Chapter 17 – Protection Features

"We're fools whether we dance or not, so we might as well dance."

-Japanese Proverb

*A List of the Protection Features*

Transient Journal

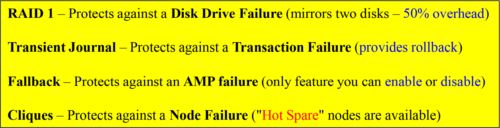
Fallback

RAID

Clique

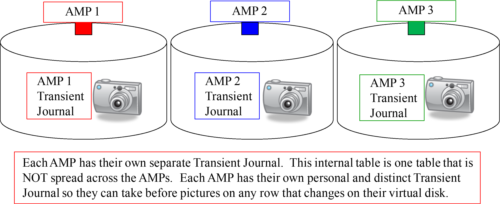
Permanent Journal

Archive Recovery Console (ARC)



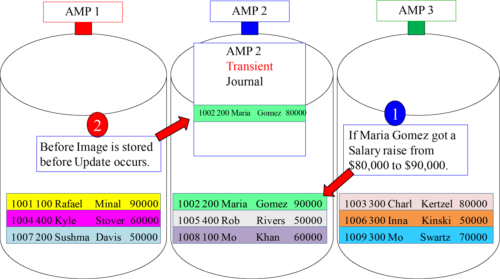
Above is a list of the major protection features of Teradata. Each is designed to serve a different purpose. Each comes with a cost, but some companies estimate that if their Teradata system is down it costs them about $1,000,000 (yes – million) an hour!

*Transient Journal Protects the Transaction Integrity*



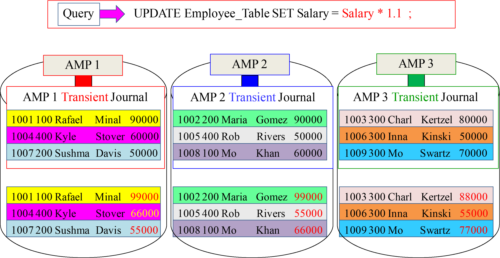
Database integrity is very important, especially when you are responsible for trillions of rows. When an AMP is going to make a change to a row, because of an SQL Update or Delete statement (for example), the AMP takes a before picture of the row that is about to change. This is for integrity purposes. If the Update or Delete is successful, the AMP will throw away the picture. If, however, there was a problem with the Update or Delete statement, then the AMP can Rollback the transaction to the way it was before. The Rollback can occur because of the before image.

*The Transient Journal in Action*



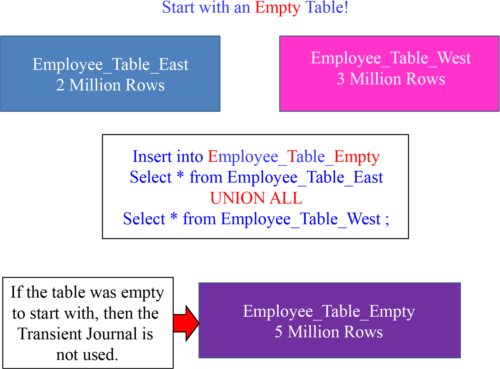
Notice that AMP 2 has opened its Transient Journal because an Update statement was issued to update Maria Gomez’s salary to $90,000. A before image of the row shows the old salary of $80,000. Also notice the other AMPs did NOT open their journal.

*A Single Transaction could Involve All AMPs*



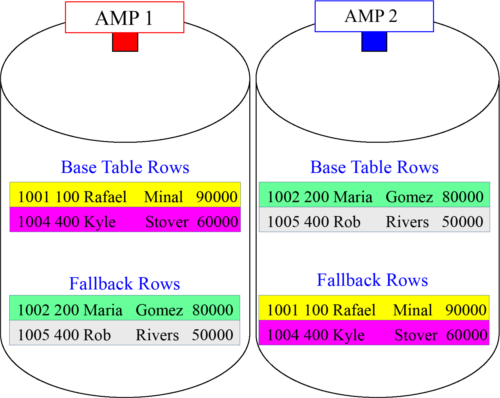
This transaction is considered one transaction, and everyone gets a salary raise (commit) or it will Rollback. The transaction will fail, and nobody will get the raise. Until the last AMP tells the Parsing Engine it is ready to commit the Transient Journal on each AMP stays, but after the commit each AMP empties their Transient Journal.

*The Secret to turning off the Transient Journal*



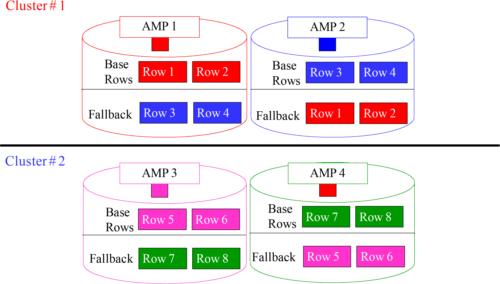
There is no Transient Journal if the table starts empty. That is because it is easy to Rollback an empty table. Only one row goes in the Transient to instruct the AMP to re-empty the table on any Rollback scenarios! The UNION ALL is Brilliant here!

*Fallback to Protect against an AMP Failure*



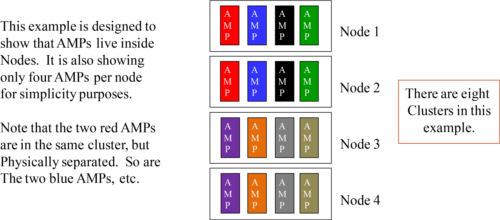
Fallback makes a duplicate copy of every row in a table and keeps that row on a different AMP. This is done in case we lose an AMP. If an AMP goes down, the system can still process the query because the rows on the failed AMP are also held by another AMP. The cost of Fallback is that the table is twice as big and uses twice the space. Remember, **Fallback** is a protection mechanism that can be **disabled**. The Transient Journal , Down AMP Recovery Journal and Raid 1 **cannot** be **disabled**.

*Fallback Clusters*



AMPs that are in the same Cluster protect each other. This way, you can lose one AMP per cluster. A better way to think of this in terms of Fallback is that each AMP has a buddy AMP where they both keep a backup of the other AMP’s rows on all Fallback protected tables. Two buddy AMPs who back each other AMP are in the same cluster.

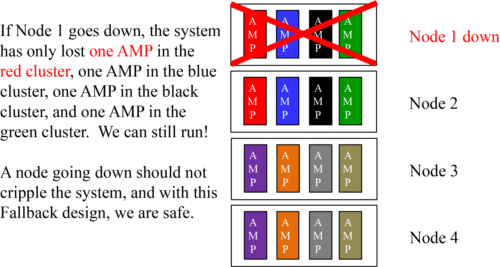
*AMPs in a Cluster are Physically Separated*



AMPs that are in the same Cluster protect each other. This way you can lose one AMP per cluster. There are now always two AMPs in a cluster, and each depends on the other if one of them goes down. When one AMP is down, its buddy AMP will read and write to the Fallback rows for the down AMP. They have each other's back!

