Overview of BTEQ

Introduction

BTEQ is an abbreviation of Basic Teradata Query. It is a general-purpose, command-based program that allows users on a workstation 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, a file, or to a designated printer.

Environment

You install and run BTEQ on the client portion of either a channel-attached or a networkattached system.

BTEQ Sessions

A BTEQ session provides a quick and easy way to access a Teradata Database. In a BTEQ session, you can do the following:

* enter Teradata SQL statements to view, add, modify, and delete data.
* enter BTEQ commands.
* enter operating system commands.
* create and use Teradata stored procedures.

BTEQ Session Modes

BTEQ operates in both batch and interactive modes:

interactive mode: start a BTEQ session, and submit commands to the database as needed.

batch mode: prepare scripts or macros, and then submit them to BTEQ for processing.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Teradata Director Program (TDP) (or MTDP):

The TDP manages data communications, including:

• logging

• recovery

• restarts

• physical I/O

• session and security management

In channel attached system.

The Micro Teradata Director Program (MTDP) performs the same functions as the TDP in network attached system.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LOGON Command Elements:

syntax Optional/Required… Specifies…

----------- ------------------------------- ----------------------------------------------------------------------------------------------------------

tdpid: Optional the Teradata Director Program ID of the Teradata server that you are logging on to.

user id: required your user identifier.

password: required the password for your userid.

acctid : Optional the account identifier for your userid.

------------------------------------------------------------------------------------------------------------------------------------------------------

Logging on to the RDBMS:

Logging On in Interactive Mode:

.LOGON tdpid/userid,, 'acctid'

Password= 'password'

---------------------------------------------------------------------------------

.logon 10.74.161.248/td\_user1

OR

.logon localtd/dbc,,'dbc'

.logon localtd/dbc,

Password:

OR

.logon <IP Address>/dbc,,'dbc'

.logon <IP Address>/dbc,

Password:

Logging On in Batch Mode:

Submit the LOGON command in an input file,including the password, as follows:

.LOGON tdpid/userid, password, 'acctid'

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Logging Off the Teradata RDBMS / Exiting BTEQ:

LOGOFF : end the current RDBMS sessions without exiting BTEQ

EXIT or QUIT : end the current RDBMS sessions and exit BTEQ.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C:\Users\Administrator>bteq

Teradata BTEQ 13.00.00.03 for WIN32.

Copyright 1984-2009, Teradata Corporation. ALL RIGHTS RESERVED.

Enter your logon or BTEQ command:

.logon 127.0.0.1/dbc

.logon 127.0.0.1/dbc

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.quit

.quit

\*\*\* You are now logged off from the DBC.

\*\*\* Exiting BTEQ...

\*\*\* RC (return code) = 0

C:\Users\Administrator>bteq

Teradata BTEQ 13.00.00.03 for WIN32.

Copyright 1984-2009, Teradata Corporation. ALL RIGHTS RESERVED.

Enter your logon or BTEQ command:

.logon localtd/dbc

.logon localtd/dbc

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.exit

.exit

\*\*\* You are now logged off from the DBC.

\*\*\* Exiting BTEQ...

\*\*\* RC (return code) = 0

C:\Users\Administrator>bteq

Teradata BTEQ 13.00.00.03 for WIN32.

Copyright 1984-2009, Teradata Corporation. ALL RIGHTS RESERVED.

Enter your logon or BTEQ command:

.logon localtd/dbc

.logon localtd/dbc

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.logoff

.logoff

\*\*\* You are now logged off from the DBC.

Teradata BTEQ 13.00.00.03 for WIN32. Enter your logon or BTEQ command:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**DCL statements in Teradata**

Teradata has the following list of Data control statements.

* GIVE
* GRANT
* REVOKE
* GRANT LOGON
* REVOKE LOGON

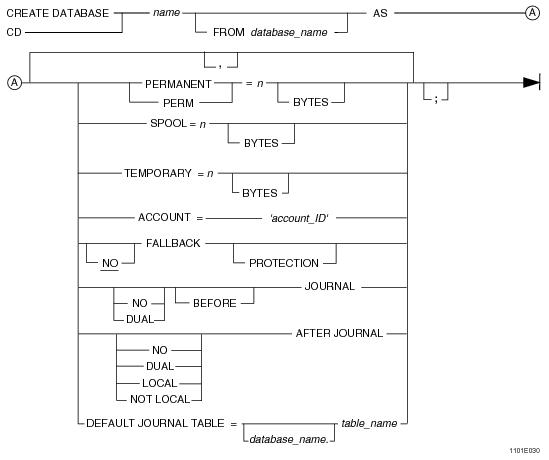
**Teradata Database:**

A Teradata database is a defined, logical repository for:

• Tables (requires perm space)

• Views (uses no perm space)

• Macros (uses no perm space) - prdefined set of SQL statements.



where:

|  |  |
| --- | --- |
| **Syntax Element …** | **Specifies …** |
| *name* | the name of the new database. See *SQL Fundamentals* for the rules for naming database objects. |
| *database\_name* | the name of the immediate owning user or database. The default is the user name associated with the current session.  See *SQL Fundamentals* for the rules for naming database objects. |
| PERMANENT=*n* BYTES | the number of bytes to be reserved for permanent storage for this database. The space is taken from unallocated space in the database or user of the immediate owner.  You must specify a value for this option. There is no default.  The number of bytes can be entered as an integer, decimal, or floating point value. For example, you can write *one thousand* as either 1000 or 1E3. |
| SPOOL=*n* BYTES | the number of bytes to be allowed for spool and volatile temporary files. The default is the largest value that is not greater than the owner spool space and that is a multiple of the number of AMPs on the system. *n* must not exceed the owner spool space.  The number of bytes can be entered as an integer, decimal, or floating point value. |
| TEMPORARY=*n* BYTES | a definition for how many bytes are to be allowed by default for creating materialized global temporary tables by users within this database. The default is the largest value that is not greater than the owner temporary space and that is a multiple of the number of AMPs on the system. *n* must not exceed the owner temporary space.  If no default temporary space is defined for a database, then the space allocated for any materialized global temporary tables created in that database is set to the maximum temporary space allocated for its immediate owner of the database.  Note that temporary space is reserved prior to spool space for any database defined with this characteristic.  Each materialized global temporary table requires a minimum of 512 bytes of PERM space to contain its table header.  Disk usage for a materialized global temporary table is charged to the temporary space allocation of the user who referenced the table.  The number of bytes can be entered as an integer, decimal, or floating point value. |
| ACCOUNT = ‘*account\_ID’* | the account to be charged for the space used by this database. If not specified, it defaults to the account identifier of the immediate owner database.  Each value for *account\_ID* must follow the standard Teradata Database naming rules (see *SQL Fundamentals* for a description of Teradata Database identifiers and a list of the characters they can contain). |
| FALLBACK PROTECTION  NO FALLBACK PROTECTION | whether to create and store a duplicate copy of each table created in the new database. The default value is NO FALLBACK. This setting can be overridden for a particular data table when the table is created.  The FALLBACK keyword used alone implies PROTECTION. |
| NO DUAL BEFORE JOURNAL | the number of before change images to be maintained by default for each data table created in the new database.  The JOURNAL keyword without NO or DUAL implies single copy journaling.  If journaling is specified, a DUAL journal is maintained for data tables with FALLBACK protection.  The JOURNAL keyword without BEFORE implies both types (BEFORE and AFTER) of images. |
| NO DUAL LOCAL NOT LOCAL AFTER JOURNAL | the type of image to be maintained by default for data tables created in the new database.  If journaling is specified, a DUAL journal is maintained for data tables with FALLBACK protection.  NOT LOCAL and LOCAL specify whether single after-image journal rows for non-fallback data tables are written on the same virtual AMP (LOCAL) as the changed data rows, or on another virtual AMP in the cluster (NOT LOCAL). See also the CREATE DATABASE “Local Journaling” topic in *SQL Data Definition Language Detailed Topics*.  The JOURNAL keyword without AFTER implies both types (BEFORE and AFTER) of images.  The default is no journaling.  If only AFTER JOURNAL is specified, then a before change image is not maintained. If both types are specified, the two specifications must not conflict.  This setting can be overridden for a particular data table when the table is created (see “CREATE TABLE” in *SQL Data Definition Language*). |
| DEFAULT JOURNAL TABLE =*database\_name.table\_name* | the default table that is to receive the journal images of data tables created in the new database.  A database can contain only one default journal table. However, any table in a particular database can use a journal table in a different database.  *table\_name* must be defined if journaling is requested, but it need not reside in the new database. See *SQL Fundamentals* for the rules for naming database objects.  *table\_name* is automatically created in the new database if *database\_name*is not specified, or the new database is specified.  If a different database is specified, then that database must exist and*table\_name* must have been defined as its default journal table. |

Database Creation:

In Teradata database having 3 types of space:

Perm Space = max amount of space available for tables

Spool Space = max amount of work space available for requests

Temp Space = used for temporary tables - will survive a restart

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE DATABASE marketing FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

FALLBACK,

BEFORE JOURNAL,

DUAL AFTER JOURNAL,

DEFAULT JOURNAL TABLE= marketing\_jrnl;

CREATE DATABASE marketing FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

FALLBACK,

BEFORE JOURNAL,

DUAL AFTER JOURNAL,

DEFAULT JOURNAL TABLE= marketing\_jrnl;

\*\*\* Database has been created.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE DATABASE finance FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

FALLBACK,

BEFORE JOURNAL,

DUAL AFTER JOURNAL,

DEFAULT JOURNAL TABLE=fin\_jrnl;

CREATE DATABASE finance FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

FALLBACK,

BEFORE JOURNAL,

DUAL AFTER JOURNAL,

DEFAULT JOURNAL TABLE=fin\_jrnl;

\*\*\* Database has been created.

