF.
- Enables you to validate index recommendations before implementing the new indexes.
- Enables you to perform what-if analysis on the workload. Teradata Index Wizard allows you to determine whether your recommendations actually improve query performance.
- Interfaces with other Teradata Tools and Utilities, such as Teradata SET to perform offline query analysis by importing the workload of a production system to a test system
- Uses Teradata Visual Explain and Compare tools to provide a comparison of the query plans with and without the index recommendations.

Teradata Index Wizard can be started from Teradata Visual Explain and Teradata SET. Teradata Index Wizard can also open these applications to help in your evaluation of recommended indexes.

Demographics

Teradata Index Wizard needs demographic information to perform index analysis and to make recommendations. You can collect the following types of data demographics using SQL:

Teradata Index Wizard needs demographic information to perform index analysis and to make recommendations. You can collect the following types of data demographics using SQL:

- Query demographics

  Use the INSERT EXPLAIN statement with the WITH STATISTICS and DEMOGRAPHICS clauses to collect table cardinality and column statistics.
- Table demographics

Use the COLLECT DEMOGRAPHICS statement to collect the row count and the average row size in each of the subtables in each AMP on the system.

Query Capture Facility

The Query Capture Facility (QCF) allows you to capture the steps of query plan information into well-known user tables called the Query Capture Database (QCD).

The source of the captured data is produced by the Teradata Database Optimizer, which outputs the text of the EXPLAIN request modifier detailing the final stage of optimization.

Applications of QCF include:

- Store all query plans for customer queries. You can then compare and contrast queries as a function of software release, hardware platform, and hardware configuration.
- Provide data so that you can generate your own detailed analyses of captured query steps using standard SQL DML statements and third-party query management tools.
- Provide the input for these utilities: Teradata Index Wizard (which recommends index definitions for improved efficiency) and Teradata VE (which presents a graphical view of the flow of data during query execution).

Target Level Emulation

Teradata Database supports Target Level Emulation (TLE) both on Teradata Database and in the client as follows.

Teradata Database supports…	On the…
Target Level Emulation (TLE)	Teradata Database.
Teradata System Emulation Tool (SET)	client. For information about Teradata SET, see “Teradata System Emulation Tool”.

Teradata Database provides the infrastructure for TLE. You can use the standard SQL interface to capture the system configuration details and table demographics on one system and store them on another.

Usually the information is obtained from a production system, then stored on a smaller test or development system. With this capability, the Optimizer can generate access plans similar to those that are generated on a production system. You can use the plans to analyze Optimizer-related production problems. This information can also be used by Teradata SET.

Database Query Log

The Database Query Log (DBQL) is a Teradata Database feature that provides a series of predefined tables that can store historical records of queries and their duration, performance, and target activity based on rules you specify.

DBQL is flexible enough to log information on the variety of SQL requests that run on Teradata Database, from short transactions to longer-running analysis and mining queries.

You can request that DBQL log particular query information or just a count of qualified queries. You can specify a hierarchy of rules that DBQL applies to sessions. This allows great flexibility in capturing the right amount of logging data for each use of the Teradata Database system. The rules may target:

- Application names, for example FastLoad or MultiLoad
- A user and a specific account
- A user and all accounts
- All users and a specific account
- All users and all accounts

Each rule may specify that the recording criteria be a mix of:

- Summarizing data based on elapsed time, CPU time, or I/O counts as either a series of intervals or a threshold limit
- Reporting details at the request level, including identification, performance, counts, error code, and SQL text
- Additional, optional detailed data, which may include any or all:
    - Objects used in the request
    - Steps and performance data per step
    - Full SQL text
    - Explain text
    - Optimizer query plan information logged as an XML document

You can define rules, for instance, that tell the DBS to log the first 4,000 SQL characters of a query that runs during a session invoked by a specific user under a specific account if the query exceeds a specified time threshold.

To implement DBQL, you use simple SQL requests to control the starting and ending of the logging activity. These rules may be dynamically changed as needed to support management of the Teradata Database system.

Database Object Use Count

The database administrator and application developer can use Database Object Use Count to capture the number of times an application refers to an object.

Database Object Use Count captures counts for the following:

- Databases
- Tables
- Columns
- Indexes
- Views
- Macros
- Teradata Database stored procedures
- Triggers
- User-defined functions

Object use access information is not counted for EXPLAIN, INSERT EXPLAIN, or DUMP EXPLAIN statements.

Once captured, you can use the information to identify possibly obsolete or unused database objects, particularly those that occupy significant quantities of valuable disk space. Further, Database Object Use Count information can be useful to database query analysis tools like Teradata Index Wizard.

For More Information

For more information on…	See…
Teradata Visual Explain	*Teradata Visual Explain User Guide*
Teradata System Emulation Tool	*Teradata System Emulation Tool User Guide*
Teradata Index Wizard, including support for Partitioned Primary Index	• *Teradata Index Wizard User Guide* • *SQL Request and Transaction Processing* • *SQL Data Definition Language*
Query Capture Database	• *SQL Data Definition Language* • *SQL Request and Transaction Processing*
Database Query Log	• *Database Administration* • *Data Dictionary* • *SQL Data Definition Language*
Target Level Emulation	• *SQL Request and Transaction Processing* • *Teradata System Emulation Tool User Guide*
Database Object Use Count	*Database Administration*

CHAPTER 16
Teradata Database Security

Teradata Database Security Overview

Teradata Database security is based on the following concepts.

Security Element	Description
User	An individual or group of individuals represented by a single user identity.
Privileges	The database privileges explicitly or automatically granted to a user or database.
Logon	The process of submitting user credentials when requesting access to the database.
Authentication	The process by which the user identified in the logon is verified.
Authorization	The process that determines the database privileges available to the user.
Security Mechanism	A method that provides specific authentication, confidentiality, and integrity services for a database session.
Network Traffic Protection	The process for protecting message traffic between Teradata Database and mainframe-attached and workstation-attached clients against interception, theft, or other form of attack.
Message Integrity	Checks data sent across the network against what was received to ensure no data was lost or changed.
Access Logs	Logs that provide the history of users accessing the database and the database objects accessed.

For detailed information on these topics, see *Security Administration*.

Users

Users that access Teradata Database must be defined in the database or a supported directory.

Permanent Database Users

The CREATE USER statement defines permanent database users. Teradata recommends that the username represent an individual. Each username must be unique in the database.

Directory-based Users

Directory-based users that access the database must be defined in a supported directory. Creation of a matching database user may or may not be required, depending upon implementation. One or more configuration tasks must be completed before directory users can access the database.