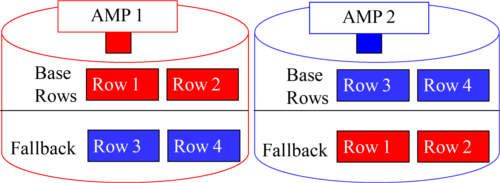
Here is the real surprise. The two AMPs in the cluster will be separated physically as far apart from each other as possible. In a well-designed system, no two AMPs in a cluster will ever be in the same node or clique. We will soon learn about cliques.

*The Reason AMPs in a Cluster are Physically Separated*



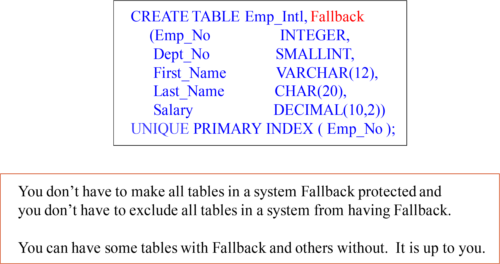
AMPs that are in the same Cluster protect each other. This way you can lose one AMP per cluster. The two AMPs in the cluster will be separated physically, as far apart from each other as possible. Notice that if we lost Node 1, the system would be able to still run because we only lost one AMP in the red cluster, one AMP in the blue cluster, one AMP in the black cluster, and one AMP in the green cluster. In a well-designed system, you should be able to lose an entire Node or Clique and still never lose more than one AMP in a paired buddy AMP cluster.

*The Price you pay for Fallback*



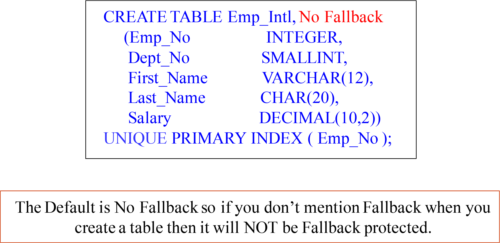
Fallback makes a duplicate copy of every row in a table and keeps that row on a different AMP. The cost of Fallback is that the table space is twice as big. If you have a secondary index on a fallback table, then the secondary index subtable is also fallback protected. So, its size is also doubled in perm space. Your loads and inserts, updates, and deletes take longer because the transaction or the load isn’t complete until the fallback rows are also updated. The price is big until an AMP goes down!

*How to Create a Table with Fallback*



Notice the Keyword Fallback in our Table Create statement. This table will be created with Fallback in mind. As each row is loaded, it is also be loaded to its Fallback AMP. The Fallback rows will never be used unless an AMP goes down, and then the Fallback rows are accessed only if the base rows are inaccessible because of the failed AMP.

*How to Create a Table with No Fallback*



Notice the Keywords No Fallback in our Table Create statement. This table will NOT be created with Fallback in mind. If you don’t specify Fallback, the default is usually No Fallback.

*How to Alter a Table to Add or Drop Fallback*

This is how you Alter a table and add Fallback

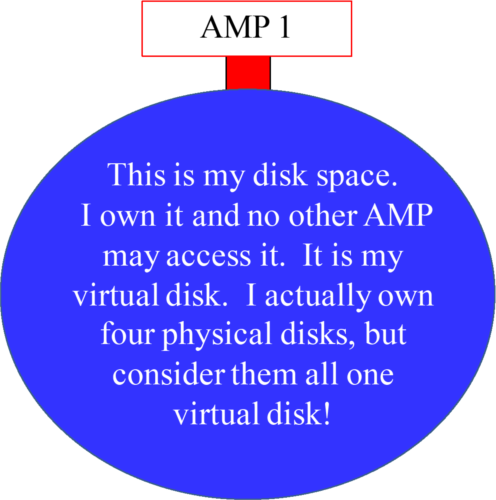


This is how you Alter a table and drop the Fallback



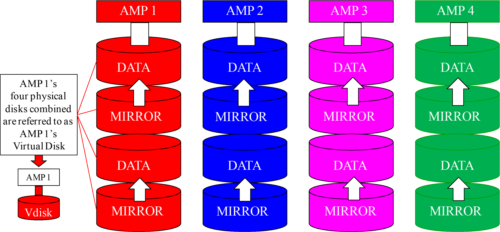
You can Alter a table at any time and add or drop the Fallback protection. After the command is entered, the table will either add or drop the fallback immediately depending on which command you entered.

*What is a Virtual Disk?*



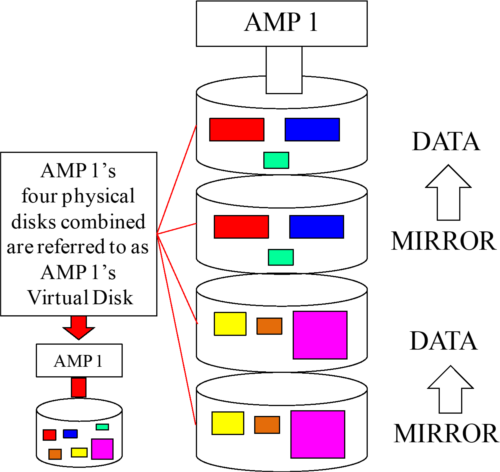
Each AMP has one virtual disk. Although an AMP is usually connected to four physical disks, that AMP is the only AMP allowed; accessing its disks and thus the term one virtual disk per AMP. Think of it like your home. Each person on your block may have four rooms in their home, but because each individual household owns their own home, we can call it one virtual room. For simplicity purposes, almost all diagrams show an AMP and its disk. That is its virtual disk, and only that AMP can access its virtual disk. Physically, that AMP is the only AMP that can access its four physical disks.

*Why do AMPs each have Four Physical Disks?*



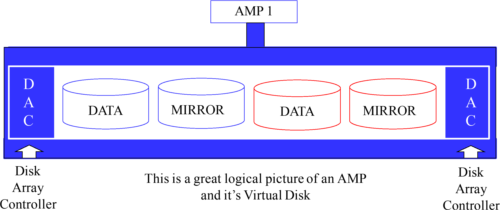
Each AMP has one virtual disk, but usually four physical disks. This concept is meant to understand a key point, and that is that an AMP owns its virtual disk and only that AMP can access it’s disks. The AMP uses two physical disks to store information and two disks to mirror in case of a disk failure. An AMP always has multiple disks, but no matter how many disks it has, it always has only one virtual disk.