\*\*\* Total elapsed time was 2 seconds.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE DATABASE sales FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

NO FALLBACK,

NO JOURNAL;

CREATE DATABASE sales FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

NO FALLBACK,

NO JOURNAL;

\*\*\* Database has been created.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE DATABASE hr FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

FALLBACK,

BEFORE JOURNAL,

DUAL AFTER JOURNAL,

DEFAULT JOURNAL TABLE=**finance.fin\_jrnl**;

CREATE DATABASE hr FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

FALLBACK,

BEFORE JOURNAL,

DUAL AFTER JOURNAL,

DEFAULT JOURNAL TABLE=finance.fin\_jrnl;

\*\*\* Database has been created.

\*\*\* Total elapsed time was 1 second.

**NOTE: This new database is using journal table of another database (finance.fin\_jrnl);**

select \* from dbc.databases where databasename in ('finance','mkt','sales','hr');

DatabaseName CreatorName OwnerName

------------------------------ ------------------------------ -------------

mkt DBC DBC

hr DBC DBC

sales DBC DBC

finance DBC DBC

select \* from dbc.Database\_Default\_Journals;

\*\*\* Query completed. 4 rows found. 3 columns returned.

\*\*\* Total elapsed time was 1 second.

DatabaseName Journal\_DB JournalName

------------------------------ ------------------------------ -------------

tdhari tdhari tdharijrnl

finance finance fin\_jrnl

mkt mkt mkt\_jrnl

hr finance fin\_jrnl

BTEQ -- Enter your DBC/SQL request or BTEQ command:

DROP DATABASE sales;

DROP DATABASE sales;

\*\*\* Database/User has been dropped.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

**NOTE: This database do not have default journal table so it can be deleted easily.**

Set as default database for the current session.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

database mkt;

database mkt;

\*\*\* New default database accepted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

Delete all database objects from the specified database.

This statement does not delete the definition for a database from the data dictionary.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

delete database mkt;

delete database mkt;

\*\*\* Database's tables deleted.

\*\*\* Total elapsed time was 5 seconds.

DROP DATABASE

A database must be empty before it can be dropped.

So dropping a database is a two step process.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

delete database mkt;

delete database mkt;

\*\*\* Database's tables deleted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

DROP database mkt;

DROP database mkt;

\*\*\* Failure 3552 Cannot DROP databases with tables, journal tables, views,

or macros.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

Drop default journal table of finance database.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

MODIFY DATABASE mkt AS DROP DEFAULT JOURNAL TABLE;

MODIFY DATABASE mkt AS DROP DEFAULT JOURNAL TABLE;

\*\*\* Database/User has been modified.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

DROP DATABASE mkt;

DROP DATABASE mkt;

\*\*\* Database/User has been dropped.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

delete database FINANCE;

delete database FINANCE;

\*\*\* Database's tables deleted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

DROP database FINANCE;

DROP database FINANCE;

\*\*\* Failure 3552 Cannot DROP databases with tables, journal tables, views, or macros.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

MODIFY DATABASE FINANCE AS DROP DEFAULT JOURNAL TABLE;

MODIFY DATABASE FINANCE AS DROP DEFAULT JOURNAL TABLE;

MODIFY DATABASE FINANCE AS DROP DEFAULT JOURNAL TABLE;

$

**\*\*\* Failure 3772 You cannot DROP a journal that is the default for other databases.**

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

**NOTE: You cannot drop default journal table of finance database because it is also used by HR database.**

BTEQ -- Enter your DBC/SQL request or BTEQ command:

select \* from dbc.Database\_Default\_Journals;

select \* from dbc.Database\_Default\_Journals;

\*\*\* Query completed. 3 rows found. 3 columns returned.

\*\*\* Total elapsed time was 1 second.

DatabaseName Journal\_DB JournalName

------------------------------ ------------------------------ -------------

tdhari tdhari tdharijrnl

finance finance fin\_jrnl

**hr finance fin\_jrnl**

BTEQ -- Enter your DBC/SQL request or BTEQ command:

DELETE DATABASE hr;

DELETE DATABASE hr;

\*\*\* Database's tables deleted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

DROP DATABASE hr;

DROP DATABASE hr;

\*\*\* Database/User has been dropped.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

MODIFY DATABASE FINANCE AS DROP DEFAULT JOURNAL TABLE;

MODIFY DATABASE FINANCE AS DROP DEFAULT JOURNAL TABLE;

\*\*\* Database/User has been modified.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

DROP database FINANCE;

DROP database FINANCE;

\*\*\* Database/User has been dropped.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

select \* from dbc.databases where databasename in ('finance','mkt','sales','hr');

\*\*\* Query completed. No rows found.

\*\*\* Total elapsed time was 1 second.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Teradata User:**

* A Teradata user is a database with an assigned password.
* A Teradata user may also own tables, views, macros, and triggers but users with no permspace may not own tables.
* A user may logon to Teradata and access objects within:
  + Itself
  + Other databases for which it has access rights
* A user is created with the CREATE USER command.

**In Teradata, a user is also similar to a database. They both can be assigned space and contain database objects except that the user is assigned a password.**

Syntax

Following is the syntax for CREATE USER.

CREATE USER username

AS

[PERMANENT|PERM] = n BYTES

PASSWORD = password

TEMPORARY = n BYTES

SPOOL = n BYTES;

**While creating a user, the values for user name, Permanent space and Password is mandatory. Other fields are optional.**

SELECT \* FROM DBC.USERS;

SELECT \* FROM DBC.USERS;

\*\*\* Query completed. 9 rows found. 30 columns returned.

\*\*\* Total elapsed time was 1 second.

UserName CreatorName Passwor

------------------------------ ------------------------------ -------------

TDPUSER DBC 09

SystemFe DBC 09

SysAdmin DBC 09

Crashdumps DBC 09

Sys\_Calendar DBC 09

twm DBC 09

tdwm DBC 09

tduser DBC 09

DBC DBC 09

SELECT \* FROM DBC.USERDB;

SELECT \* FROM DBC.USERDB;

\*\*\* Query completed. 25 rows found. 2 columns returned.

\*\*\* Total elapsed time was 1 second.

Id Name

-------- ------------------------------

0000EA03 TDPUSER

0000F203 SQLJ

0000F603 FINANCIAL

0000ED03 SYSADMIN

0000F703 MANUFACTURING

0000EF03 SYS\_CALENDAR

0000EC03 SYSTEMFE

00000200 DEFAULT

0000F103 SYSSPATIAL

0000EE03 CRASHDUMPS

00000100 DBC

0000E903 SYSLIB

0000FA03 TWM

0000F303 TDUSER

0000F903 TPCH

0000F403 SAMPLES

0000F003 SYSUDTLIB

0000FC03 TWM\_SOURCE

00000000 ALL

0000F503 RETAIL

0000FB03 TWM\_RESULTS

0000EB03 TDWM

00000300 PUBLIC

00000500 EXTUSER

0000F803 TRANSPORTATION

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE USER TD01

AS

PERMANENT = 100000 BYTES

PASSWORD = admin$123

TEMPORARY = 100000 BYTES

SPOOL = 100000 BYTES;

CREATE USER TD01

AS

PERMANENT = 100000 BYTES

PASSWORD = admin$123

TEMPORARY = 100000 BYTES

SPOOL = 100000 BYTES;

\*\*\* User has been created.

\*\*\* Total elapsed time was 1 second.

NOTE: Without delimeters.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE USER TD02

AS

PERMANENT = 100000 BYTES,

PASSWORD = admin$123,

TEMPORARY = 100000 BYTES,

SPOOL = 100000 BYTES;

CREATE USER TD02

AS

PERMANENT = 100000 BYTES,

PASSWORD = admin$123,

TEMPORARY = 100000 BYTES,

SPOOL = 100000 BYTES;

\*\*\* User has been created.

\*\*\* Total elapsed time was 1 second.

NOTE: Parameter delimited by coma.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE USER TD03

AS

PERMANENT = 100000 BYTES,

PASSWORD = admin$123;

CREATE USER TD03

AS

PERMANENT = 100000 BYTES,

PASSWORD = admin$123;

\*\*\* User has been created.

\*\*\* Total elapsed time was 1 second.

NOTE: Parameter TEMPORARY,SPOOL are optional.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE USER TD04 FROM DBC

AS

PERMANENT = 100000 BYTES,

PASSWORD = admin$123,

TEMPORARY = 100000 BYTES,

SPOOL = 100000 BYTES;

CREATE USER TD04 FROM DBC

AS

PERMANENT = 100000 BYTES,

PASSWORD = admin$123,

TEMPORARY = 100000 BYTES,

SPOOL = 100000 BYTES;

\*\*\* User has been created.

\*\*\* Total elapsed time was 1 second.

NOTE: Parameter FROM clause is used.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE USER TD05 FROM DBC

AS

PERMANENT = 100000 BYTES,

PASSWORD = admin$123,

TEMPORARY = 100000 BYTES,

SPOOL = 100000 BYTES,

ACCOUNT='$LUSER'

DEFAULT DATABASE = transportation;

CREATE USER TD05 FROM DBC

AS

PERMANENT = 100000 BYTES,

PASSWORD = admin$123,

TEMPORARY = 100000 BYTES,

SPOOL = 100000 BYTES,

ACCOUNT='$LUSER'

DEFAULT DATABASE = transportation;

\*\*\* User has been created.

\*\*\* Total elapsed time was 1 second.