Proxy Users

Proxy users are end users who access Teradata Database through a middle-tier application set up for trusted sessions. They are authenticated by the application rather than the database. The GRANT CONNECT THROUGH statement assigns role-based database privileges to proxy users. To establish privileges for a particular connection to the database, the application submits the SET QUERY_BAND statement when it connects the user. The SET QUERY_BAND statement and the rules for how it is applied to each proxy user must be coded into the application as part of setup for trusted sessions.

Database Privileges

Users can access only database objects on which they have privileges. The following table lists the types of database privileges and describes how they are acquired by a user.

Privilege	Description
Implicit (Ownership)	Privileges implicitly granted by the database to the owner of the space in which database objects are created.
Automatic	Privileges automatically provided by the system to: • The creator of a database, user, or other database object. • A newly created user or database.
Inherited	Privileges that are passed on indirectly to a user based on its relationship to another user or role to which the privileges were granted directly. • Directory users inherit the privileges of the database users to which they are mapped. Directory users who are members of groups also inherit the privileges defined in external roles to which the groups are mapped. • All users inherit the privileges of PUBLIC, the default database user, whether or not they have any other privileges.
Explicit (GRANT)	Privileges granted explicitly to a user or database in one of the following ways: • GRANT directly to a user or database. • GRANT to a role, then GRANT the role to one or more users.

Directly Granted Privileges

Privileges can be directly given to users with the GRANT statement. Administrators GRANTing a privilege must have been previously granted the privilege they are granting, as well as the WITH GRANT OPTION privilege on the privilege.

For additional information on how to use SQL statements to GRANT and REVOKE privileges, see *SQL Data Control Language*.

Roles

Roles can be used to define privileges on database objects for groups of users with similar needs, rather than granting the privileges to individual users. Roles also require less dictionary space than individually granted privileges. Use the CREATE ROLE statement to define each role, then use the GRANT statement to grant roles to users. The CREATE USER statement must also specify the default role for the user. The MODIFY USER statement can be used to assign additional user roles.

A member of a role may access all objects to which a role has privileges. Users can employ the SET ROLE statement to switch from the default to any alternate role of which the user is a member, or use SET ROLE ALL to access all roles.

For more information on use of roles, see *Database Administration*.

Roles for Proxy Users

Proxy users are users that access the database through a middle-tier application set up to offer trusted sessions. Proxy users are limited to privileges defined in roles that are assigned to them using the GRANT CONNECT THROUGH statement.

For details on using GRANT CONNECT THROUGH, see *Security Administration* and *SQL Data Control Language*.

External Roles

Use external roles to assign role privileges to directory users. External roles are created exactly like database roles; however, they cannot be granted to directory users because these users do not exist in the database. Instead, directory users must be members of groups that are mapped to external roles in the directory.

Directory users mapped to multiple external roles have access to all of them at logon to the database.

For information on mapping directory users to database objects, see *Security Administration*.

Profiles

To simplify user management, an administrator can define a profile and assign it to a group of users who share similar values for the following types of parameters:

- Default database assignment
- Spool space capacity
- Temporary space capacity
- Account strings permitted
- Password security attributes
- Query band

For further information on profiles, see *Database Administration*.

User Authentication

At logon to the database, users are authenticated, meaning their identity is verified and checked against a list of approved users. Authentication comprises the following elements:

- Authentication method
- Logon formats and controls
- Password formats and controls

Authentication Method

Two categories of authentication are available:

- Teradata Database authentication
- External authentication

Teradata Database Authentication

Authentication by Teradata Database requires that the user and its privileges are defined in the database. The TD2 and TDNEGO security mechanisms support Teradata Database authentication. Unless another mechanism has been set as the default, TD2 need not be specified at logon.

External Authentication

External authentication allows Teradata Database users to be authenticated by an agent running on the same network as Teradata Database and its clients.

Teradata Database supports the following external authentication logon types:

- Single sign-on (authentication in the client domain; user credentials do not have to be resubmitted to Teradata Database). The Kerberos, NTLM, SPNEGO, and TDNEGO security mechanisms support Single Sign-on.
- Directory sign-on (authentication by the directory). The LDAP and TDNEGO security mechanisms support Directory Sign-on.
- Sign-on As (user logs on to Teradata Database with credentials the client also recognizes). The Kerberos, LDAP, NTLM, SPNEGO, and TDNEGO security mechanisms support Sign-on As.

For complete information on authentication logon types and the mechanisms that support them, see *Security Administration*.

Logon Format

Logons to Teradata Database can take the following forms:

- Command line
- Graphical User Interface (GUI)
- Logon from a mainframe-attached client
- Logon through a middle-tier application

Command-Line Logon

Users provide the following information when logging on from a workstation-attached client:

- **.logmech:** Specifies the name of the security mechanism that defines the method by which the user will be authenticated and authorized. If a mechanism is not specified, the logon proceeds using the designated default mechanism.
- **.logdata:** Used only for external authentication. Specifies the username and password, and optionally names an individual username, profile, or role where the logon user has access to multiple user identities, profiles, or roles.
- **.logon:** Must be used for Teradata Database authentication. May be used for external authentication except when the logon requires specification of user=, profile=, or realm= to differentiate among multiple user identities, profiles, or realms.

  For either authentication type, .logon specifies the tdpid, username, password, and optional account string information.

  Note:
  The format required for the username and password information varies depending on the user is sign-on method.

For more information on command-line logons, see *Security Administration*.

GUI Logons

Some Teradata Database client applications provide a logon GUI in the form of dialog boxes. The dialog boxes provide fields and buttons that prompt the user to enter the same logon information shown for command-line logons.

For an example of a GUI logon, see *Security Administration*.

Logons from Mainframe-Attached Clients

Sessions logged on from mainframe-attached clients do not support network security features, such as security mechanisms, encryption, or directory management of users. Such logons require submittal of only the username, password, tdpid, and optional account string information using the .logon command.

For information on logons from mainframe-attached clients, see *Security Administration* and *Teradata Director Program Reference*.

Logons Through Connection Pools

Some users access the database through a middle-tier application. The application must log on to Teradata Database with a database username. The application then sets up a connection pool that allows individual end users access to the database but does not require that the users formally logon. End user privileges are determined by whether or not the application and its end-users are set up for trusted sessions.

For information, see Authorization of Middle-tier Application Users.

