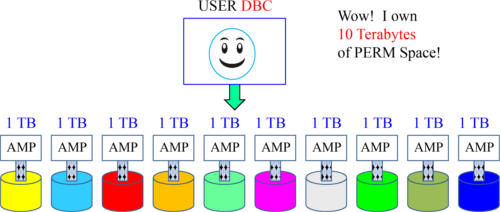
Chapter 8 – The User Environment

“The afternoon knows what the morning never suspected.”

– Swedish Proverb

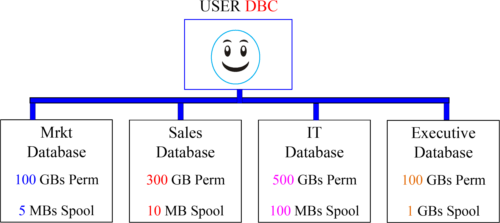
DBCis the only user when the system first arrives



When the system is new and arrives at your company, DBC is the only USER . DBC counts up all the disk space attached to each AMP and considers that PERM space owned by DBC .

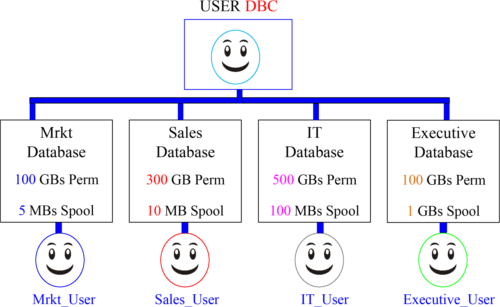
User DBC owns all the disk space on day one of your system’s arrival. DBC will then begin to allocate space to other databases or users. Think of PERM space like money. If DBC has 10 Terabytes of space it is like having 10 dollars. If you give away 5 dollars, you only have 5 dollars left. Spool space is more like a speed limit.

DBCwill Create Databases and Give them Space



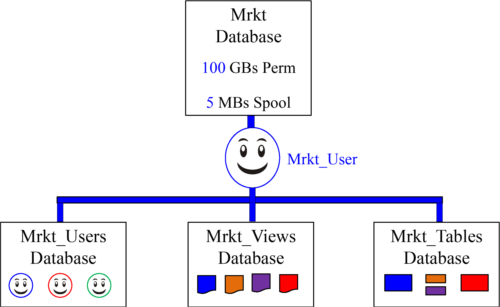
DBC will begin the hierarchy. Anyone above you is your parent, and anyone below you is your child. DBC is always a parent of every database and user in the system. If DBC started with 10 Terabytes, then DBC only has 9 Terabytes left because DBC gave Mrkt, Sales, IT, and the Executive database some of the space.

DBCwill create some initial Users



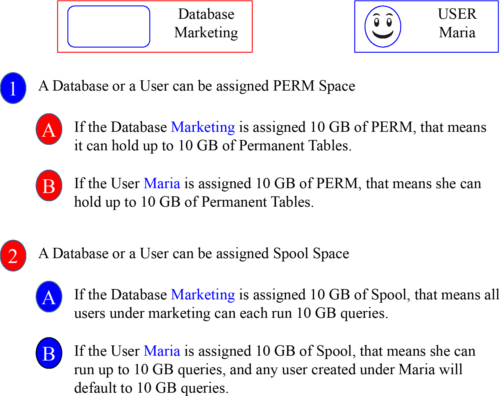
Now that the user DBC has created the databases and some initial users, it can rest. Each user can now create other users and other databases within their own hierarchy. Each database has the space and each user has the control to build their own environment of users, databases and tables. The hierarchy is just beginning.

ATypical Teradata Environment



Users in the Mrkt\_Users database are given Select and Execute Grants to the Mrkt\_Views. The Mrkt\_Views database holds the views that access the Mrkt\_Tables database. This provides security and allows the users to access the information.

Whatare Similarities between a DATABASE and a USER?

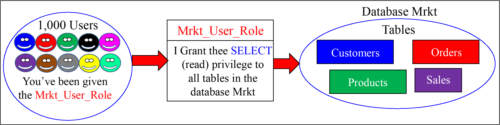


In Teradata, the only difference between a database and a user is that a user can login and run queries.

Roles

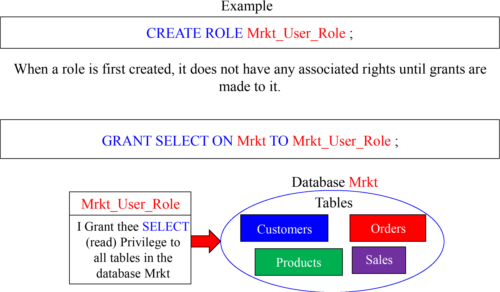
Roles simplify database administration by assigning access rights to tables and other objects, and then groups of people with similar job functions (or roles) can access these objects.