*Is a Mirror just like Looking into a Mirror?*



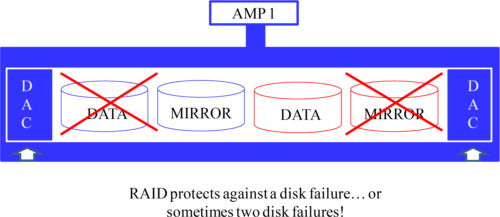
A mirror is an exact copy, so it is just like looking into a mirror. Notice above that our AMP has four disks, and the mirrors reflect an exact copy of the disk above. Also, notice that our Vdisk has all the blocks of data because a Vdisk is a logical representation of an AMPs disk farm.

*RAID 1 Mirroring – Redundant Array of Independent Disks*



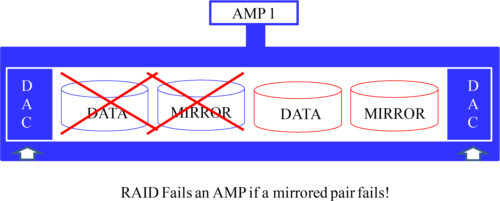
Almost every Teradata shop I have ever seen only uses Raid 1 Mirroring on each AMP. RAID 1 provides each AMP two disks for storing data and two disks for mirroring. The data disk and the mirror disk are called a mirrored pair. RAID 1 costs 50% of the disk space, but it ensures a 99% up time for customers. If a single disk goes down, it is easily replaced and Teradata isn’t even effected. The disaster happens when a mirrored pair fail! This is not Fallback. This is disk mirroring. They are separate concepts.

*What does RAID Protect?*



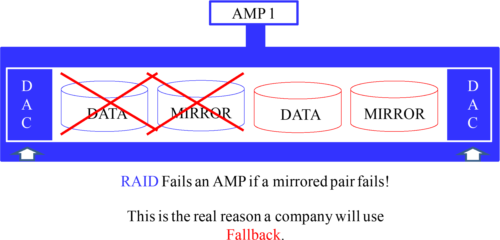
RAID protects against a disk failure. Each AMP could potentially lose two disks and still be up and running without interruption. In the example above, we have lost two disks, but our mirrored pairs are still considered up and running. Both failed disks will need to be replaced immediately, and the Teradata support team have been alerted. Disaster only strikes an AMP and their RAID disks if both a data disk and its mirror fail, but that is the real reason one chooses Fallback.

*How Does RAID Fail?*



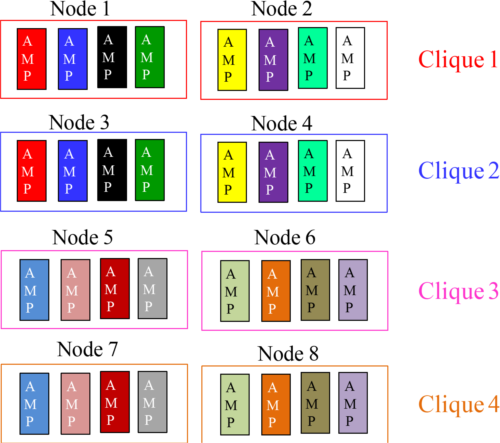
RAID protects against a disk failure. Each AMP could potentially lose two disks and still be up and running without interruption, but not in the example above. The AMP has lost a mirrored pair. This is a disaster unless you have Fallback.

*Do RAID and Fallback have a Connection?*



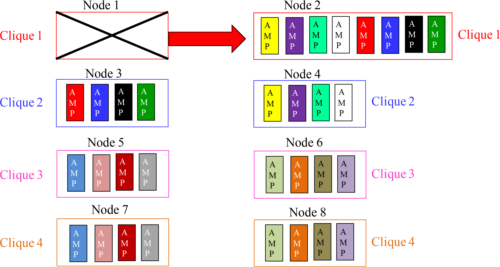
RAID protects against a disk failure. Fallback is supposed to protect an AMP failure, but AMPs don’t really fail any more. If they do, since they are virtual processes, they can usually be recovered quickly. Fallback is really designed to protect from the above example where an AMP loses a mirrored pair. If an AMP loses a mirrored pair it is considered down, but if the tables are Fallback protected, then the system uses the Fallback rows for any query involving the failed AMP. Fallback protects a disk disaster.

*What is a Clique?*



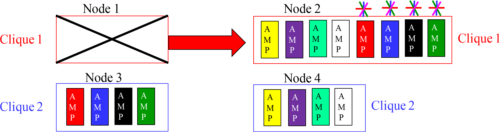
A clique protects against a node failure. The above picture is designed at the simplest level. You see two nodes in each clique. The purpose of grouping nodes into cliques is in case a node should fail the AMPs in the failed node can migrate to another node. Our next slide will demonstrate this concept.

*If a Node goes down the AMPs migrate within the Clique?*



A clique protects against a node failure. When Node 1 fails, all the AMPs inside the Node’s memory are down. Teradata does a quick reset and when the system restarts the AMPs in Node 1 migrate to Node 2. This is not permanent, and Node 2 is considered degraded, but all the AMPs are still functional. Stay tuned for more!

*Does Teradata Reset during a Node Failure?*



When a node fails:

1) Teradata resets

2) On the restart the AMPs in Node 1 Migrate

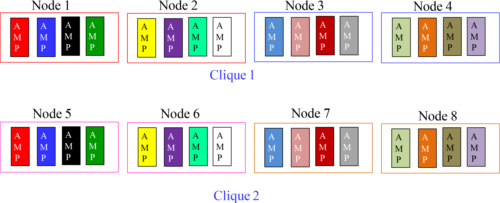
3) The system is degraded but still able to function

4) The down node is fixed

5) Another reset is done and the AMPs return home

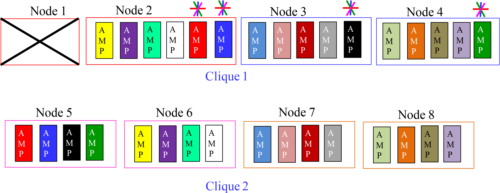
When Node 1 fails, Teradata does a reset. During the restart, the Teradata database knows about the failed node and instructs the AMPs from Node 1 to migrate to Node 2 (since they are in the same clique). The system performance is degraded, but still operational. When Node 1 is fixed, another Teradata reset is required. The AMPs go back to their normal home in Node 1 and the system is back to normal.