Logon Controls

Teradata Database automatically grants permission for all users defined in the database to logon from all connected client systems. But administrators can, for example:

- Modify current or default logon privileges for specific users.
- Give individual users permission to log on to the database only from specific mainframe or workstation interfaces.
- Set the maximum number of times a user can submit an unsuccessful logon string.
- Enable authentication of the user by an external application, such as Kerberos or LDAP.
- Restrict access to the database based on the IP address of the machine from which a user logs on.

Password Format Requirements

Passwords must conform to password format rules, which govern the type and number of characters allowed in the password.

Password Controls

Teradata Database provides controls to enable the administration of passwords. Administrators can, for example:

- Restrict the content of passwords:
    - Define limits on minimum and maximum password characters.
    - Allow or require that passwords contain upper and lowercase characters, digits, or special characters.
    - Allow or deny use of restricted words.
- Set the number of days for which a password is valid.
- Assign a temporary password.
- Set the lockout time after the user has exceeded the maximum number of logon attempts.
- Define the period during which a user may not reuse a previous password.

User Authorization

Once users have been authenticated, they are authorized database privileges according to their defined privileges.

Authorization of Permanent Database Users

Permanent database users are defined in the database with a CREATE USER statement. Once authenticated at logon, permanent users are authorized the following privileges:

- Privileges granted directly to the user with the GRANT statement.
- Privileges indirectly given to the user (automatic, implicit, and inherited privileges).
- Privileges granted to a role that has been granted to the user.

Note:
Users with more than one role automatically have access to the default role or all their roles, as specified in the CREATE USER statement. The SET ROLE statement allows users to access roles other than their default role.

Authorization of Directory-Based Users

After being authenticated by the directory, directory-based users are authorized database access privileges according to the following rules:

- If the directory maps users to Teradata Database objects (users, external roles, and profiles), each directory user is authorized the privileges of the objects to which it is mapped.
- If the directory *does not* map users to Teradata Database objects, but the directory username matches a Teradata Database username, the directory user is authorized all the privileges belonging to the matching Teradata Database user.
- If a directory user is neither mapped to any database objects, nor does the directory username match a Teradata Database username, the directory user has no privileges to access the database.

Note:
One or more setup tasks (depending on implementation) must be completed before a directory user can access the database. For information, see *Security Administration*.

Authorization of Middle-tier Application Users

Middle-tier applications may stand between end users and Teradata Database, accepting requests from users, constructing queries from those requests, passing the queries to the database, and then returning results to the users. The middle-tier application logs on to the database, is authenticated as a permanent database user, and establishes a connection pool. The application then authenticates the individual application end users, some of whom may request access to the database through the connection pool.

By default, all end-users accessing the database through a middle-tier application are authorized database privileges and are audited in access logs, based on the single permanent database user identity of the application.

For sites that require end users to be individually identified, authorized, and audited, the middle-tier application can be configured to offer *trusted sessions*. Application end-users that access the database through a trusted session must be set up as *proxy users* and assigned one or more database roles, which determine their privileges in the database. When a proxy user requests database access, the application automatically forwards the user identity and applicable role information to the database.

For further information about the tasks required to set up trusted sessions and proxy users, see *Security Administration*.

Data Protection

Teradata Database provides the following features to enhance data protection:

- By default, the logon string is encrypted to maintain the confidentiality of the username and password employed to log on to Teradata Database.
- Optional data encryption of messages maintains confidentiality of the data transmitted to and from Teradata Database.
- Automatic integrity checking insures that the data is not corrupted during the encryption/transmission/decryption cycle.
- Optional BAR and DSA encryption provides confidentiality of data backups between the BAR server and the storage device.
- Optional SSL/TLS protection for systems using LDAP authentication with simple binding, including:

  - Encryption of the LDAP authentication sequence between Teradata Database and the directory server.
  - Advanced TLS protection, which requires that the database authenticate itself to the directory or that the database and directory mutually authenticate.
- Optional SASL protection for systems using LDAP authentication with Digest-MD5 binding:
  - Protection of the LDAP authentication token exchange sequence between Teradata Database and the directory server.

Directory Management of Users

Normally, users that log on to Teradata Database have been defined in the database using a CREATE USER request. However, because many potential database users may already be defined in a directory running within the client network, Teradata Database allows for authentication and authorization of users by supported directories. Integration of directory managed users simplifies administration by eliminating the need to create a database instance for every user.

For information on how to set up the database and the directory for integrated user management, see *Security Administration*.

Supported Directories

Teradata Database is certified for use with the following directories:

- Active Directory
- Active Directory Application Mode (ADAM)
- Novell eDirectory
- Sun Java System Directory Server
- Other LDAPv3-compliant directories may be usable. Contact Teradata Professional Services for integration assistance.

Database Security Monitoring

To ensure optimal database security, the security administrator should configure the system to audit security events, and then monitor the logs to detect breaches in security and, where necessary, repel security threats. If unauthorized or undesirable activity is detected, the administrator can take remedial actions to address the problem.

Security Monitoring

Teradata Database provides the capability for two types of user security monitoring:

- All user logon and logoff activity is automatically collected in the Event Log and can be accessed using the DBC.LogOnOffV system view. Listed parameters include:
  - Database username
  - Session number
  - Logon events, including the causes of any unsuccessful logons

- Optional access logging records user attempts to access the database and can be accessed using the DBC.AccessLogV view, including the following access parameters:
    - Type of access
    - Type of request
    - Requesting database username
    - Referenced database object
    - Frequency of access

Note:
Access log entries are generated each time the database checks a privilege to determine whether or not it will honor a user request.

Logging of Directory Users

Directory users are logged by directory username rather than by the name of any database user they may be mapped to.

Logging of Middle-tier Application Users

Users that access the database through a middle-tier application by means of a trusted session and who are set up as proxy users are logged by their proxy user name. If the middle-tier application and its end users are *not* set up for trusted sessions, all such users will appear in the log as the same username, that is, the name used by the application.

Enabling and Disabling Access Logging

Use the BEGIN and END LOGGING statements to enable and disable logging and to indicate which access parameters should be logged. Access logging can be set up for almost any database object, for instance, users, databases, or tables.

Security-related System Views

The Data Dictionary provides s number of security-related system views, including the following.

View	Description
DBC.AccessLogV	Each entry indicates a privileges check that has resulted from a user request.
DBC.AccLogRulesV	Lists the access logging rules contained in each BEGIN and END LOGGING statement. These rules are used by the database to determine the access logging criteria for a particular user or database object.
DBC.LogOnOffV	Lists all logon and logoff activity.
DBC.LogonRulesV	Lists the logon rules that result from GRANT and REVOKE LOGON statements. These rules are used by the database to determine whether or not to allow logon privileges to a particular user.