It is as simple as creating different roles for different job functions and responsibilities, and then granting specific privileges (access rights) on database objects to these roles, then granting a role or roles to users who share the same privileges.



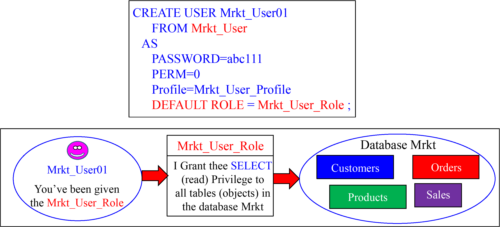
Imagine a thousand users accessing certain databases, tables, views, macros, or stored procedures and having to grant each individual user access rights to these objects. Why not, instead, create a role that provides access to these objects, and then assign everyone needing this access that role. Roles simplify this very important security assignment. This is the way that Teradata prefers to implement Access Rights.

Createa Role and then Assign that Role Its Access Rights



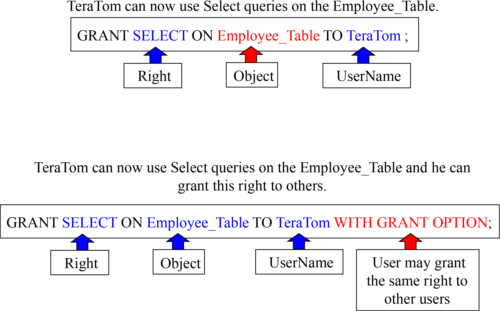
We first create a role and it has no associated rights with it. Then, we grant the rights on the database or objects we want to that role. You can use the Grant or Revoke commands to add or take away rights for a role. Above, we created a role name Mrkt\_Users\_Role, and then we granted SELECT access to all objects in that database.

Createa User and Assign them a Default Role



Let’s recap. We gave Mrkt\_User the right to create and drop roles. Then Mrkt\_User created a role called Mrkt\_User\_Role. Then we assigned the SELECT right to the Mrkt database to the Mrkt\_User\_Role. In our example above, we created a user named Mrkt\_User01 and assigned them a default role of Mrkt\_User\_Role. Now our user (Mrkt\_User01) can run SELECT statements (queries) on all tables (objects) in the Mrkt database.

GrantingAccess Rights



These examples are designed to teach you the mere basics of rights on objects.

Thereare Three Types of Access Rights

Access Rights consist of Automatic Rights, Implicit Rights, and Explicit Rights. Here is an explanation of each:

1. Automatic Rights are often referred to as Default Rights because these are privileges that are automatically given to creators and, in the case of users and databases, each of their created objects.

2. Implicit Rights are sometimes called Ownership Rights. Anybody above you in the Teradata hierarchy is considered your parent or owner, and a parent has an implied right over their children. So, these Owners (Parents) have the implicit right to grant rights on their owned objects (Children), either to themselves or to any other user or database. Ownership rights cannot be taken away unless ownership is transferred.

3. Explicit rights are privileges a user explicitly grants to another user with an actual GRANT statement. This statement inserts new rows into the DBC.AccessRights table. Explicit rights can be removed by explicitly using the REVOKE statement.

When a user submits a CREATE statement, new rows are inserted in the table DBC.AccessRights and they are removed for an object if it is dropped.

Descriptionof the Three Types of Access Rights



Implicit rights belong to the owners of objects. Owners don’t require rows in the DBC.AccessRights table to grant privileges on owned objects. Ownership rights can’t be revoked . A parent or owner has the implied right to GRANT privileges over their children. DBC and SysDBA hold implicit roles on all the other databases above.

Automatic rights happen whenever a CREATE statement is submitted, and new rows are automatically added to the DBC.AccessRights table. When the databases above were created, they automatically received all but four access rights on themselves. Automatic rights are removed with REVOKE or DROP statements.

Explicit rights are completely controlled by when a user explicitly and literally submits a GRANT or REVOKE statement. The GRANT statement adds new rows to the DBC.AccessRights table, and the rights are granted. The REVOKE removes them.

Profiles

It is easy to get mixed up between a Profile and a Role. Think of it as “Profiles are for People” and “Roles are for Rights”.

Profiles define system attributes for users (people). By assigning a profile to a group of users, you can rest assured that all group members will operate under a common set of attributes.

Create a different profile for each user group based on system attributes that members share.

If you do set the value of a parameter in the profile, the settings override the settings for the user in a CREATE USER or MODIFY USER statement.

If you do not set the value of a parameter, the system uses the setting defined for the user in a CREATE USER or MODIFY USER statement.