NOTE: Parameter ACCOUNT,DEFAULT DATABASE clause is used.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE USER TD06 FROM DBC AS

PASSWORD= admin$123

PERM=200000

SPOOL=500000

TEMPORARY = 100000

DEFAULT DATABASE = financial;

CREATE USER TD06 FROM DBC AS

PASSWORD= mkb

PERM=200000

SPOOL=500000

TEMPORARY = 100000

DEFAULT DATABASE = financial;

\*\*\* User has been created.

\*\*\* Total elapsed time was 1 second.

NOTE: Parameter DEFAULT DATABASE clause is used.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

SELECT \* FROM DBC.DATABASES;

SELECT \* FROM DBC.DATABASES;

\*\*\* Query completed. 33 rows found. 16 columns returned.

\*\*\* Total elapsed time was 1 second.

DatabaseName CreatorName OwnerName

------------------------------ ------------------------------ -------------

financial DBC Samples

tdhari DBC DBC

TD01 DBC DBC

TD02 DBC DBC

TD03 DBC DBC

TD04 DBC DBC

TD05 DBC DBC

TD06 DBC DBC

Sys\_Calendar DBC DBC

SystemFe DBC DBC

SYSLIB DBC DBC

BLR DBC DBC

DBC DBC DBC

Crashdumps DBC DBC

twm DBC Samples

twm\_source DBC Samples

tpch DBC Samples

retail DBC Samples

SYSUDTLIB DBC DBC

Samples DBC DBC

All DBC DBC

tdwm DBC DBC

twm\_results DBC Samples

EXTUSER EXTUSER EXTUSER

PUBLIC DBC DBC

SYSSPATIAL DBC DBC

manufacturing DBC Samples

transportation DBC Samples

TDPUSER DBC DBC

tduser DBC DBC

Default DBC DBC

SysAdmin DBC DBC

SQLJ DBC DBC

BTEQ -- Enter your DBC/SQL request or BTEQ command:

SELECT \* FROM DBC.USERS;

SELECT \* FROM DBC.USERS;

\*\*\* Query completed. 16 rows found. 30 columns returned.

\*\*\* Total elapsed time was 1 second.

UserName CreatorName Passwor

------------------------------ ------------------------------ -------------

TD01 DBC 17

TD02 DBC 17

TD03 DBC 17

TD04 DBC 17

TD05 DBC 17

TD06 DBC 17

TDPUSER DBC 09

SystemFe DBC 09

SysAdmin DBC 09

Crashdumps DBC 09

twm DBC 09

tdwm DBC 09

Sys\_Calendar DBC 09

tduser DBC 09

DBC DBC 09

SELECT \* FROM dbc.allrights

WHERE username='user-id'

AND databasename='database-name';

SELECT \* FROM dbc.allrights

WHERE username='TD06'

AND databasename= 'FINANCIAL';

SELECT \* FROM dbc.allrights

WHERE username='TD06'

AND databasename= 'FINANCIAL';

\*\*\* Query completed. No rows found.

\*\*\* Total elapsed time was 1 second.

**NOTE: Implicit privileges are not vislible in dbc.allrights.**

BTEQ -- Enter your DBC/SQL request or BTEQ command:

grant create database on financial to td06;

grant create database on financial to td06;

\*\*\* Grant accepted.

\*\*\* Total elapsed time was 1 second.

**NOTE: Create database privilege is given to user TD06 on database financial.**

BTEQ -- Enter your DBC/SQL request or BTEQ command:

Grant create database on td06 to td06;

Grant create database on td06 to td06;

\*\*\* Grant accepted.

\*\*\* Total elapsed time was 1 second.

**NOTE: Create database privilege is given to user TD06 on database TD06.**

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.logon localtd/td06

.logon localtd/td06

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

select user;

select user;

\*\*\* Query completed. One row found. One column returned.

\*\*\* Total elapsed time was 1 second.

User

------------------------------

TD06

BTEQ -- Enter your DBC/SQL request or BTEQ command:

select database;

select database;

\*\*\* Query completed. One row found. One column returned.

\*\*\* Total elapsed time was 1 second.

Database

------------------------------

FINANCIAL

**NOTE: This new database will consume space from default FINANCIAL database.**

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE DATABASE sales FROM FINANCIAL AS

PERM = 20000 BYTES,

SPOOL = 50000 BYTES,

TEMPORARY = 10000 BYTES,

NO FALLBACK,

NO JOURNAL;

CREATE DATABASE sales FROM FINANCIAL AS

PERM = 20000 BYTES,

SPOOL = 50000 BYTES,

TEMPORARY = 10000 BYTES,

NO FALLBACK,

NO JOURNAL;

\*\*\* Database has been created.

\*\*\* Total elapsed time was 1 second.

**NOTE: This new database will consume space from user TD06.**

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE DATABASE sports FROM td06 AS

PERM = 20000 BYTES,

SPOOL = 50000 BYTES,

TEMPORARY = 10000 BYTES,

NO FALLBACK,

NO JOURNAL;

CREATE DATABASE sports FROM td06 AS

PERM = 20000 BYTES,

SPOOL = 50000 BYTES,

TEMPORARY = 10000 BYTES,

NO FALLBACK,

NO JOURNAL;

\*\*\* Database has been created.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.logon localtd/dbc

.logon localtd/dbc

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

.set width 150

BTEQ -- Enter your DBC/SQL request or BTEQ command:

SELECT UserName, DatabaseName, TableName, ColumnName, AccessRight

FROM dbc.allrights

WHERE username='TD06'

AND databasename= 'FINANCIAL';

SELECT UserName, DatabaseName, TableName, ColumnName, AccessRight

FROM dbc.allrights

WHERE username='TD06'

AND databasename= 'FINANCIAL';

\*\*\* Query completed. One row found. 10 columns returned.

\*\*\* Total elapsed time was 1 second.

UserName DatabaseName TableName ColumnName AccessRight

------------------------------ ------------------------------ ------------------------------ ------------------------------ -----------

TD06 financial All All CD

BTEQ -- Enter your DBC/SQL request or BTEQ command:

SELECT UserName, DatabaseName, TableName, ColumnName, AccessRight

FROM dbc.allrights

WHERE username='TD06'

AND databasename= 'SALES';

SELECT UserName, DatabaseName, TableName, ColumnName, AccessRight

FROM dbc.allrights

WHERE username='TD06'

AND databasename= 'SALES';

\*\*\* Query completed. 25 rows found. 10 columns returned.

\*\*\* Total elapsed time was 1 second.

UserName DatabaseName TableName ColumnName AccessRight

------------------------------ ------------------------------ ------------------------------ ------------------------------ -----------

TD06 sales All All CD

TD06 sales All All CV

TD06 sales All All DD

TD06 sales All All CP

TD06 sales All All DP

TD06 sales All All I

TD06 sales All All E

TD06 sales All All CU

TD06 sales All All DU

TD06 sales All All PD

TD06 sales All All R

TD06 sales All All U

TD06 sales All All D

TD06 sales All All CT

TD06 sales All All ST

TD06 sales All All DA

TD06 sales All All CA

TD06 sales All All DF

TD06 sales All All DG

TD06 sales All All CG

TD06 sales All All RS

TD06 sales All All DM

TD06 sales All All DV

TD06 sales All All DT

TD06 sales All All CM

BTEQ -- Enter your DBC/SQL request or BTEQ command:

SELECT UserName, DatabaseName, TableName, ColumnName, AccessRight

FROM dbc.allrights

WHERE username='TD06'

AND databasename= 'SPORTS';

SELECT UserName, DatabaseName, TableName, ColumnName, AccessRight

FROM dbc.allrights

WHERE username='TD06'

AND databasename= 'SPORTS';

\*\*\* Query completed. 25 rows found. 5 columns returned.

\*\*\* Total elapsed time was 1 second.

UserName DatabaseName TableName ColumnName AccessRight

------------------------------ ------------------------------ ------------------------------ ------------------------------ -----------

TD06 sports All All CU

TD06 sports All All ST

TD06 sports All All DU

TD06 sports All All DD

TD06 sports All All DP

TD06 sports All All DM

TD06 sports All All DV

TD06 sports All All DT

TD06 sports All All CD

TD06 sports All All E

TD06 sports All All R

TD06 sports All All U

TD06 sports All All D

TD06 sports All All CT

TD06 sports All All CP

TD06 sports All All CV

TD06 sports All All RS

TD06 sports All All CG

TD06 sports All All CM

TD06 sports All All DG

TD06 sports All All PD

TD06 sports All All DF

TD06 sports All All CA

TD06 sports All All I

TD06 sports All All DA

BTEQ -- Enter your DBC/SQL request or BTEQ command:

select databasename, creatorname, ownername, accountname from dbc.databases

where creatorname='td06';

select databasename, creatorname, ownername, accountname from dbc.databases

where creatorname='td06';

\*\*\* Query completed. 2 rows found. 4 columns returned.

\*\*\* Total elapsed time was 1 second.

DatabaseName CreatorName OwnerName AccountName

------------------------------ ------------------------------ ------------------------------ ---------------

sales TD06 financial DBC

sports TD06 TD06 DBC

**NOTE: Sales database got space from financial database and Sports database got space from user TD06.**

BTEQ -- Enter your DBC/SQL request or BTEQ command:

DROP USER TD01;

DROP USER TD01;

**\*\*\* Database/User has been dropped.**

\*\*\* Total elapsed time was 1 second.