For a complete listing of security-related system views, see *Security Administration*.

Defining a Security Policy

Your security policy should be based on the following considerations:

- Determine your security needs to balance the need for secure data against user needs for quick and efficient data access.
- Review Teradata Database security features to meet your needs.
- Develop a policy that includes both system-enforced and personnel-enforced features.

Publishing a Security Policy

To ensure administrators and users at your site understand and follow site-specific security procedures, the administrator should create a security handbook. The handbook should summarize how you are using Teradata Database security features for your database and should be published and made available to all users.

A security policy document should include:

- Why security is needed.
- Benefits for both the company and the users of adhering to the security policy.
- A description of the specific implementation of Teradata Database security features at your site.
- Suggested/required security actions for users and administrators to follow.
- Whom to contact when security questions arise.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database and Teradata Tools and Utilities books.

IF you want to learn more about…	See…
Security concepts	*Security Administration*
Users	• *Security Administration* • *SQL Data Control Language*
Database privileges, including information on how to use security-related SQL statements, such as GRANT and REVOKE.	
User authentication	*Security Administration*
User authorization, including detailed logon requirements for Teradata Database client applications	the user guide for the respective application
Data protection	*Security Administration*
Directory management of users	

IF you want to learn more about…	See…
Database security monitoring, including security-related system tables and views	*Data Dictionary*
Defining a security policy	*Security Administration*
Publishing a security policy	

CHAPTER 17
System Administration

System Administration Overview

The system administrator is responsible for creating databases and users, and for managing the database.

For information on creating database and users, see Chapter 7: "Database Objects, Databases, and Users."

Session Management

Users must log on to Teradata Database and establish a session before system administrators can do any system accounting.

Session Requests

A session is established after the database accepts the username, password, and account number and returns a session number to the process.

Subsequent Teradata SQL requests generated by the user and responses returned from the database are identified by:

- Host id
- Session number
- Request number

The database supplies the identification automatically for its own use. The user is unaware that it exists.

The context for the session also includes a default database name that is often the same as the user name. When the session ends, the system discards the context and accepts no further Teradata SQL statements from the user.

Establishing a Session

To establish a session, the user logs on to the database.

The procedure varies depending on the client system, the operating system, and whether the user is an application program, or a user in an interactive terminal session using an application program or an interactive user.

Logon Operands

The logon string can include any of the following operands:

- Optional identifier for the database server, called a tdpid

- User name
- Password
- Optional account number

Administrative and Maintenance Utilities

A large number of utilities are available to system administrators to perform various administrative and maintenance functions.

The following table lists some of the major Teradata Database utilities.

Utility	**Purpose**
Abort Host (aborthost)	Aborts all outstanding transactions running on a failed host, until the system restarts the host.
CheckTable (checktable)	Checks for inconsistencies between internal data structures such as table headers, row identifiers, and secondary indexes.
CNS Run (cnsrun)	Allows running of database utilities from scripts.
Configuration Utility (config)	Defines AMPs, PEs, and hosts, and describes their interrelationships for Teradata Database. **Note:** Configuration is documented in *Support Utilities*.
Control GDO Editor (ctl)	Displays the fields of the PDE Control Parameters GDO, and allows modification of the settings.
Cufconfig Utility (cufconfig)	Displays configuration settings for the user-defined function and external stored procedure subsystem, and allows these settings to be modified.
Database Initialization Program (DIP)	Executes one or more of the standard DIP scripts packaged with Teradata Database. These scripts create a variety of database objects that can extend the functionality of Teradata Database with additional, optional features.
DBS Control (dbscontrol)	Displays the DBS Control Record fields, and allows these settings to be modified.
Dump Unload/Load (DUL, DULTAPE)	Saves system dump tables to tape, and restores system dump tables from tape.
Ferret Utility (ferret)	Defines the scope of an action, such as a range of tables or selected vprocs, displays the parameters and scope of the action, and performs the action, either moving data to reconfigure data blocks and cylinders, or displaying disk space and cylinder free space percent in use of the defined scope.

Utility	Purpose
Filer Utility (filer)	Finds and corrects problems within the file system. **Note:** Filer is documented in *Support Utilities*.
Gateway Control (gtwcontrol)	Modifies default values in the fields of the Gateway Control Globally Distributed Object (GDO).
Gateway Global (gtwglobal)	Monitors and controls the Teradata Database workstation-connected users and their sessions.
Lock Display (lokdisp)	Displays a snapshot capture of all real-time database locks and their associated currently-running sessions.
Query Configuration (qryconfig)	Reports the current Teradata Database configuration, including the Node, AMP, and PE identification and status.
Query Session (qrysessn)	Monitors the state of selected Teradata Database sessions on selected logical host IDs.
Reconfiguration Utility (reconfig)	Uses the component definitions created by the Configuration Utility to establish an operational Teradata Database. **Note:** Reconfiguration is documented in *Support Utilities*.
Reconfiguration Estimator (reconfig_estimator)	Estimates the elapsed time for reconfiguration based upon the number and size of tables on the current system, and provides time estimates for the following phases: • Redistribution • Deletion • NUSI building **Note:** Reconfiguration Estimator is documented in *Support Utilities*.
Recovery Manager (rcvmanager)	Displays information used to monitor progress of a Teradata Database recovery.
Show Locks (showlocks)	Displays locks placed by Archive and Recovery and Table Rebuild operations on databases and tables. For details Archive and Recovery, see *Teradata Archive/Recovery Utility Reference* . For details on Table Rebuild, see *Utilities*.

Utility	Purpose
System Initializer (sysinit)	Initializes Teradata Database. Creates or updates the DBS Control Record and other Globally Distributed Objects (GDOs), initializes or updates configuration maps, and sets hash function values in the DBS Control Record. **Note:** System Initializer is documented in *Support Utilities*.
Table Rebuild (rebuild)	Rebuilds tables that Teradata Database cannot automatically recover, including the primary or fallback portions of tables, entire tables, all tables in a database, or all tables in an Access Module Processor (AMP). Table Rebuild can be run interactively or as a background task.
Teradata Locale Definition Utility(tdlocaledef)	Converts a Specification for Data Formatting file (SDF) into an internal, binary format (a GDO) for use by Teradata Database. The SDF file is a text file that defines how Teradata Databaseformats numeric, date, time, and currency output.
Tpareset (tpareset)	Resets the PDE and database components of Teradata Database.
Update DBC (updatedbc)	Recalculates the PermSpace and SpoolSpace values in the DBASE table for the user DBC, and the MaxPermSpace and MaxSpoolSpace values of the DATABASESPACE table for all databases based on the values in the DBASE table.
Update Space (updatespace)	Recalculates the permanent, temporary, or spool space used by a single database or by all databases in a system.
Vproc Manager (vprocmanager)	Manages the virtual processors (vprocs). For example, obtains status of specified vprocs, initializes vprocs, forces a vproc to restart, and forces a Teradata Database restart.