*Four Node Cliques*



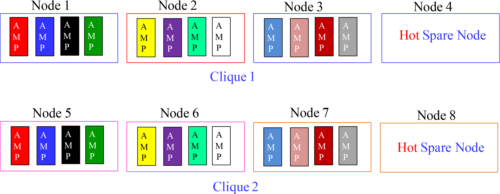
A clique protects against a node failure. The above picture is designed to show four nodes in a clique. The purpose of grouping nodes into cliques is in case a node should fail, the AMPs in the failed node can migrate to the other nodes within the clique. The reason for a larger clique is so there is less degradation when a node fails. The other three nodes in the clique will help equally when the AMPs migrate after the reset. Our next slide will demonstrate this concept.

*Migrating AMPs in Four Node Cliques*



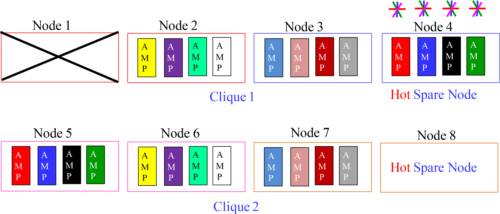
A clique protects against a node failure. The above picture is designed to show four nodes in a clique and then their migration to the other AMPs in the same clique upon a failure. The other three nodes in the clique will help equally when the AMPs migrate after the reset. Teradata resets, then the AMPs migrate, then the Node is fixed and another Teradata reset the AMPs from Node 1 return back to Node 1.

*The Hot Spare Node*



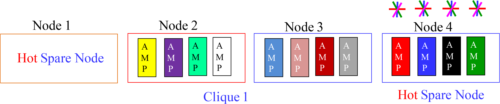
A Hot Spare is there in case of a node failure. The failing node causes Teradata to reset, but upon Teradata coming back up, the AMPs in the failed node migrate to only the Hot Spare Node within the clique. Now the system is at full strength again.

*The Hot Spare Node in Action*



The AMPs in Node 1 migrate directly to the Hot Spare Node, and the system is back to full system performance. When Node 1 is fixed, it will become the Hot Spare Node. So, the system only resets on the original node failure, but does not have to reset once the failed node is fixed. This configuration is called a 3 and 1.

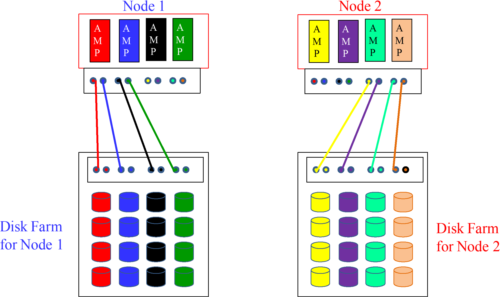
*With a Hot Spare a Second Teradata Reset isn’t Needed*



Without a Hot Spare Node, when a Teradata node fails Teradata resets. When the database resets and comes back up the AMPs in the failed node normally migrate to other nodes in the clique. When the failed Node is fixed, a second Teradata Reset is performed so the migrating AMPs can return to their newly fixed node.

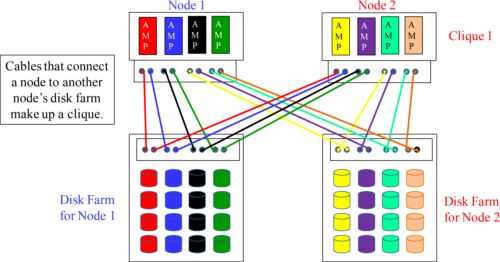
That is why a Hot Spare is often purchased. With a Hot Spare Node there is no second Teradata reset. What happens is that when a node fails Teradata resets. Then, when the database comes back up, the AMPs in the failed node migrate to the Hot Spare Node. When the failed node is fixed, it becomes the Hot Spare Node (as in the picture above). So, no second Teradata reset needs to be performed.

*A Node, It’s AMPs and their Disks*



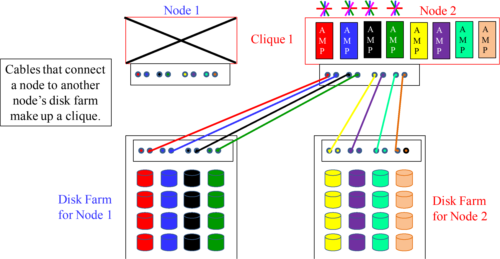
Inside the memory of each node live the AMPs. They are called AMP VProcs because they are virtual processors in memory that perform AMP duties. The node is then connected to a Disk Array cabinet running RAID 1 Mirroring. Each AMP is connected to its one Virtual Disk (Four Physical Disks), and Teradata is ready to run.

*How Cliques are Physically Defined*



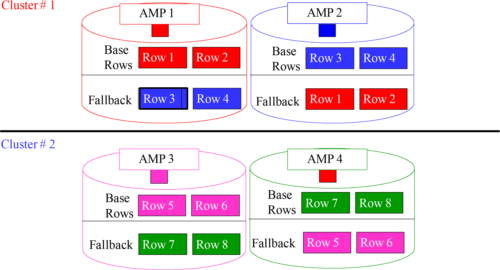
Cliques are defined physically. This is done by connecting a node to another nodes disk farm. It is physically accomplished with cables connecting each node to the others disk farm. Now, if a node fails and the AMPs migrate to the other node, they still have direct access to their disks. The next page shows this even more clearly!

*Cliques are cabled so Migrating AMPs can access their Disks*



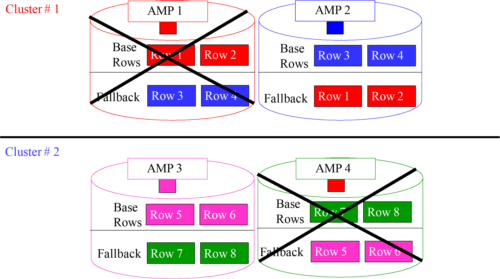
Cliques are defined physically. This is done by connecting a node to another nodes disk farm. It is physically accomplished with cables connecting each node to the others disk farm. Now, if a node fails and the AMPs migrate to the other node, they still have direct access to their disks. Notice above that all AMPs still have access to their same disks.

*A Review of Fallback and Clusters*



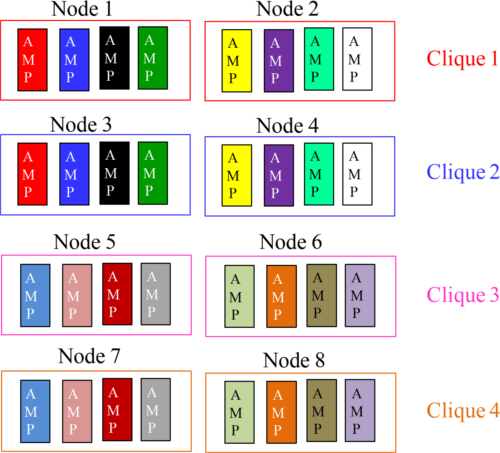
A Table with FALLBACK makes a copy of each row on the table and stores that copied row on another AMP. These two buddy AMPs are considered in the same cluster. How many AMPs can we lose, potentially, in the above example? Check out the next page!