**Modify User Password**

BTEQ -- Enter your DBC/SQL request or BTEQ command:

MODIFY USER td06 AS PASSWORD=admin#123;

MODIFY USER td06 AS PASSWORD=admin#123;

\*\*\* Database/User has been modified.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Accounts**

While creating a new user, the user may be assigned to an account. ACCOUNT option in CREATE USER is used to assign the account. A user may be assigned to multiple accounts.

## Syntax

Following is the syntax for CREATE USER with account option.

CREATE USER username

PERM = n BYTES

PASSWORD = password

ACCOUNT = accountid

## Example

The following example creates the user TD02 and assigns the account as IT and Admin.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE USER TD07

AS

PERMANENT = 100000 BYTES

PASSWORD = admin$123

TEMPORARY = 100000 BYTES

SPOOL = 100000 BYTES

ACCOUNT = ('IT','Admin');

CREATE USER TD07

AS

PERMANENT = 100000 BYTES

PASSWORD = admin$123

TEMPORARY = 100000 BYTES

SPOOL = 100000 BYTES

ACCOUNT = ('IT','Admin');

\*\*\* User has been created.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

grant create database on dbc to td07;

grant create database on dbc to td07;

\*\*\* Grant accepted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.logon localtd/td07

.logon localtd/td07

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Echo accepted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

CREATE DATABASE sales FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

NO FALLBACK,

NO JOURNAL;

CREATE DATABASE sales FROM dbc AS

PERM = 2000000 BYTES,

SPOOL = 5000000 BYTES,

TEMPORARY = 1000000 BYTES,

NO FALLBACK,

NO JOURNAL;

\*\*\* Database has been created.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

select database ;

select database ;

\*\*\* Query completed. One row found. One column returned.

\*\*\* Total elapsed time was 1 second.

Database

------------------------------

TD07

BTEQ -- Enter your DBC/SQL request or BTEQ command:

database sales;

database sales;

\*\*\* New default database accepted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

select database ;

select database ;

\*\*\* Query completed. One row found. One column returned.

\*\*\* Total elapsed time was 1 second.

Database

------------------------------

SALES

BTEQ -- Enter your DBC/SQL request or BTEQ command:

select user;

select user;

\*\*\* Query completed. One row found. One column returned.

\*\*\* Total elapsed time was 1 second.

User

------------------------------

TD07

BTEQ -- Enter your DBC/SQL request or BTEQ command:

The user can specify the account id while logging into Teradata system or after being logged into the system using SET SESSION command.

.LOGON username, passowrd,accountid

OR

SET SESSION ACCOUNT = 'accountid' FOR REQUEST | SESSION;

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.LOGON localtd/td07,, 'IT','ADMIN'

.LOGON localtd/td07,

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.LOGON 127.0.0.1/td07

.LOGON 127.0.0.1/td07

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

set session account ='it' for request;

set session account ='it' for request;

\*\*\* Set SESSION accepted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

set session account ='admin' for request;

set session account ='admin' for request;

\*\*\* Set SESSION accepted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

set session account ='admin' for session;

set session account ='admin' for session;

\*\*\* Set SESSION accepted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.logon localtd/dbc

.logon localtd/dbc

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 15 seconds.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.set retlimit 100

.set retlimit 100

BTEQ -- Enter your DBC/SQL request or BTEQ command:

SELECT \* FROM DBC.ACCOUNTS;

SELECT \* FROM DBC.ACCOUNTS;

\*\*\* Query completed. 31 rows found. 3 columns returned.

\*\*\* Total elapsed time was 1 second.

UserId AccountName RowTy

-------- ----------------------------- ---------------------------

0000FD03 DBC U

00001404 DBC U

0000F203 DBC U

0000EA03 $H U

0000EB03 $H-DBC-MANAGER U

00001704 $LUSER U

0000EC03 SystemFe U

0000ED03 SysAdmin U

0000EF03 Sys\_Calendar U

0000EE03 Crashdumps U

00001B04 IT U

00001B04 Admin U

BTEQ -- Enter your DBC/SQL request or BTEQ command:

**Granting Privileges in Teradata**

You can use the GRANT statement to:

* Explicitly give users, databases, roles, and PUBLIC privileges on a database object
* Grant a role to a user or another role.

The following table summarizes the privileges, authority, and requirements for a recipient (grantee) of a new privilege.

|  |  |
| --- | --- |
| **Recipient** | **Privilege Acquisition and Type** |
| Creators | Some general usage notes include:   * The creator of a user or database needs to submit an explicit GRANT CREATE statement to grant to that new user the explicit privilege to create functions, databases, users, or procedures in the space that the user owns. Also, to create roles and profiles, the user or database must be granted the privilege to do so. Note that roles and profiles are not owned. * The creator or owner of a table or view needs to submit an explicit GRANT statement to grant to other users the SELECT privilege or any other privilege on that table or view. * The creator has the EXECUTE privilege to statements in the body of the created macro or stored procedure. The system checks that privileges to the objects targeted by those statements are performed against the immediately owing user or database. * To grant the EXECUTE PROCEDURE privilege to other users, the creator must first have the EXECUTE PROCEDURE privilege with the WITH GRANT OPTION. * To grant the EXECUTE FUNCTION privilege to other users, the creator must first have the EXECUTE FUNCTION privilege with the WITH GRANT OPTION. |
| Owners | Owners have the implicit privilege to grant explicit privileges on their owned objects (except for stored procedures and functions). |
| Users | Although a user or database has certain explicit privileges automatically granted on itself when it is created, a new user/database must be explicitly granted others. |

If a user has lost explicit privileges on tables, views, or macros he owns, the system may grant him the ability to still access them because the owner has implicit privileges on tables, views, and macros. He can also explicitly grant the privilege back to himself by submitting the appropriate GRANT statement.

**Using the GRANT *privilege* Statement**

The following table describes some of the privileges you might grant to other users or roles. For example, to grant the privilege to alter functions, submit the following SQL statement:

GRANT **ALTER FUNCTION** ON *object* TO *grantee*;

Some list of privileges.

|  |  |
| --- | --- |
| **Privilege** | **Purpose** |
| ALTER FUNCTION | Recompiles a user-defined function and allows you to switch the protection modes on functions. |
| ALTER PROCEDURE  ALTER EXTERNAL PROCEDURE | Recompiles a stored procedure or external stored procedures. |
| CHECKPOINT | Creates a synchronization entry in a journal table or an index analysis workload. Checkpoints are used in ARC and load client utilities such as TPump, FastLoad, and MultiLoad. |
| CREATE… | Creates a database, user, role, profile, table, view, macro, stored procedure, permanent journal table, index (secondary), join index, hash index, trigger, or user-defined function. |
| DELETE | Deletes rows from a table. |
| DROP… | Remove a database, user, role, profile, table, view, macro, journal table, stored procedure, index (secondary), join index, hash index, trigger or user-defined function.  DROP TABLE also allows ALTER TABLE, creating an index on a table, collecting statistics on a table, and modifying a database or user. |
| DUMP | Archive an AMP, AMP range, or AMP cluster, or one, several, or a range of databases, data tables, journal tables, or users. Used in ARC as well as online archiving. |
| EXECUTE | Execute a macro. |
| EXECUTE FUNCTION | Execute a UDF. |
| EXECUTE PROCEDURE | Refers to the corresponding CALL statement. |
| INDEX (table level only) | Privilege that includes the:  **•**CREATE INDEX and DROP INDEX privileges  **•**COLLECT STATISTICS and DROP STATISTICS privileges |
| INSERT | Load new rows in a table, directly or through a view. |
| REPLCONTROL | To define and alter replication groups, you must first have the REPLCONTROL privilege. This privilege is similar to the ROLE and PROFILE privileges in that it cannot be granted to specific tables, databases, or users. |
| RESTORE | Used in ARC to:  **•**Restore by AMP, database or user, journal or table.  **•**or execute the following:   * DELETE JOURNAL to drop a permanent journal. * ROLLBACK to use a before-image permanent journal to restore tables (than write to that journal) to their state before a modification. * ROLLFORWARD — Uses an after-image permanent journal to update tables (that write to that journal) to reflect a modification. |
| SELECT | Select the information in one, many, or all columns from a table or view. |
| UDTMETHOD | Allows you to use, create, alter, and drop UDTs as well as create new methods or drop and replace existing methods. |
| UDTTYPE | Create or drop UDTs. It also permits you to alter the ordering, casting, or transform behavior by referencing pre-existing methods. |
| UDTUSAGE | Execute all SQL statements that reference existing UDTs and their existing methods. |
| UPDATE | Modify column values in a table, directly or through a view. |

**Privileges Automatically Received**

Automatic privileges are granted by the Teradata Database system to:

• User DBC

• The creator (and in some cases, the modifier) of a database, user, or object

• A newly created user or database

As a creator, you automatically receive specific privileges. However, a created user or database may need to be explicitly granted privileges not automatically granted upon creation.

As an owner of database objects, you implicitly receive privileges on those objects.

**Implicit Privileges for Owners**

Implicit privileges for an owner are the same privileges a creator gets automatically.

Implicit privileges for an owner are never automatically inserted into DBC.AccessRights. If the owner happens to be also the creator, then the system will automatically insert rows into the DBC.AccessRights table for the owner as it does for all creators. If the owner is not the same user as the creator, then only the creator gets rows in the DBC.AccessRights table.

If an owner wants access to a database object that he owns but the object was created by a different user, and the owner does not have a row for the privilege on that object in DBC.AccessRights, then the owner will have to do a self-grant to get a row inserted into DBC.AccessRights before he can access the object.

In other words, the owner must submit a GRANT statement on the object to grant himself privileges on the object. He must do this to exercise his implicit privilege on the object when a required row for an operation in DBC.AccessRights is missing.