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database and Teradata Tools and Utilities books.

IF you want to learn more about…	See…
Roles and profiles	• *Database Administration*

IF you want to learn more about…	See…
	• *SQL Fundamentals* • *SQL Data Definition Language*
Session management	*Database Administration*
Administrative and maintenance utilities	*Utilities*

CHAPTER 18

Database Management Tools and Utilities

Overview

Teradata Database offers a wide variety of utilities, management tools, and peripherals.

Some of these reside on the Teradata Database system itself, and others are part of the Teradata Tools and Utilities management suite for installation in client environments.

With database management tools, you can back up and restore important data, save dumps, and investigate and control Teradata Database configuration, user sessions, and various aspects of its operation and performance. Management and analysis tools help keep the database running at optimum performance levels.

Data Archiving Utilities

To archive, restore, and recover data in Teradata Database, you can use either:

- Teradata Data Stream Architecture, which is accessible via the Viewpoint BAR Operations portlet
- Teradata Archive and Recovery utility (ARC)

These programs can co-exist at a customer site; however, only the program that created the archive can read it and restore it. Teradata Data Stream Architecture cannot restore an archive created by ARC and vice versa.

For more information, see the documentation for Teradata Data Stream Architecture.

Teradata Parallel Transporter

Teradata Parallel Transporter (Teradata PT) is an object-oriented client application that provides scalable, high-speed, parallel data extraction, loading, and updating.

Teradata PT uses and expands on the functionality of the traditional Teradata standalone extract and load utilities, that is, FastLoad, MultiLoad, FastExport, and TPump.

Teradata PT supports:

- **Process-specific operators**: Teradata PT jobs are run using *operators*. These are discrete object-oriented modules that perform specific extraction, loading, and updating processes. There are four functional operator types:
    - **Producer operators** that read data from a source and write it to data streams.
    - **Consumer operators** that read from data streams and write it to a data target.

    - **Filter operators** that read data from data streams, perform data filtering operations such selection, validation, and condensing, and then write filtered data to data streams.
    - **Standalone operators** that perform processing that does not involve receiving data from, or sending data to, the data stream.
- **Access modules**: These are software modules that give Teradata PT access to various data stores.
- **A parallel execution structure**: Teradata PT can execute multiple instances of an operator to run multiple and concurrent loads and extracts and perform inline updating of data. Teradata PT maximizes throughput performance through scalability and parallelism.
- **The use of data streams**: Teradata PT distributes data into data streams shared with multiple instances of operators. Data streaming eliminates the need for intermediate data storage (data is not written to disk).
- **A single SQL-like scripting language**: Unlike the standalone extract and load utilities, each of which uses its own scripting language, Teradata PT uses a single script language to specify extraction, loading, and updating operations.
- **A GUI-based Teradata PT Wizard**: The Teradata PT Wizard helps you generate simple Teradata PT job scripts.

Teradata Parallel Transporter Application Programming Interface

The Teradata Parallel Transporter Application Programming Interface (Teradata PT API) is a set of application programming interfaces used to load into, and extract from, Teradata Database.

Unlike Teradata PT and the standalone utilities, which are script-driven, Teradata PT API is a functional library that is part an application. This allows applications to have more control during load and extract operations. Closer control of the runtime environment simplifies management processes.

Standalone Data Load and Unload Utilities

Teradata FastExport

Teradata FastExport extracts large quantities of data in parallel from Teradata Database to a client. The utility is the functional complement of the FastLoad and MultiLoad utilities.

Teradata FastExport can:

- Export tables to client files.
- Export data to an Output Modification (OUTMOD) routine. You can write an OUTMOD routine to select, validate, and preprocess exported data.
- Perform block transfers with multisession parallelism.

Teradata FastLoad

Teradata FastLoad loads data into unpopulated tables only. Both the client and server environments support Teradata FastLoad.

Teradata FastLoad can:

- Load data into an empty table.

  FastLoad loads data into one table per job. If you want to load data into more than one table, you can submit multiple FastLoad jobs.

- Perform block transfers with multisession parallelism.

Teradata MultiLoad

Teradata MultiLoad supports bulk inserts, updates, and deletes against initially unpopulated or populated database tables. Both the client and server environments support Teradata MultiLoad.

Teradata MultiLoad can:

- Run against multiple tables.
- Perform block transfers with multisession parallelism.
- Load data from multiple input source files.
- Pack multiple SQL statements and associated data into a request.
- Perform data Upserts.

Teradata Parallel Data Pump

Teradata TPump uses standard SQL (not block transfers) to maintain data in tables.

TPump also contains a resource governing method whereby you can control the use of system resources by specifying how many inserts and updates occur minute-by-minute. This allows background maintenance for insert, delete, and update operations to take place at any time of day while Teradata Database is in use.

TPump provides the following capabilities:

- Has no limitations on the number of instances running concurrently.
- Uses conventional row hash locking, which provides some amount of concurrent read and write access to the target tables.
- Supports the same restart, portability, and scalability as Teradata MultiLoad.
- Perform data Upserts.

Access Modules

Access modules are dynamically linked software components that provide block-level I/O interfaces to external data storage devices.

Access modules import data from data sources and return it to a Teradata utility. Access modules are dynamically linked to one or more Teradata utilities by the Teradata Data Connector API.

Access modules provide transparent, uniform access to various data sources, isolating you from, for example, device and data store dependencies.

Teradata Database supports the following access modules, which read data, but do not write it.

- **Named Pipes Access Module**, which enables you to load data into the Teradata Database from a UNIX OS named pipe. A pipe is a type of data buffer that certain operating systems allow applications to use for the storage of data.
- Teradata **WebSphere MQ Access Module,**which enables you to load data from a message queue using IBM's WebSphere MQ (formerly known as MQ Series) message queuing middleware.
- **Teradata Access Module for JMS**, which enables you to load data from a JMS-enabled messaging system using JMS message queuing middleware.
- **Teradata OLE DB Access Module**, which enables you to transfer data between an OLE DB provider and Teradata Database.
- **Custom Access Modules**, which you can use with the DataConnector operator for access to specific systems.