*An Example of Fallback and Clusters*



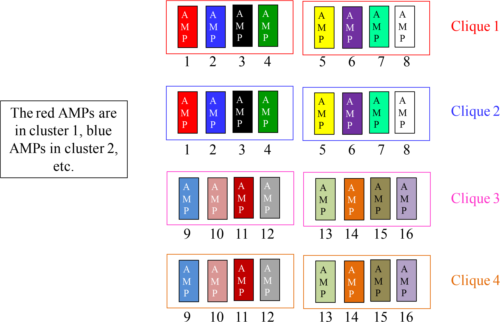
A Fallback protected table allows for the loss of one AMP per Cluster! Notice that we can still read Rows 1, 2, 3, 4, 5, 6, 7 and 8.

*Quiz 1 – How Many Clusters do you see?*



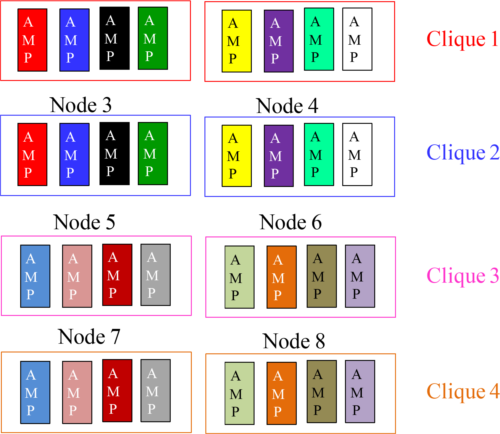
You probably understand Fallback, Clusters and Cliques, but the difficult part is seeing how they are designed to work together. Let me take you there with a series of quizzes. How many Fallback Clusters do you see in the above picture? I actually did you a favor and color coded the AMPs to clearly allow you to see the answer and build upon it.

*Quiz 1 Answer – How Many Clusters do you see?*



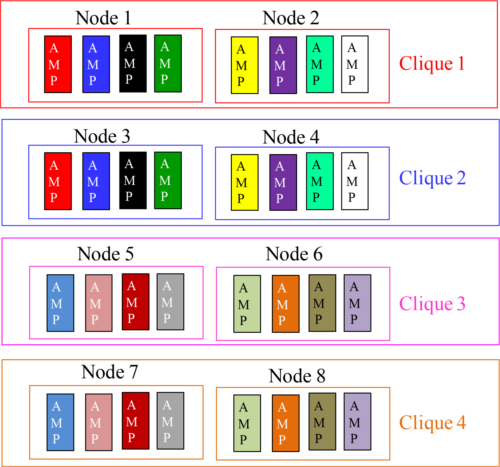
There are 16 Fallback clusters each containing two AMPs in the above picture. The red AMPs are in cluster 1. The blue AMPs are in cluster 2, and so on. I have placed the cluster numbers in there for your convenience.

*Quiz 2 – How Many Cliques do you see?*



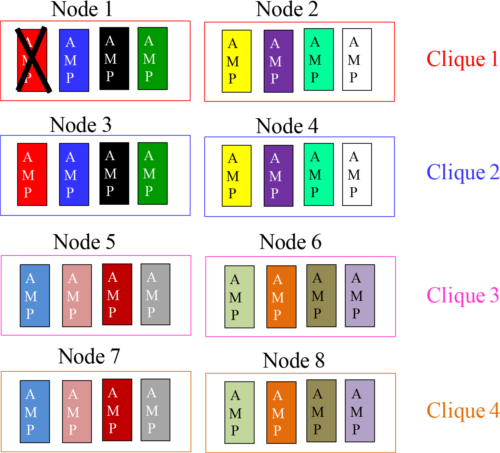
You probably understand Fallback, Clusters and Cliques, but the difficult part is seeing how they are designed to work together. Let me take you there with a series of quizzes. How many Cliques do you see in the above picture?

*Quiz 2 Answer – How Many Cliques do you see?*



How many Cliques do you see in the above picture? There are four Cliques. Each clique is made up of two nodes in this example.

*Quiz 3 – What have we lost? Multiple Choice Answer*



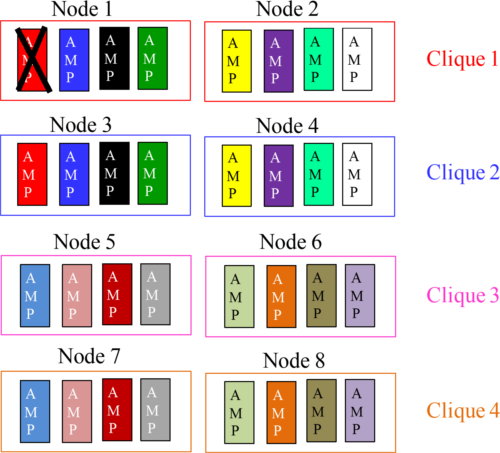
A) We have lost Node 1

B) We have lost query capabilities on Full Table Scans for Fallback protected tables

C) We have lost one AMP in cluster 1 (red cluster)

D) We have lost all AMPs in Clique 1

*Quiz 3 Answer – What have we lost? Multiple Choice Answer*



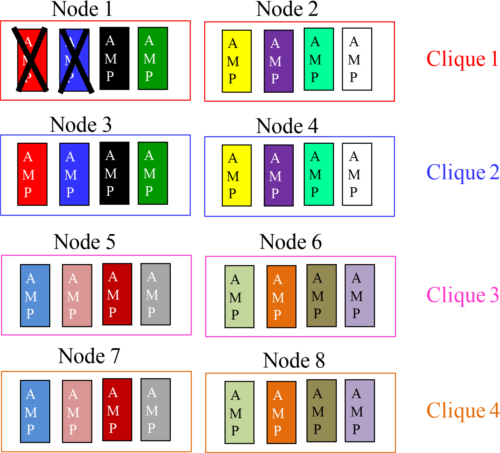
A) We have lost Node 1

B) We have lost query capabilities on Full Table Scans for Fallback protected tables

C) We have lost one AMP in cluster 1 (red cluster)