Note: A row for a privilege in the DBC.AccessRights table is usually missing if the privilege has been revoked or the user is not the immediate owner of the object.

Operations that require CHECKPOINT, DUMP, RESTORE, or ANY privilege may be authorized based on ownership and do not need any self-grant.

**Automatic Privileges For Creators (of Users and Databases)**

When you submit a CREATE USER or CREATE DATABASE statement, **as the creator of a database or user, the system automatically grants you the following privileges on the database or user you just created.** They include the following privileges:

• ANY

• CHECKPOINT

• CREATE AUTHORIZATION

• CREATE DATABASE

• CREATE MACRO

• CREATE TABLE

• CREATE TRIGGER

• CREATE USER

• CREATE VIEW

• DELETE

• DROP AUTHORIZATION

• DROP DATABASE

• DROP FUNCTION

• DROP MACRO

• DROP PROCEDURE

• DROP TABLE

• DROP TRIGGER

• DROP USER

• DROP VIEW

• DUMP

• EXECUTE

• INSERT

• SELECT

• STATISTICS

• RESTORE

• UPDATE

**Automatic Privileges For Created User/Database**

The created user or database gets the following privileges automatically on itself.

• ANY

• CHECKPOINT

• CREATE AUTHORIZATION

• CREATE MACRO

• CREATE TABLE

• CREATE TRIGGER

• CREATE VIEW

• DELETE

• DROP AUTHORIZATION

• DROP FUNCTION

• DROP MACRO

• DROP PROCEDURE

• DROP TABLE

• DROP TRIGGER

• DROP VIEW

• DUMP

• EXECUTE

• INSERT

• SELECT

• STATISTICS

• RESTORE

• UPDATE

Note: The privileges a new user/database receives after its creation is the same as those listed for automatic creator above except for CREATE USER, CREATE DATABASE, DROP USER, and DROP DATABASE.

In addition to not getting CREATE USER, CREATE DATABASE, DROP USER, or DROP DATABASE automatically, the created user/database also does not get the privileges listed in the following section “Privileges That Must Be Explicitly Granted”.

Automatically granted privileges may be redundant with inherited privileges from roles or implicit privileges for immediate owners. If so, you should consider revoking redundant privileges to reduce size of the AccessRights table.

**Privileges That Must Be Explicitly Granted**

**For Creators**

The creator does not get the following privileges automatically on the created user/database. They must explicitly be granted:

• ALTER EXTERNAL PROCEDURE

• ALTER FUNCTION

• ALTER PROCEDURE

• CREATE EXTERNAL PROCEDURE

• CREATE FUNCTION

• CREATE PROCEDURE

• EXECUTE FUNCTION

• EXECUTE PROCEDURE

• SHOW

**For Created User/Database**

The created user or database does not get the following privileges automatically on itself; they must be granted explicitly:

• ALTER EXTERNAL PROCEDURE

• ALTER FUNCTION

• ALTER PROCEDURE

• CREATE EXTERNAL PROCEDURE

• CREATE FUNCTION

• CREATE PROCEDURE

• CREATE DATABASE

• CREATE USER

• DROP DATABASE

• DROP USER

• EXECUTE FUNCTION

• EXECUTE PROCEDURE

• SHOW

These are the same as the privileges a creator must be explicitly granted plus the four excluded privileges (CREATE USER/DATABASE and DROP USER/DATABASE).

**Additional Privileges**

The following privileges must be granted by DBC or another user who already has these privileges:

**Table level privileges**

• INDEX

• REFERENCES

**GLOP Data privileges**

• CREATE GLOP

Note: The creator and owner of a GLOP SET are automatically granted GLOP MEMBER and DROP GLOP privilege on the created GLOP set.

• DROP GLOP

• GLOP MEMBER

**Monitor privileges**

• ABORTSESSION

• MONRESOURCE

• MONSESSION

• SETRESRATE

• SETSESSRATE

**System level privileges**

• CREATE PROFILE

• CREATE ROLE

• DROP PROFILE

• DROP ROLE

• REPLCONTROL

• CTCONTROL

**UDT privileges**

• UDTMETHOD

• UDTUSAGE

• UDTTYPE

**Required Privileges**

You must be one of the following to grant privileges on an object using the SQL GRANT statement:

* User *DBC.*
* An owner of the object.
* A user possessing each privilege to be granted.

A user might have a privilege either by having been granted it explicitly or by inheriting it from a role as a result of creating a

view, macro, or stored procedure.

Note the following details about the privileges required to execute a GRANT (SQL Form) request.

* A user need not be related to a grantor through ownership to receive a privilege.

A grantor does not need to have any privilege, including WITH ADMIN OPTION, on the grantee to grant a privilege to it, whether the grantee is a role, a user, database, or PUBLIC.

* If a GRANT statement is on a database or user, the privilege applies to all objects, both current and future, created in that space.

If a REVOKE statement later removes the privilege, the privilege is dropped for all objects, regardless of when they were created.

A REVOKE statement at the object level cannot remove a privilege from that object that was granted on the database or user.

* When you specify the WITH GRANT OPTION phrase, the recipient of the privilege can then grant that privilege to other users.

An owner implicitly has the WITH GRANT OPTION privilege on any database, user, or object it owns.

|  |  |
| --- | --- |
| **An owner can explicitly grant any or all privileges on any of the following …** | **TO …** |
| **•**a child database  **•**a child user  **•**a database object | **•**any other database  **•**any other user  **•**a role  **•**PUBLIC  You cannot assign row‑level security privileges to PUBLIC. |

* Any privilege granted automatically or explicitly can be revoked using the REVOKE statement.
* **Implicit privileges cannot be revoked.**

**Valid Privileges for Teradata Database SQL**

The privileges listed in the following table are all valid SQL privileges for Teradata Database.

The term [(*column\_list*)] following a privilege indicates a parenthetically enclosed set of optional comma-separated column names on which the privilege is to be granted to the specified user or role.

|  |  |  |
| --- | --- | --- |
| **Privilege** | | |
| **•**ABORT SESSION  **•**ALTER EXTERNAL PROCEDURE  **•**ALTER FUNCTION  **•**ALTER PROCEDURE  **•**AUTHORIZATION  **•**CHECKPOINT  **•**CONSTRAINT ASSIGNMENT  **•**CONSTRAINT DEFINITION  **•**CREATE AUTHORIZATION  **•**CREATE DATABASE  **•**CREATE EXTERNAL PROCEDURE  **•**CREATE FUNCTION  **•**CREATE GLOP | **•**CREATE MACRO  **•**CREATE OWNER PROCEDURE  **•**CREATE PROCEDURE  **•**CREATE PROFILE  **•**CREATE ROLE  **•**CREATE TABLE  **•**CREATE TRIGGER  **•**CREATE USER  **•**CREATE VIEW  **•**CTCONTROL  **•**DATABASE  **•**DELETE  **•**DROP  **•**DROP AUTHORIZATION  **•**DROP DATABASE  **•**DROP FUNCTION | **•**DROP GLOP  **•**DROP MACRO  **•**DROP PROCEDURE  **•**DROP PROFILE  **•**DROP ROLE  **•**DROP TABLE  **•**DROP TRIGGER  **•**DROP USER  **•**DROP VIEW  **•**DUMP  **•**EXECUTE  **•**EXECUTE FUNCTION  **•**EXECUTE PROCEDURE  **•**FUNCTION  **•**GLOP  **•**GLOP MEMBER  **•**INDEX |
| **•**INSERT [(*column\_list*)]  **•**INSERT [(ALL BUT] …*column\_list*)]  **•**MACRO  **•**MONITOR RESOURCE  **•**MONITOR SESSION  **•**NONTEMPORAL  **•**OVERRIDE DELETE CONSTRAINT  **•**OVERRIDE DUMP CONSTRAINT  **•**OVERRIDE INSERT CONSTRAINT  **•**OVERRIDE RESTORE CONSTRAINT | **•**OVERRIDE SELECT CONSTRAINT  **•**OVERRIDE UPDATE CONSTRAINT  **•**PROCEDURE  **•**REFERENCES [(*column\_list*)]  **•**REFERENCES [(ALL BUT] …*column\_list*)]  **•**REPLCONTROL  **•**RESTORE  **•**SELECT [(*column\_list*)]  **•**SELECT [(ALL BUT] …*column\_list*)]  **•**SET RESOURCE RATE | **•**SET SESSION RATE  **•**SHOW  **•**STATISTICS  **•**TABLE  **•**TRIGGER  **•**UDTMETHOD  **•**UDTTYPE  **•**UDTUSAGE  **•**UPDATE [(*column\_list*)]  **•**UPDATE [(ALL BUT] …*column\_list*)]  **•**USER  **•**VIEW |

**SHOW Privilege**

The SHOW privilege enables you to have access to database object definitions and create text without having access to the data contained by the objects on which the privilege is granted.

For example, SHOW permits a user to execute HELP and SHOW requests against an object while at the same time not being able to SELECT from it.

SHOW is an explicit privilege. Teradata Database does not grant the creator of an object this privilege automatically on the created user, database, or database object; SHOW must be granted explicitly. You must have the SHOW privilege WITH GRANT OPTION to be able to grant this privilege explicitly to other users and databases.

User DBC automatically has the SHOW privilege and can grant it on dictionary tables to other users and databases.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.logon localtd/dbc

.logon localtd/dbc

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

grant show on tdhari.emp1\_dup4 to td07;

grant show on tdhari.emp1\_dup4 to td07;

\*\*\* Grant accepted.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

.logon localtd/td07

.logon localtd/td07

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.00.00.12

\*\*\* Teradata Database Version is 13.00.00.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Character Set Name is 'ASCII'.