Teradata access modules work on many operating systems with the following Teradata Tools and Utilities:

- BTEQ
- Teradata FastExport
- Teradata FastLoad
- Teradata MultiLoad
- Teradata PT
- Teradata TPump

Basic Teradata Query

Basic Teradata Query (BTEQ) software is a general-purpose, command-based program that allows users on a workstation or mainframe to communicate with one or more Teradata Database systems and to format reports for both print and screen output.

Using BTEQ you can submit SQL queries to the Teradata Database. BTEQ formats the results and returns them to the screen, to a file, or to a designated printer.

A BTEQ session provides a quick and easy way to access a Teradata Database. In a BTEQ session, you can:

- Enter Teradata SQL statements to view, add, modify, and delete data.
- Enter BTEQ commands.
- Enter operating system commands.
- Create and use Teradata stored procedures.

Session and Configuration Management Tools

Database management tools include utilities for investigating active sessions and the state of Teradata Database configuration, such as:

- Query Session
- Query Configuration
- Gateway Global

The following table contains information about the capabilities of each utility.

This utility…	**Does the following…**
Query Session	• Provides information about active Teradata Database sessions. • Monitors the state of all or selected sessions on selected logical host IDs attached to Teradata Database. • Provides information about the state of each session including session details for Teradata Index Wizard. For more information about Teradata Index Wizard, see Teradata Index Wizard.
Query Configuration	Provides reports on the current Teradata Database configuration, including:

This utility…	Does the following…
	• Node • AMP • PE identification and status
Gateway Global	Allows you to monitor and control the sessions of Teradata Database workstation-connected users.

System Resource and Workload Management Tools and Protocols

Teradata Database supports specific tools, protocols, and tool architectures for system resource and workload management. Among these are:

- Write Ahead Logging
- Ferret utility
- Priority Scheduler
- Teradata Active System Management

Write Ahead Logging

Teradata Database uses a Write Ahead Logging (WAL) protocol. According to this protocol, writes of permanent data are written to a log file that contains the records representing updates. The log file is written to disk at key moments, such as at transaction commit.

Modification to permanent data from different transactions, all written to the WAL log, can also be batched. This achieves a significant reduction in I/O write operations. One I/O operation can represent multiple updates to permanent data.

The WAL Log is conceptually similar to a table, but the log has a simpler structure than a table. Log data is a sequence of WAL records, different from normal row structure and not accessible via SQL.

The WAL Log includes the following:

- Redo Records for updating disk blocks and insuring file system consistency during restarts, based on operations performed in cache during normal operation.
- Transient Journal (TJ) records used for transaction rollback.

WAL protects all permanent tables and all system tables, except Transient Journal (TJ) tables, user journal tables, and restartable spool tables (global temporary tables). Furthermore, WAL allows Teradata Database to be reconstructed from the WAL Log in the event of a system failure.

The file system stages in-place writes through a disk area called the DEPOT, a collection of cylinders. Staging in-place writes through the DEPOT ensures that either the old or the new copy is available after a restart.

Ferret Utility

The Ferret utility, ferret, lets you display and set storage space utilization attributes of the Teradata Database. Ferret dynamically reconfigures the data in the Teradata file system while maintaining data integrity during changes. Ferret works on various data levels as necessary: vproc, table, subtable, WAL log, disk, and cylinder.

The Ferret utility allows display and modification of the WAL Log and its index.

Ferret includes the following:

- The SCANDISK option includes checking the WAL Log by default.
- The SCOPE option can be set to just the WAL Log.
- The SHOWBLOCKS and SHOWSPACE options display log statistics and space.

Teradata Active System Management

Teradata Active System Management (TASM) is a set of products, including system tables and logs, that interact with each other and a common data source. It facilitates automation in the following four key areas of system management:

- Workload management
- Performance tuning
- Capacity planning
- Performance monitoring

With careful planning, TASM can improve and optimize your workload management and performance. It can also improve response times and ensure more consistent response times for critical work. This can reduce the effort required by DBAs.

Some Key TASM Products and Components

The following table describes some of the key products and components of TASM.

Product / Component	**Description**
Teradata Workload Analyzer (Teradata WA)	A product that analyzes collected DBQL data to help group workloads and define rules to manage system performance. See “Database Query Log”.
Teradata Viewpoint	A product that enables users to: • Create rules for filtering, throttling, and defining classes of queries (workload definitions) • Create events to monitor system resources
Open APIs	An interface, invoked from any application, that provides an SQL interface to PMPC through User-Defined Functions (UDFs) and external stored procedures. See “Workload Management Application Programming Interface”.
Query Bands	A set of name-value pairs that are defined by the user or middle-tier application. Query Bands allow the user

Product / Component	Description
	to tag session or transactions with an ID through an SQL interface. For specific information on Query Bands, see *SQL Data Definition Language* and *Database Administration*.
Resource Usage Monitor	Data collection subsystem that includes TASM-related data collection. See "Resource Usage Monitoring".

Teradata Viewpoint

Teradata Viewpoint, including the Workload Designer portlet, supports the creation of the following based on business-driven allocations of operating resources:

- Filter rules
- Throttle rules
- Rules that define classes of queries (Workload Definitions [WDs])
- Events to monitor system resources
- States to allow changes to rule values

Request-Specific Performance Management

Teradata Viewpoint Workload Designer portlet lets users define rules according to which workload is managed. The following table describes the three categories of Teradata Active System Management (ASM) rules.

Rules	Description
Filter	Reject unwanted logon and query requests before they are executed. Filters restrict access to specific database objects for some or all types of SQL requests. You can prohibit queries that are estimated to access too many rows, take too long, and perform some types of joins.
Throttle (also called concurrency rules)	Enforce session and query concurrency limits on specific objects. When creating throttle rules, you can: • Restrict the number of requests simultaneously executed against a database object (such as requests made by a user, or against a table). • Reject requests over the concurrency limit on a state-by-state basis (where *state* is a complete set of working values for a rule set). • Enforce concurrency limits on FastLoad, MultiLoad, FastExport, DSA, and ARC utilities. • Apply them to Priority Scheduler Performance Groups (PGs).

Rules	Description
Workload (also called Workload Definitions [WDs])	Specify how Teradata Database should handle queries while they are executing by specifying parameters for up to 36 separate workload definitions. In each workload definition, you can specify: • The Include and Exclude conditions or database objects, or both the workload definition and database objects that determine whether a query is assigned to the class. **Note:** Wildcard characters (such as "*" and "?") can be used to include all or a group of database objects, and then exclude specific ones. • The execution priority (by more or less transparently creating a Priority Scheduler configuration). • The query concurrency limits. Requests over the concurrency limit can be rejected on a state-by-state basis. • The set of conditions that invoke an exception once a query has started running.