D) We have lost all AMPs in Clique 1

*Quiz 4 – What have we lost? Multiple Choice Answer*



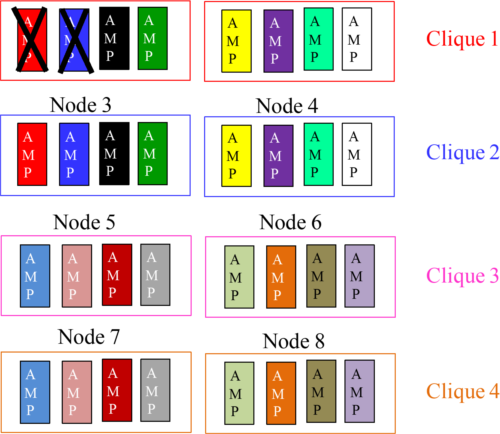
A) We have lost Node 1

B) We have lost query capabilities on Full Table Scans for Fallback protected tables

C) We have lost one AMP in cluster 1 (red cluster) and one AMP in cluster 2 (Blue)

D) We have lost all AMPs in Clique 1

*Quiz 4 Answer – What have we lost? Multiple Choice Answer*



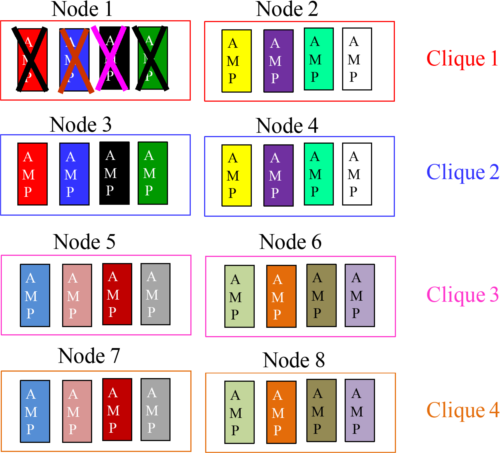
A) We have lost Node 1

B) We have lost query capabilities on Full Table Scans for Fallback protected tables

C) We have lost one AMP in cluster 1 (red cluster) and one AMP in cluster 2 (Blue)

D) We have lost all AMPs in Clique 1

*Quiz 5 – What have we lost? Which Answer is False?*



A) We have lost all AMPs in Node 1

B) We have lost query capabilities on Full Table Scans for Fallback protected tables

C) We have lost only one AMP in each of the following clusters (1, 2, 3, and 4)

D) Fallback protected tables can still be queried with a Full Table Scan

*Quiz 5 Answer – What have we lost? Which Answer is False?*



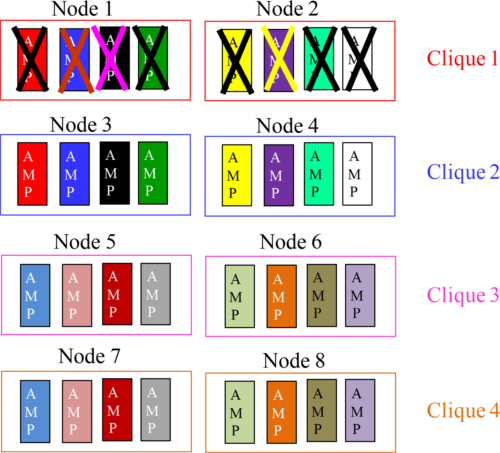
A) We have lost all AMPs in Node 1

B) We have lost query capabilities on Full Table Scans for Fallback protected tables

C) We have lost only one AMP in each of the following clusters (1, 2, 3, and 4)

D) Fallback protected tables can still be queried with a Full Table Scan

*Quiz 6 – What have we lost? Pick Two True Answers*



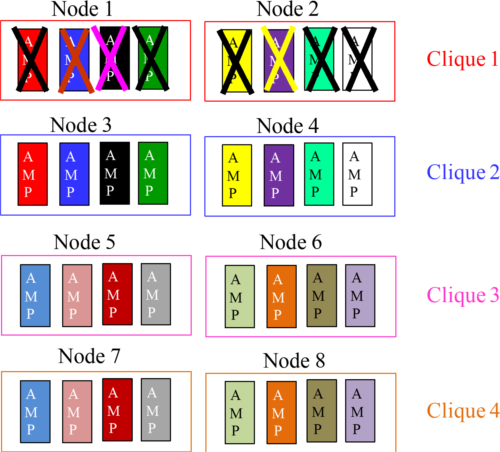
A) We have lost Clique 1

B) We have lost query capabilities on Full Table Scans for Fallback protected tables

C) We have lost only one AMP in each of the following clusters (1, 2, 3, 4, 5, 6, 7, 8)

D) The Teradata system can’t handle any queries until Clique 1 is fixed

*Quiz 6 Answer – What have we lost? Pick Two True Answers*



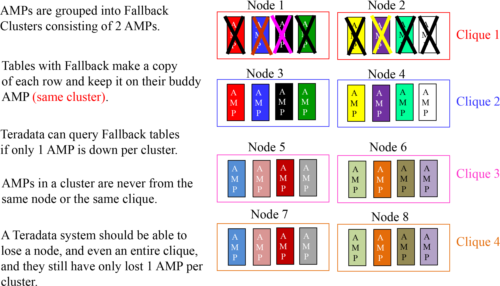
A) We have lost Clique 1

B) We have lost query capabilities on Full Table Scans for Fallback protected tables

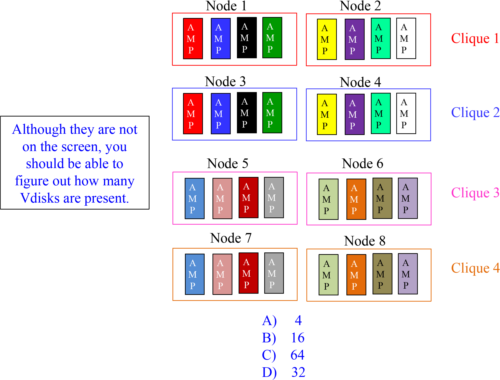
C) We have lost only one AMP in each of the following clusters (1, 2, 3, 4, 5, 6, 7, 8)

D) The Teradata system can’t handle any queries until Clique 1 is fixed

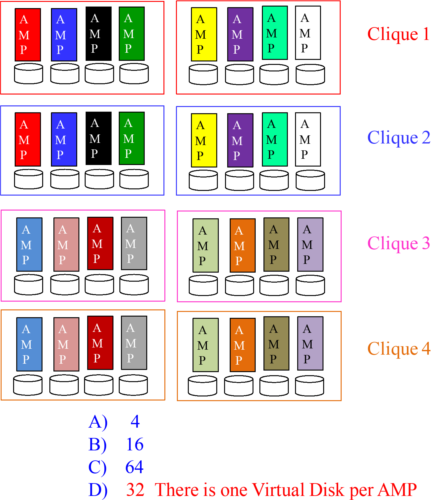
*Summary of the facts for Fallback, Clusters, and Cliques*