\*\*\* Echo accepted.

\*\*\* Total elapsed time was 1 second.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

show table tdhari.emp1\_dup4;

show table tdhari.emp1\_dup4;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------------

CREATE SET TABLE tdhari.emp1\_dup4 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2))

PRIMARY INDEX ( EMP\_NO );

BTEQ -- Enter your DBC/SQL request or BTEQ command:

select \* from tdhari.emp1\_dup4;

select \* from tdhari.emp1\_dup4;

\*\*\* Failure 3523 The user does not have SELECT access to tdhari.emp1\_dup4.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your DBC/SQL request or BTEQ command:

help table tdhari.emp1\_dup4;

help table tdhari.emp1\_dup4;

\*\*\* Help information returned. 4 rows.

\*\*\* Total elapsed time was 1 second.

Column Name Type Comment

------------------------------ ---- ------------------

EMP\_NO I ?

DEPT\_NO I ?

FIRST\_NAME CV ?

SALARY D ?

BTEQ -- Enter your DBC/SQL request or BTEQ command:

**Granting Multiple Privileges With a Single Keyword**

Teradata Database has a special category of keywords that you can use to grant multiple privileges. For example, the following GRANT request grants both the CREATE DATABASE and DROP DATABASE privileges to user *df2* using the keyword DATABASE:

     GRANT DATABASE on df2;

The following table lists this class of keywords and indicates the multiple privileges that they confer when they are granted on a user or database:

|  |  |
| --- | --- |
| **The following keyword …** | **Indicates the following privileges …** |
| ALL | all implicit and explicit object privileges owned by the grantor WITH GRANT OPTION that pertain to the type of object specified, and *only* those privileges, are granted on the specified database object. |
| AUTHORIZATION | **•**CREATE AUTHORIZATION  **•**DROP AUTHORIZATION |
| CHECKPOINT | **•**ability to execute the CHECKPOINT SQL statement.  **•**ability to execute the HUT CHECKPOINT command. |
| DATABASE | **•**CREATE DATABASE  **•**DROP DATABASE |
| DROP | **•**If the ON clause specifies FUNCTION, then the privilege granted is DROP FUNCTION.  **•**If the ON clause specifies PROCEDURE, then the privilege granted is DROP PROCEDURE. |
| EXECUTE | **•**If the ON clause specifies FUNCTION, then the privilege granted is EXECUTE FUNCTION.  **•**If the ON clause specifies PROCEDURE, then the privilege granted is EXECUTE PROCEDURE.  **•**If the ON specifies nothing, then the privilege granted is EXECUTE MACRO. |
| FUNCTION | **•**CREATE FUNCTION  **•**DROP FUNCTION |
| GLOP | **•**CREATE GLOP  **•**DROP GLOP |
| INDEX | **•**CREATE INDEX  **•**DROP INDEX |
| MACRO | **•**CREATE MACRO  **•**DROP MACRO |
| PROCEDURE | **•**CREATE PROCEDURE  **•**DROP PROCEDURE |
| PROFILE | **•**CREATE PROFILE  **•**DROP PROFILE |
| RESTORE | ability to execute the following HUT commands on the specified object:  **•**DELETE JOURNAL  **•**ROLLBACK  **•**ROLLFORWARD |
| ROLE | **•**CREATE ROLE  **•**DROP ROLE |
| SHOW | the ability to execute the following SQL statements *only*:  **•**HELP *database\_object*  **•**SHOW *database\_object* |
| TABLE | **•**CREATE TABLE  **•**DROP TABLE |
| TRIGGER | **•**CREATE TRIGGER  **•**DROP TRIGGER |
| USER | **•**CREATE USER  **•**DROP USER |
| VIEW | **•**CREATE VIEW  **•**DROP VIEW |

**Differences Between GRANT and GIVE**

The GRANT statement is used only to assign specific *privileges*, while the GIVE statement transfers the *ownership* of a database or user to another database or user.

**Granting Logon Privileges**

The following conditions should exist to use the GRANT LOGON and REVOKE LOGON statements:

* DIPVIEWS has been run from the DIP utility to create the DBC.LogonRule macro.
* A Teradata Database security administrator user has been created; for example, username SecAdmin.
* User SecAdmin has been granted the EXECUTE privilege on DBC.LogonRule.

If these conditions exist, the security administrator can execute the GRANT LOGON or REVOKE LOGON statements any time after installation to add or remove user names on individual host connections as needed.

|  |  |
| --- | --- |
| **Statement** | **Comments** |
| GRANT LOGON | Gives users permission to log on to Teradata Database from specific client systems using a pre-validated logon request.  To execute a GRANT LOGON statement, you must hold execute privileges on the macro DBC.LogonRule. |
| REVOKE LOGON | Retracts permission to log on to Teradata Database from specific client systems.  After installation, use the REVOKE LOGON statement to change the system default by first removing logon privileges from all users from all hosts. Then, you can submit the GRANT LOGON statement to assign individual users to specific host IDs. |

To change the system default:

1. Submit the REVOKE LOGON statement to remove logon privileges from all users from all hosts.
2. Submit the GRANT LOGON statement to assign individual users to specific host IDs.

The GRANT LOGON and REVOKE LOGON statements store rows in the DBC.LogonRuleTbl.

You can also use Teradata Administrator to grant logon privileges.

select \* from DBC.LogonRuleTbl;

select \* from DBC.LogonRuleTbl;

\*\*\* Query completed. 5 rows found. 8 columns returned.

\*\*\* Total elapsed time was 1 second.

UserID LogicalHostID LogonStatus NullPassword CreateUID CreateTimeStamp LastAccessTimeStamp AccessCount

-------- ------------- ----------- ------------ --------- ------------------- ------------------- --------------

0000EE03 0 G T 00000100 2009-08-31 18:05:40 ? ?

0000EB03 0 G T 00000100 2009-08-31 18:02:38 ? ?

0000EA03 1024 G T 00000100 2009-08-31 18:02:38 ? ?

0000EC03 0 G T 00000100 2009-08-31 18:02:40 ? ?

0000EF03 0 G T 00000100 2009-08-31 18:05:45 ? ?

BTEQ -- Enter your DBC/SQL request or BTEQ command:

**What is logon privilege in Teradata?**

When a user is created in the database it automatically gets database logon privileges, and can logon from any configured, connected client using Teradata authentication (TD2 mechanism).

**By default, the database automatically grants permission to log on for all users defined in the database, from all client system connections (hostids).**

You can use the REVOKE LOGON statement to restrict:

* All logons to the database for a particular user
* Logons to the database by a user through one or more client connections (hostids).

Example :

GRANT/REVOKE LOGON ON hostid1,hostid2 FROM username1,username2;

or

GRANT/REVOKE LOGON ON All FROM username1,username2;

where hostid corresponds to a host group (HostNo value).

• GRANT LOGON statement:– Gives users permission to logon to Teradata from specific client connections and optionally use a pre-validated logon request.

GRANT LOGON ON ALL - TO username WITH NULL PASSWORD ;

ALL : The ALL keyword, used in place of a host ID, applies to any source through which a logon is attempted, including the Teradata database console.

WITH NULL PASSWORD : The initial Teradata database default is that all logon requests must include a password. The WITH NULL PASSWORD option, in conjunction with a TDP security exit procedure,permits a logon string that has no password to be accepted on a Teradata system.

Grant logon on all to bmk with null password;

Grant logon on all to bmk with null password;

\*\*\* Logon has been granted.

\*\*\* Total elapsed time was 1 second.

Grant all on kmr to bmk;

\*\*\* Grant accepted.

\*\*\* Total elapsed time was 1 second.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Creating Tables

Creating a table requires:

• Defining columns

• Assigning a primary index

• Optional assignment of secondary indexes

There are two types of table creation.

SET- its not allowing duplicate rows.

MULTISET- its allowing duplicate rows.

CREATE TABLE emp (

eno INTEGER,

name VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,

sal INTEGER,

comm INTEGER,

netsal INTEGER)

Unique PRIMARY INDEX ( eno )

index (name);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Statement Processing:

There are two ways of Statement processing in BTEQ:

1. a single-statement request: at the end of the statement as the last nonblank character of the line.

2. a multi-statement request : at the beginning of the line that begins the second through the final statement, and at the end of the final

statement as the last nonblank character of the line.

Single-Statement Example

BTEQ submits the following statements to the Teradata Database as three single-statement requests:

SELECT \* FROM Employee;

DELETE FROM Employee WHERE Name =’Inglis C’ AND Empno = 10014;

SELECT Name FROM Employee;

Multistatement Example

To submit the same three statements as a multistatement request, enter:

SELECT \* FROM Employee

; DELETE FROM Employee WHERE Name = ’Inglis C’ AND Empno = 10014

; SELECT Name FROM Employee;

after creating the set table has to insert the values:

insert into emp (101, 'ab', 1000, 10, 1100);

;insert into emp (102, 'ac', 2000, 10, 2200);

;insert into emp (103, 'ad', 3000, 10, 3300);

;insert into emp (104, 'ae', 4000, 10, 4400);

;insert into emp (105, 'af', 5000, 10, 5500);

;insert into emp (106, 'ag', 6000, 10, 6600);

;insert into emp (107, 'ah', 7000, 10, 7700);

;insert into emp (108, 'ai', 8000, 10, 8800);

;insert into emp (109, 'aj', 9000, 10, 9900);

;insert into emp (110, 'ak', 10000, 10, 11000);

to verify the table emp:

select \* from emp

order by 1;

\*\*\* Query completed. 10 rows found. 5 columns returned.