Event-Based Performance Management

Teradata Viewpoint Workload Designer portlet allows you to specify filter, throttle, and workload rules (WDs) that dynamically adjust their behavior based on system and user-defined events.

An event is any condition or indication that you think is pertinent to workload management.

Event	Description
Health Condition	Reflects the health of the system, such as a Teradata Database component degrading or failing (a node down, for example), or resources below a threshold for some period of time.
Planned Environment	Includes the kinds of work Teradata Database is expected to perform, such as batch and loads or month-end processing, defined as time periods.

Utility Management

You can manage load utilities (FastLoad, MultiLoad, FastExport) and Teradata Archive/Recovery utility similarly to how you manage SQL requests: by classifying them into workload definitions with throttle limits based on:

- Utility name
- "Who" criteria (such as user, account, or queryband)
- "Where" criteria (the name of the database, table, or view) (not available for Archive/Recovery)

For example, you can specify that:

- User A cannot run more than two FastLoad jobs at the same time
- Only one MultiLoad job can run at a time against Database XYZ

In addition, using Workload Designer, a portlet of Viewpoint, you can define rules that control the number of sessions that Archive/Recovery or a load utility can use. To create session configuration rules, specify at least one of the following criteria:

- Utility name (required)
- Data size
- “Who” criteria (for example, user, account, client address, or query band)

and then specify the number of sessions to be used when the criteria are met.

For example, you can specify using:

- Ten sessions for standalone MultiLoad jobs submitted by Joe
- Four sessions for JDBC FastLoad from the application called WebApp

If you do not create a session configuration rule or the utility currently running does not meet the rule criteria, then Workload Designer automatically uses default session rules to select the number of sessions based on these criteria:

- Utility name
- System configuration (the number of AMPs)
- Optional data size

Teradata Workload Analyzer

Teradata Workload Analyzer (WA) is a tool to analyze resource usage patterns of workloads and to correlate these patterns with specific classification and exception criteria. Teradata WA helps:

- DBAs identify classes of queries (workloads).
- Provides recommendations on workload definitions and operating rules to ensure that database performance meets Service Level Goals (SLGs).

Through graphical displays such as pie charts showing CPU and I/O utilization by accounts, applications, users, profiles, or query bands, through histograms showing actual service levels, as well as recommended settings, Teradata WA makes it easier for DBAs to manage distribution of resources effectively.

Teradata SQL Assistant

Teradata SQL Assistant stores, retrieves, and manipulates data from Teradata Database (or any database that provides an ODBC interface). You can store the data on your desktop PC, and produce consolidated results or perform data analyses using tools such as Microsoft Excel. In addition to ODBC connectivity, Teradata SQL Assistant can connect to Teradata Database using .NET Data Provider for Teradata, which can be downloaded from the Teradata Corporation website.

Teradata SQL Assistant Java Edition can be used to attach to Teradata Database or any other database that provides a JDBC interface.

The following table contains information about key features of Teradata SQL Assistant.

This feature…	Allows you to…
Queries	• Use SQL syntax examples to help compose your SQL statements. • Send statements to any ODBC database or the same statement to many different databases.

This feature…	Allows you to…
	• Limit data returned to prevent runaway execution of statements.
Reports	• Create reports from any database that provides an ODBC interface. • Use an imported file to create many similar reports (query results or answer sets); for example, display the DDL (SQL) that was used to create a list of tables.
Data manipulation	• Export data from the database to a file on a PC. • Import data from a PC file directly to the database. • Create a historical record of the submitted SQL with timings and status information, such as success or failure. • Use the Database Explorer Tree to easily view database objects.
Teradata Database stored procedures	Use a procedure builder that gives you a list of valid statements for building the logic of a stored procedure, using Teradata Database syntax.

Teradata SQL Assistant electronically records your SQL activities with data source identification, timings, row counts, and notes. Having this historical data allows you to build a script of the SQL that produced the data. The script is useful for data mining.

Teradata Studio

Teradata Studio is a client-based GUI used to perform database administration, query development, and management tasks on Teradata Databases, Teradata Aster Databases, and Hadoop systems. Teradata Studio has a modular display that allows users to create a custom user interface. Teradata Studio allows you to:

- Search for and view database objects or space and skew
- Write SQL using content assistance, edit and parse SQL, view history, and use SQL templates and annotations
- Build and edit SQL requests visually with SQL Query Builder
- Export result sets to Excel, XML, or text files
- Generate and run COLLECT STATISTICS requests
- Load and extract data with a smart data loader that automatically determines data types, creates tables, and loads the data
- Copy database objects to another database or system
- Move space between databases
- Compare SQL DDL side-by-side

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database and Teradata Tools and Utilities books.

IF you want to learn more about…	See…
Data Archiving Utilities, including: • Archive/Recovery utility • Backup and Recovery (BAR), including Backup Application Software products	• *Teradata Archive/Recovery Utility Reference* • *Teradata BAR Solutions Guide*
Teradata Parallel Transporter	• *Teradata Parallel Transporter User Guide* • *Teradata Parallel Transporter Reference* • *Teradata Parallel Transporter Operator Programmer Guide*
Teradata Parallel Transporter Application Programming Interface	*Teradata Parallel Transporter Application Programming Interface Programmer Guide*
Standalone Data Load and Export Utilities	• *Teradata FastExport Reference* • *Teradata FastLoad Reference* • *Teradata MultiLoad Reference* • *Teradata Parallel Data Pump Reference*
Access Modules	*Teradata Tools and Utilities Access Module Programmer Guide*
Basic Teradata Query (BTEQ)	*Basic Teradata Query Reference*
Session and Configuration Management	*Utilities*
System Resource and Workload Management, including: • Write Ahead Logging (WAL) • Ferret utility • Priority Scheduler • Teradata Active System Management (ASM)	• *Support Utilities* • *Utilities* • *Database Administration* • *Teradata Workload Analyzer User Guide*
Teradata SQL Assistant	• *Teradata SQL Assistant for Microsoft Windows User Guide* • Teradata Downloads
Teradata Studio	Teradata Studio documentation

CHAPTER 19

System Monitoring

Overview

This chapter discusses various aspects of monitoring Teradata Database, including the monitoring tools used to track system and performance issues.

Teradata Viewpoint

Teradata Viewpoint enables database and system administrators and business users to monitor and manage Teradata Database systems from anywhere using a standard web browser.