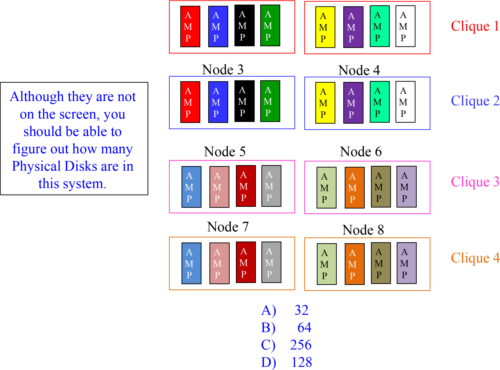
*Quiz 7 How Many Virtual Disks (Vdisks) are in this System?*



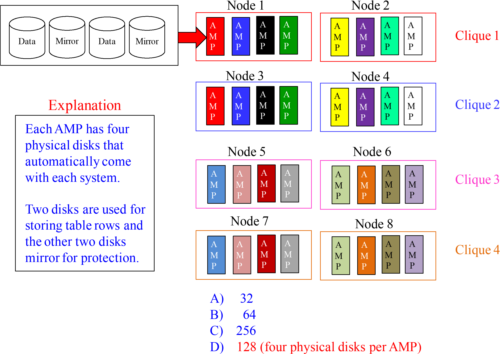
*Quiz 7 Answer –How Many Virtual Disks are in this System?*



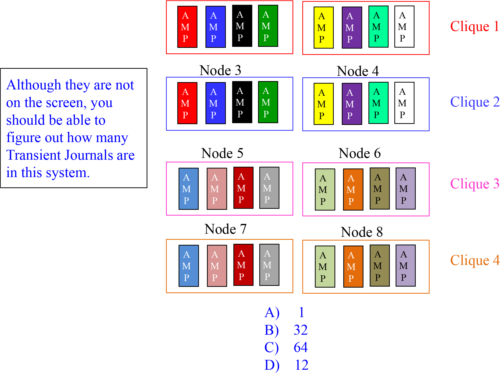
*Quiz 8 –How Many Physical Disks are in this System?*



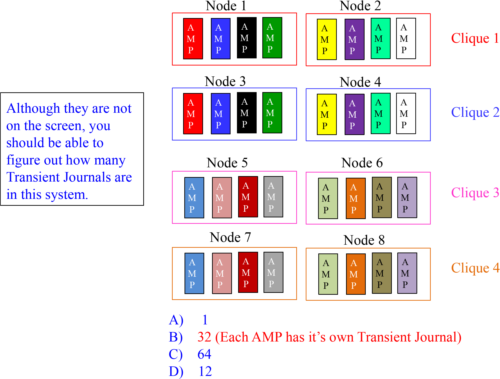
*Quiz 8 Answer–How Many Physical Disks are in this System?*



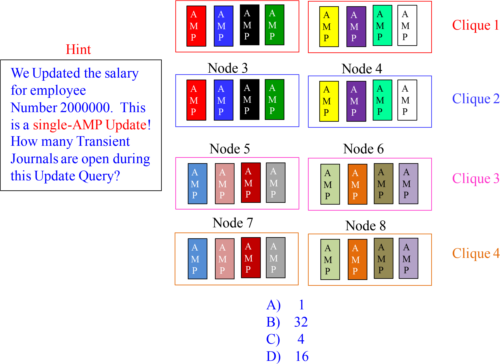
*Quiz 9 – How Many Transient Journals in this System?*



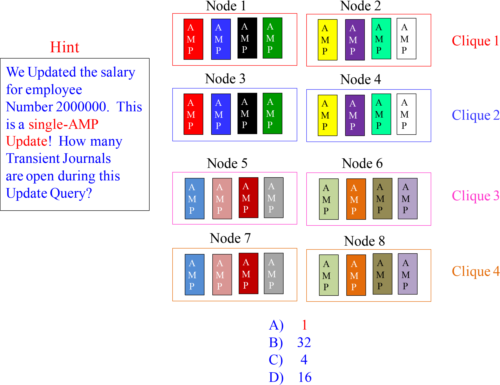
*Quiz 9 Answer –How Many Transient Journals in this System?*



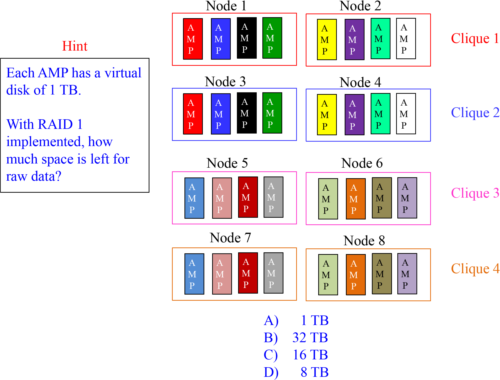
*Quiz 10 – How Many Transient Journals are Open?*



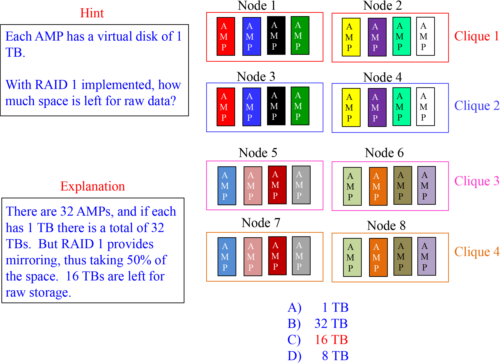
*Quiz 10 Answer – How Many Transient Journals are Open?*



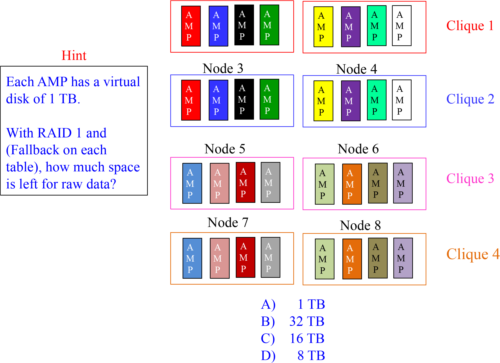
*Quiz 11 – How Much Space?*



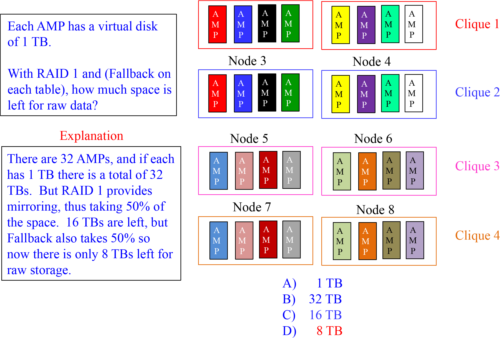
*Quiz 11 Answers – How Much Space?*



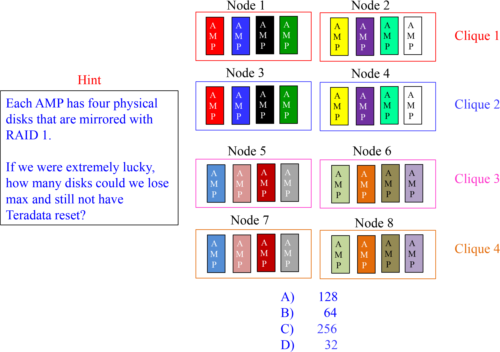
*Quiz 12 – How Much Space with Fallback?*