\*\*\* Total elapsed time was 1 second.

eno name sal comm netsal

-----------------------------------------------------------------------------

101 ab 1000 10 1100

102 ac 2000 10 2200

103 ad 3000 10 3300

104 ae 4000 10 4400

105 af 5000 10 5500

106 ag 6000 10 6600

107 ah 7000 10 7700

108 ai 8000 10 8800

109 aj 9000 10 9900

110 ak 10000 10 11000

create one more table with the same definition and data.

create table dept as emp with data;

To verify the the table dept with the follwing sql:

select \* from dept

order by 1;

\*\*\* Query completed. 10 rows found. 5 columns returned.

\*\*\* Total elapsed time was 1 second.

eno name sal comm netsal

----------- --------------- ----------- ----------- -----------

101 ab 1000 10 1100

102 ac 2000 10 2200

103 ad 3000 10 3300

104 ae 4000 10 4400

105 af 5000 10 5500

106 ag 6000 10 6600

107 ah 7000 10 7700

108 ai 8000 10 8800

109 aj 9000 10 9900

110 ak 10000 10 11000

Add one or more column to the above table dept:

alter table dept

add deptno integer,

add department varchar(15);

updating the dept table using the follwing sql:

update dept

set deptno=1001,department='support'

where eno between 101and 105;

update dept

set deptno=1002,department='hr'

where eno between 106 and 110;

now verify the out put by the follwing sql

select eno,name,deptno,department,sal from dept

order by 1;

\*\*\* Query completed. 10 rows found. 5 columns returned.

\*\*\* Total elapsed time was 1 second.

eno name deptno department sal

------------------------ --------------- -----------------------------------------

101 ab 1001 support 1000

102 ac 1001 support 2000

103 ad 1001 support 3000

104 ae 1001 support 4000

105 af 1001 support 5000

106 ag 1002 hr 6000

107 ah 1002 hr 7000

108 ai 1002 hr 8000

109 aj 1002 hr 9000

110 ak 1002 hr 10000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Creating View:

CREATE VIEW dept1

as

SELECT \* from dept;

to verify the view dept1:

select \* from dept1;

\*\*\* Query completed. 10 rows found. 7 columns returned.

\*\*\* Total elapsed time was 1 second.

eno name sal comm netsal deptno

----------- --------------- ----------- ----------- ----------- -----------

110 ak 10000 10 11000 1002

101 ab 1000 10 1100 1001

106 ag 6000 10 6600 1002

108 ai 8000 10 8800 1002

102 ac 2000 10 2200 1001

104 ae 4000 10 4400 1001

109 aj 9000 10 9900 1002

107 ah 7000 10 7700 1002

103 ad 3000 10 3300 1001

105 af 5000 10 5500 1001

similerly to create the view from emp table

CREATE VIEW emp1

as

SELECT \* from emp;

to verify the view emp1:

select \* from emp1;

\*\*\* Query completed. 10 rows found. 5 columns returned.

\*\*\* Total elapsed time was 1 second.

eno name sal comm netsal

----------------------- --------------- ----------- ----------- -----------

110 ak 10000 10 11000

101 ab 1000 10 1100

106 ag 6000 10 6600

108 ai 8000 10 8800

102 ac 2000 10 2200

104 ae 4000 10 4400

109 aj 9000 10 9900

107 ah 7000 10 7700

103 ad 3000 10 3300

105 af 5000 10 5500

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Creating and Using Macros

Additional Database Privileges : You must have the following additional database access privileges, which are granted by the owner of the database,

to create or use a macro:

• The CREATE MACRO privilege.

• Access to the appropriate database elements. (You cannot create a macro to access a database table that you do not have permission to access.)

Also, after you have created a macro, you must grant EXECUTE privileges to any other users before they can

execute your macro.

Executing Commands

BTEQ executes the commands and SQL statements in the same order that you enter them.You must enclose each BTEQ command in a

Teradata SQL ECHO statement. Then, when you

execute the macro, the Teradata Database executes the Teradata SQL statements and returns the contents of the ECHO statements

—your BTEQ commands—to BTEQ.

Warning: It is suggested that you echo BTEQ commands only in Field mode. You will not get the behavior

you might expect when you attempt to echo commands to BTEQ using other response modes.

You will not receive error messages, plus the echoed text may be dumped rather than executed.

Creating a Macro

To create a macro, enter the Teradata SQL CREATE MACRO and ECHO commands in an interactive BTEQ session as follows:

CREATE MACRO macroname AS

( ECHO ’.BTEQcommand’

;Teradata SQLrequest;

);

The distribution of Teradata SQL requests and BTEQ commands across the command line is

arbitrary. The following syntax, for example, performs the same function, but requires additional punctuation:

CREATE MACRO macroname AS

(ECHO ’.BTEQ command’

;Teradata SQLrequest’;);

Macro Example

To create a macro named deptdisplay, for example, that includes the following BTEQ

SEPARATOR command and Teradata SQL SELECT statement:

.SET SEPARATOR ’ | ’

SELECT \* FROM department;

enter:

CREATE MACRO deptdisplay AS

( ECHO ’.SET SEPARATOR ’’ | ’’ ’

;SELECT \* FROM department;

);

Note that when you enclose single quotes in quotes, you double the inner quotes—each inner single quote becomes two single quotes.

After creating your macro, BTEQ displays:

\*\*\* Macro has been created.

\*\*\* Total elapsed time was 3 seconds.

Note: Develop more complex macros by placing them in a file and running the file in a BTEQ batch job.

Granting the Macro EXECUTE Privilege

As the creator of a macro, you automatically receive the execute privilege on the macro. To

successfully execute the macro, however, you must have been granted access privileges to all of

the objects named in the macro. For example, to execute the deptdisplay macro that you

created in the prior subsection, you must have the select privilege on the Department table.

Similarly, anyone that you grant permission to execute the deptdisplay macro must also have

the same privileges.

Use the Teradata SQL GRANT statement to allow other users to execute your macro:

GRANT EXECUTE ON macroname TO userid;

To grant the execute privilege to user jnw on the macro deptdisplay, enter:

GRANT EXECUTE ON deptdisplay TO jnw;

Executing Macros

You can execute macros either interactively, or include them in BTEQ scripts or input stream

files.

To execute a macro interactively, use the Teradata SQL EXECUTE statement:

EXECUTE macroname;

To execute the macro deptdisplay, enter:

EXECUTE deptdisplay;

Including a Macro in a Script

Use the editor on your system to include a BTEQ macro in script or input stream file. To

execute the deptdisplay macro from a script, for example, enter the Teradata SQL EXECUTE

statement after the LOGON statement in the script:

.LOGON tdpid/userid, password

EXECUTE deptdisplay;

.LOGOFF

Creating Reports with Macros

You can create a report from a macro by instructing BTEQ in your prepared script or input

stream file to export the data returned as a report. To do this, you can either:

• Create your macro on the Teradata Database and execute it from the script file. In this case,

the macro would contain the appropriate BTEQ commands to generate a report, and your

script file would contain the BTEQ EXPORT command, the Teradata EXECUTE macro

statement, and other BTEQ commands. This is the preferred method.

• Place your macro in the script file and create and execute it from there. This method

creates the macro every time it executes the request—a major disadvantage.

Chapter 3: Using BTEQ

Using Scripts, Run Files, Macros, and Stored Procedures

46 Basic Teradata Query Reference

Macro Example

The following is an example of a macro to produce a report called Department Information:

CREATE MACRO deptdisplay AS (

ECHO ’.SET SEPARATOR “ | ” ’;

ECHO ’.SET RTITLE “Department Information” ’;

ECHO ’.SET FORMAT ON’;

SELECT \* FROM department;);

Teradata extensions--Macro

Simple macro to insert data in to a table or view.

create macro mcr\_emp ( eno INTEGER,

name VARCHAR(15) ,

sal INTEGER,

comm INTEGER,

netsal INTEGER )

AS (

insert into emp

(eno,name,sal,comm,netsal)

Values

(:eno

,:name

,:sal

,:comm

,:netsal);

/\* the follwing select verifies the insert\*/

select \* from emp where eno=:eno;);

if executing the above macro the data is inserted into emp table and we will seen the output result

exec mcr\_emp ( 111,'ba',11000,10,12100);

\*\*\* Insert completed. One row added.

\*\*\* Total elapsed time was 1 second.

\*\*\* Query completed. One row found. 5 columns returned.

eno name sal comm netsal

----------- --------------- ----------- ----------- -----------

111 ba 11000 10 12100

to view the emp table :

select \* from emp order by 1;

\*\*\* Query completed. 11 rows found. 5 columns returned.

\*\*\* Total elapsed time was 1 second.

eno name sal comm netsal

----------- --------------- ----------- ----------- -----------

101 ab 1000 10 1100

102 ac 2000 10 2200

103 ad 3000 10 3300

104 ae 4000 10 4400

105 af 5000 10 5500

106 ag 6000 10 6600

107 ah 7000 10 7700

108 ai 8000 10 8800

109 aj 9000 10 9900

110 ak 10000 10 11000

111 ba 11000 10 12100

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BTEQ COMMAND SET:

1.SESSION CONTROL COMMAND:

START a SESSION

.LOGON - Start a BTEQ session

.SESSIONS - specify the number of sessions to use with the next LOGON command

.TDP - specify the teradata server for subsequent logons during the current session.