Teradata Viewpoint allows users to view system information, such as query progress, performance data, and system saturation and health through preconfigured portlets displayed from within the Teradata Viewpoint portal. Portlets can also be customized to suit individual user needs. User access to portlets is managed on a per-role basis.

Database administrators can use Teradata Viewpoint to determine system status, trends, and individual query status. By observing trends in system usage, system administrators are better able to plan project implementations, batch jobs, and maintenance to avoid peak periods of use. Business users can use Teradata Viewpoint to quickly access the status of reports and queries and drill down into details. Teradata Viewpoint includes the following components.

Component	Description
Viewpoint Server	The hardware on which the other components are installed. It is an Ethernet-connected server housed in the Teradata rack so it can be managed by server management software.
Viewpoint Portal	A Java-based portal that resides on the Viewpoint Server and is the framework for delivering the Teradata Viewpoint portlets. The portal includes built-in security, user management, and role-based permissions for defining which users can perform specific actions and for granting or limiting system access.
Data Collection Service	A Java process that tracks daily activities on Teradata Database and maintains data history for comparison and reporting purposes. Because it maintains a local cache database, users can access the system even during critical events, such as outages.
Portlets	Preconfigured and customizable content that provides a current view of the workload and throughput of Teradata systems and a historical view of system capacity and use.

For more information on Teradata Viewpoint, contact your Teradata representative.

QUERY STATE Command

Teradata Database is always in one of several states. You can monitor these states using the QUERY STATE command from the DBW Supervisor window.

The following table lists the valid system states:

System State	Description
Database is not running	Teradata Database has not been started; it cannot be accessed from a client or used for processing.
Database Startup	Teradata Database is undergoing startup processing and is not yet ready to accept requests.
Logons are disabled - Users are logged on	No new sessions can log on, but existing sessions are still logged on.
Logons are disabled - The system is quiescent	Logons are disabled and no sessions are logged on.
Logons are enabled - Users are logged on	New sessions can log on and work is in process.
Logons are enabled - The system is quiescent	Logons are enabled, but no sessions are logged on.
Only user DBC Logons are enabled	Only new DBC sessions can log on and work is in process.
RECONFIG is running	Reconfiguration is being run.
System is operational without PEs - Sessions are not allowed	Either there are no PEs configured into the system or all PEs are offline/down.
TABLEINIT is running	Database startup has detected that there are no tables on the system and is running TABLEINIT to create the system tables. This usually occurs during the next system restart after a SYSINIT.

Resource Usage Monitoring

Teradata Database has facilities that permit you to monitor the use of resources such as:

- AMPs
- AWTs
- CPUs
- BYNET activity
- Disk activity
- Performance Groups

Resource usage data is useful for the following purposes:

- Measuring system performance
- Measuring component performance
- Assisting with on-site job scheduling
- Identifying potential performance impacts

- Planning installation, upgrade, and migration
- Analyzing performance degradation and improvement
- Identifying problems such as bottlenecks, parallel inefficiencies, down components, and congestion

Resource Usage Data Reporting

Data is reported at the logging period. When a new logging period starts, the data is gathered in the Gather Buffer, then updated to the Log Buffer and logged to the database resource usage tables.

How to Control Logging of ResUsage Data

Several mechanisms exist within Teradata Database for setting the logging rates of ResUsage data. The control sets allow users to do any of the following:

- Specify data logging rate.
- Enable or disable ResUsage data logging on a table-by-table basis.
- Enable or disable summarization of the data.
- Enable or disable active row filtering.

Logging rates control the frequency that resource usage data is logged to the ResUsage tables.

You can use the SET LOGTABLE command to establish the logging of resource usage information. The system inserts data into ResUsage tables every logging period for the tables that have logging enabled. You can use the statistics collected in the ResUsage tables to analyze system bottlenecks, determine excessive swapping, and detect system load imbalances.

ResUsage Tables and Views

Resource usage data is stored in system tables and views in the DBC database. Macros installed with Teradata Database generate reports that display the data.

You can use SQL to access resource usage data if you have the proper privileges. You can also write your own queries or macros on resource usage data.

ResUsage Data Categories

Each row of ResUsage data contains two broad categories of information:

- Housekeeping, containing identifying information
- Statistical

Each item of statistical data falls into a defined kind and class. Each kind corresponds to one (or several) different things that may be measured about a resource.

ResUsage Macros

The facilities for analyzing ResUsage data are provided by means of a set of ResUsage macros tailored to retrieving information from a set of system views designed to present ResUsage information.

Summary Mode

Summary Mode is one method for improving system performance. It reduces database I/O by consolidating and summarizing data on each node on the system. Because Summary Mode reduces the availability of detail, you can log normally to tables in which greater detail is needed. Activate Summary Mode individually for the tables in which great detail is not needed.

For more information about Summary Mode and those tables supported in Summary Mode, see *Resource Usage Macros and Tables*.

Performance Monitoring

Several facilities exist for monitoring and controlling system performance.

For information on the Workload Management API, including those for dynamic workload management and Query Banding, see "Workload Management Application Programming Interface".

Account String Expansion

Account String Expansion (ASE) is a mechanism that enables AMP usage and I/O statistics to be collected. ASE supports performance monitoring for an account string.

The system stores the accumulated statistics for a user/account string pair as a row in DBC.Acctg table in the Data Dictionary. You can use the DBC.AMPUsageV view to access this information.

Each user/account string pair results in a new set of statistics and an additional row. You can use this information in capacity planning or in charge back and accounting software.

At the finest granularity, ASE can generate a summary row for each SQL request. You can also direct ASE to generate a row for each user, each session, or for an aggregation of the daily activity for a user.

ASE permits you to use substitution variables to include date and time information in the account id portion of a user logon string. The system inserts actual values for the variables at Teradata SQL execution time.

The TDPTMON

Teradata Director Program (TDP) User Transaction Monitor (TDPTMON) is a client routine that enables a system programmer to write code to track TDP elapsed time statistics.

System Management Facility

The System Management Facility (SMF) is available in the z/OS environment only. This facility collects data about Teradata Database performance, accounting, and usage.

Data is grouped into the following categories:

- Session information
- Security violations
- PE stops

For More Information

For more information on the topics presented in this chapter, see the following Teradata Database and Teradata Tools and Utilities books.

IF you want to learn more about…	See…
QUERY STATE Command	"Database Window (xdbw)" in *Utilities*
Resource Usage Monitoring	*Resource Usage Macros and Tables*
Performance Monitoring, including: • Account String Expansion • the TDPTMON • System Management Facility • Workload Management Application Programming Interface	• *Database Administration* • *Security Administration* • *Application Programming Reference*