Profile parameters include:

•Account IDs

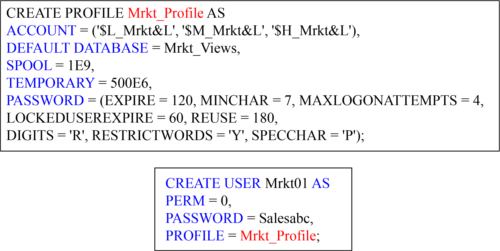
•Default database

•Spool and Temp space (but not Perm)

•Password attributes

Imagine a thousand users assigned the same amount of spool and temp space, the same default database, and the same account ID. With them all under the same profile, you can make one profile change and all thousand users receive the change. Imagine the nightmare of having to make a change to all thousand users if a profile was not used.

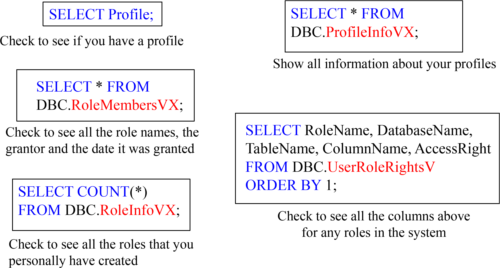
Creatinga Profile and a User



We have just created a profile. We have given them three account options, but the first account listed is the default. Each user assigned this profile will have a default database as Mrkt\_Views, and each gets 1,000,000,000 Bytes of spool space and 500,000,000 Bytes of temp space. We have also set quite a few password restrictions.

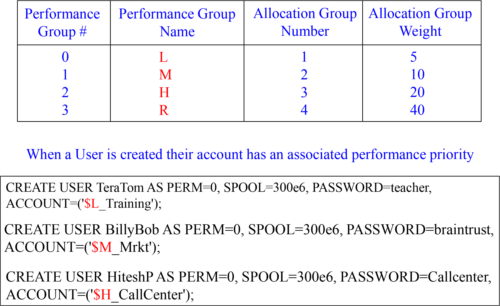
We have also created a user and assigned them to our newly created profile.

ProfileInfoVX,RoleMembers, RoleInfo and UserRoleRights



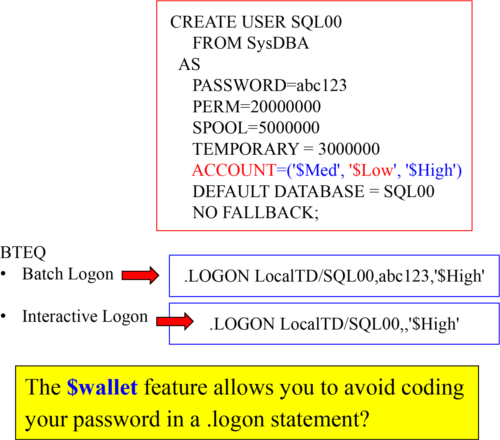
The ProfileInfoVX, RoleMembers, UserRoleRights and RoleInfoVX views show you what you need to know about Roles and Profiles. Remember that Roles are for "Rights" and Profiles are for "People".

Accountsand their Associated Priorities



Each user is given an Account ID and that will determine their system priority and this will be one of the ways the DBAs track the user. In our example above, BillyBob queries run twice as fast as Tera-Tom’s queries. HiteshP queries will run twice as fast as BillyBob’s queries and four times as fast as TeraTom’s queries.

Creatinga User with Multiple Account Priorities



Notice that when this user was created they also created three different Account IDs. The default Account ID will always be the first one listed, but now the user could login or set their session to utilize one of the other Account IDs. This is done so a user can run queries at different priorities.

AccountString Expansion (ASE)

&L This causes the logon time stamp to be inserted into the account string.

&D This causes the date to be inserted into the account string.

&T This inserts the time of day into the account string. This variable allows for one-second granularity, thus causing a row to be written for each individual SQL request.

&H This inserts the hour of the day into the account string.

&I This inserts the logon host ID/session number/request number into the account string.

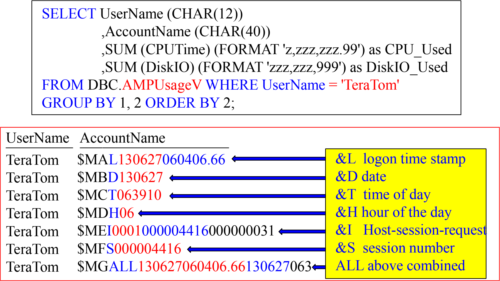
&S This inserts the current session number into the account string.



The system truncates all characters to the right of the 30th position for Account.

Account String Expansion will place additional information inside your account. After creating TeraTom, we will run queries from all accounts and check out the DBC.AmpUsage report, to see exactly how the account string expansion works.

TheDBC.AMPUsage View



The DBC.AMPUsage report is how users will be tracked in terms of how much CPU and Disk I/O usage they are using on the Teradata system. This example shows how Teradata reports the AccountName when the Account String Expansion (ASE) is used. I have highlighted in colors the values. This report didn’t show the CPU\_Used and DiskIO\_Used values that returned, but the DBAs will run this report to see who is using what resources.