.LOGONPROMPT - bypass the warnings related to conventional LOGON command use

END a session

.LOGOFF - end the current session without exiting the BTEQ

.EXIT or QUIT - End the current session and exit BTEQ

.ABORT - Abort any active request and transactions without exiting BTEQ

.HALT EXECUTION - abort any active request and transactionsand exit BTEQ

DISPLAY information about SESSION

.SHOW CONTROLS - display the current configaration of the BTEQ control commands

.SHOW VERSIONS - display the BTEQ version number,module version number,linkage date

Specify SESSION characteristcs

.SESSION CHARSET - specify the name of a character set for the current session

.SESSION SQLFLAG - specify the disposition of the warnings issued in the response to violation of ANSI-complaint syntax

.SESSION TRANSACTION - specify the transaction boundaries are determined by Teradata sql symantics or ANSI symantics

.COMPILE - create or replace a TERADATA stored procedure

2.FILE CONTROL COMMAND:

.RUN - execute teradata SQL request and BTEQ commands from a specified run file

.REPEAT - submits the next request a specified number of times

.FORMAT - enable or inhibit the page oriented format command options

.IMPORT - open a file with a specific format to transfer information to the teradata database

.EXPORT - open a file with specific format transfer information from the teradata database

.INDICDATA or RECORDMODE - specify the response mode,either field mode,indicator mode,record mode,or multi indicator mode,data selected from the

teradata database

3.SEQUENCE CONTROL COMMAND:

.ERRORLEVEL - assigns severirty levels to errors

.IF-THEN - test the validity of the condition stated in the if clause -- .IF ERRORCODE,.IF ACTIVITYCOUNT,.IF ERRORLEVEL

.REMARK - places a specified string on the standard output stream

.LABEL - identifies the point at whichBTEQ resumes processing,as specified in a previous GOTO command

.HANG - pause BTEQ processing for aspecified period of time

.GOTO - skips over all intervining BTEQ commands and sql statements until as specified label is encountered,resumes processing in

sequence.

.MAXERROR - designates a maximum error severity level beyond which BTEQ terminates job processing.

4.FORMAT CONTROL COMMANDS:

.DEFAULTS - reset the BTEQ format command options to their default configurations

.ECHOREQ - enable the echo required function that returns a copy of each Teradata SQL request and BTEQ command to the standard output

stream

.FOLDLINE - split (fold) each line of a report into two or more lines

.FOOTING - specify a footer to appear at the bottom of every page of a report

.HEADING or RTITLE - specify a header to appear at the top of every page of a report

.NULL - specify a character or character string to represent null field values returned from the Teradata Database

.OMIT - Excludes specified columns returned from SQL SELECT statements

.PAGEBREAK - Ejects a page whenever the value for one or more specified columns changes

.PAGELENGTH - specify the page length of printed reports, in lines per page

.RETCANCEL - cancel a request when the value specified by the RETLIMIT command ROWS option is exceeded

.RETLIMIT - specify the maximum number of rows and/or columns displayed or written in response to a Teradata SQL request

.SEPARATOR - specify a character string or width (in blank characters) to separate columns of a report

.SIDETITLES - position summary titles to the left of the summary lines in a report

.SKIPDOUBLE - insert two blank lines in a report whenever the value of a specified column changes

.SKIPLINE - insert a blank line in a report whenever the value of a specified column changes

.SUPPRESS - replace all consecutively repeated values with all-blank character strings

.TITLEDASHES - display a row of dash characters before each report line summarized by a WITH clause

.UNDERLINE - display a row of dash characters whenever the value of a specified column changes

.WIDTH - specify the width of screen displays and printed reports, in characters per line

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RUNNING BTEQ SCRIPT:

1. RUNNING WITH BTEQ CONDITIONAL LOGIC:

.logon demotdat/bmk,mkb

DELETE from emp\_dept all;

.IF ERRORCODE = 0 THEN .GOTO mohan

CREATE TABLE

( eno integer

, name varchar(15)

, sal integer

, comm integer

, netsal integer

, deptno integer

, department varchar(15));

.LABEL mohan

INSERT INTO emp\_dept emp\_dept

SELECT e.eno

, e.name

, e.sal

, e.comm

, d.netsal

, d.deptno

, d.department

FROM emp e INNER JOIN dept d

ON e.eno=d.eno

where e.comm >5;

.IF ACTIVITYCOUNT >0 THEN .GOTO CONTINUE

.logoff

.label continue

the above script save it in file.

now i'm saving in mpt.txt in D drive.

now i'm running this job using BTEQ RUN command

.RUN FILE=D:/mpt.txt

To verify the output:

select \* from emp\_dept;

\*\*\* Query completed. 10 rows found. 7 columns returned.

\*\*\* Total elapsed time was 1 second.

eno name sal comm netsal deptno

----------------- ----------- --------------- ----------- ----------- ---

110 ak 10000 10 11000 1002

101 ab 1000 10 1100 1001

106 ag 6000 10 6600 1002

108 ai 8000 10 8800 1002

102 ac 2000 10 2200 1001

104 ae 4000 10 4400 1001

109 aj 9000 10 9900 1002

107 ah 7000 10 7700 1002

103 ad 3000 10 3300 1001

105 af 5000 10 5500 1001

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2. BTEQ EXPORT SCRIPT:

There are 4 types of BTEQ EXPORT

1. Field Mode (REPORT) - When submitting BTEQ requests to a Teradata database, you may have noted that output is

always provided with column headings and underscores, with numerics aligned to the right,

characters to the left, and all output displayed in the center of the screen or report. This is

Field mode, the default output of BTEQ (suitable for reports).

REPORT – output is truncated to 254 characters

REPORTWIDE – output is truncated to 32 KB (only supported in some releases)

SCRIPT:

.LOGON demotdat/bmk,mkb

.EXPORT REPORT FILE=d:/exrt.txt

SELECT eno

, name

, sal

, comm

, netsal

, deptno

, department

FROM emp\_dept

WHERE comm > 5;

.EXPORT RESET

.LOGOFF

The above script save it in export report mode.txt

To run the script using .RUN FILE=d:/export report mode.txt

The data will exported to the defining file called exrt.txt in D drive.

The output as foolows:

eno name sal comm netsal deptno

-------- --------------- ----------- ----------- ----------- -----------

110 ak 10000 10 11000 1002

101 ab 1000 10 1100 1001

106 ag 6000 10 6600 1002

108 ai 8000 10 8800 1002

102 ac 2000 10 2200 1001

104 ae 4000 10 4400 1001

109 aj 9000 10 9900 1002

107 ah 7000 10 7700 1002

103 ad 3000 10 3300 1001

105 af 5000 10 5500 1001

2.Record Mode (DATA) - You might require output data in a flat-file format with binary data, no headings, etc.

Request output in this format by using Data mode.

To get the data in the above script use the DATA instead of REPORT and save it in expt text file.

the path of output file is export data mode text file.

TO RUN the script using .RUN FILE=d:/expt.txt

The data in export data mode text file as follows:

n ak'

ø\* ê hr

! e abè

L é support

j agp

Èê hr

l ai @­

`" ê hr

! f acÐ

˜é support

! h

ae

0é support

m aj(#

¬& ê hr

k ahX

‑ ê hr

! g ad¸

ä é support

! i afˆ

| é support

3.INDICDATA - Host computer systems rarely have the built-in capability to recognize or handle NULL data.

You might need to use INDICDATA if the data contains NULL columns.

4.Data Interchange Format (DIF) - Use the DIF output option if you need data in a format suitable for PC-based applications

such as VISICALC and Lotus 1-2-3. The DATALABELS option includes the column titles

of the selection results as the first row in the DIF file.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3.BTEQ IMPORT :

SCRIPT:

.LOGON demotdat/bmk,mkb

.IMPORT DATA file = d:/export data mode.txt ;

.QUIET ON

.REPEAT \*

USING eno (integer)

, name (varchar(15))

, sal (integer )

, comm (integer)

, netsal (integer)

, deptno (integer)

, department (varchar(15))

INSERT INTO test\_emp\_dept

(

eno

, name

, sal

, comm

, netsal

, deptno

, department

)

VALUES (

:eno

, :name

, :sal

, :comm

, :netsal

, :deptno

, :department

)

;

.LOGOFF

This above script save it in imp.txt file

Here also we can import in different mode like data mode ,record mode,vartext etc

To Run this script using ,RUN FILE=d:/imp.txt

We can verify the output by the follwing SQL

select \* from emp\_dept order by 1;

\*\*\* Query completed. 10 rows found. 7 columns returned.

\*\*\* Total elapsed time was 1 second.

eno name sal comm netsal deptno

------------------------------------------------------------------------------------------

101 ab 1000 10 1100 1001

102 ac 2000 10 2200 1001

103 ad 3000 10 3300 1001

104 ae 4000 10 4400 1001

105 af 5000 10 5500 1001

106 ag 6000 10 6600 1002

107 ah 7000 10 7700 1002

108 ai 8000 10 8800 1002

109 aj 9000 10 9900 1002

110 ak 10000 10 11000 1002

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Stored Procedure Compilation**

The Stored Procedure Language (SPL) compilation errors and warnings are reported by Teradata Database as part of the SUCCESS/OK parcel in response to the COMPILE command.

All syntax and semantic errors in the stored procedure source text are reported with a nonzero warning code. The activity count is set to the total number of compilation errors and warnings.

Teradata Database server cannot report more than 64 KB of SPL compilation errors and warnings. If they exceed the limit, the server returns only the first 64 KB of errors and warnings, and ignores all remaining errors. If the last error or warning text does not fit in the 64-KB buffer, it is eliminated. A message is given that indicates too many compilation errors have occurred.

The stored procedure is not created or replaced if compilation errors are found. Instead, the stored procedure is created or replaced if compilation warnings are found, and no failures are reported.

\* \* \* END \* \* \*

\* \* \* \*\*\* \* \* \*