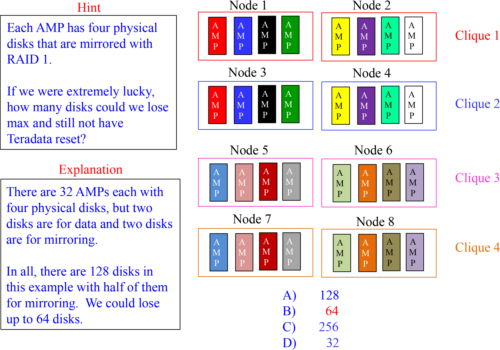
*Quiz 12 Answers – How Much Space with Fallback?*



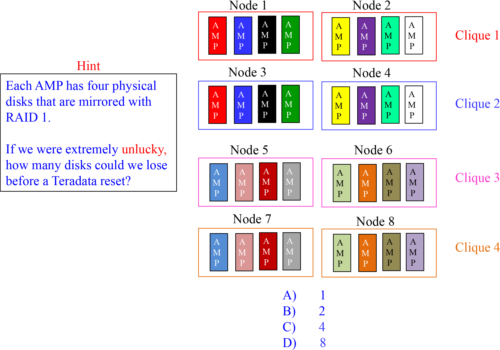
*Quiz 13 – How Many Disks could we lose with RAID 1?*



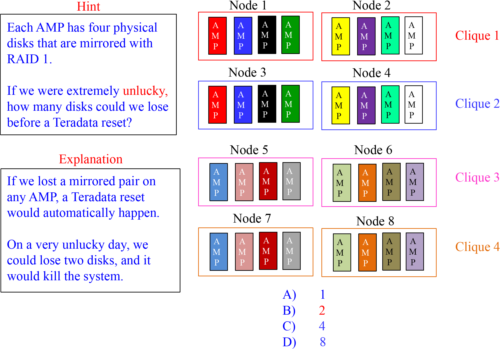
*Quiz 13 Answer – How Many Disks could we lose?*



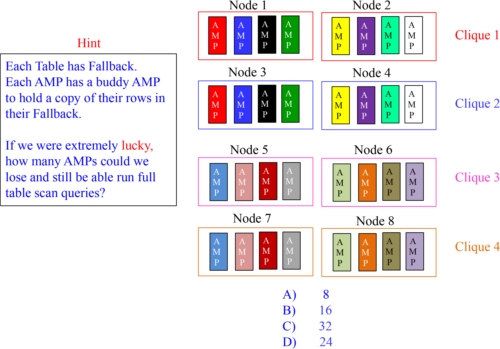
*Quiz 14 – How Many Disk losses could Kill Us?*



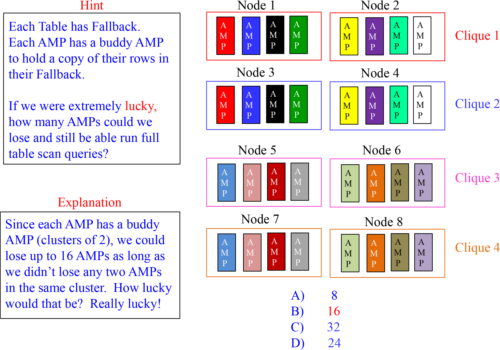
*Quiz 14 Answer – How Many Disk losses could Kill Us?*



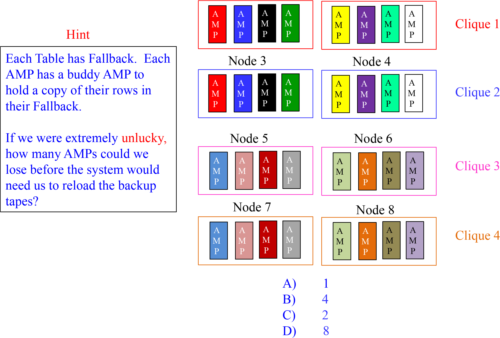
*Quiz 15 – How Many AMPs could we lose if Lucky?*



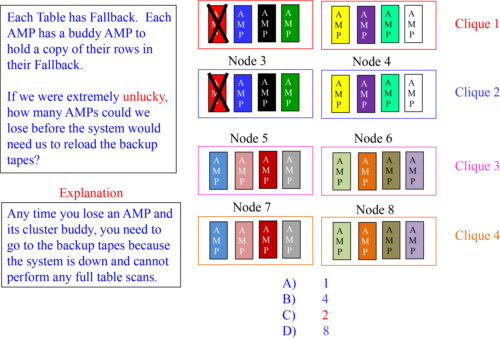
*Quiz 15 Answer – How Many AMPs could we lose if Lucky?*



*Quiz 16 – How Many AMPs could we lose if Unlucky?*

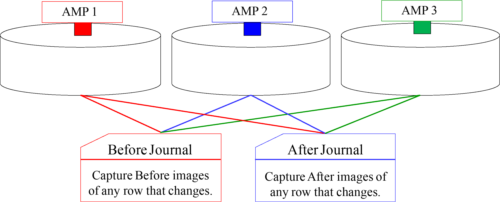


*Quiz 16 Answer – How Many AMPs could we lose Unlucky?*



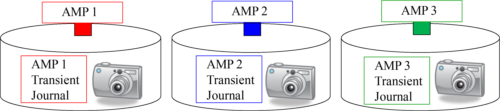
*The Permanent Journal*

A Permanent Journal captures row images of all rows changing because of an Insert, Update, or Delete and keeps them permanently.



Sometimes disaster strikes the hardware, the software, or a programmer makes a disastrous mistake. Thank goodness for the Permanent Journal. When disaster strikes, the Permanent Journal can be used to Rollback or Roll forward to a point in time when everything was great. The Permanent Journal only records images if a row changes.

*Difference between the Transient and the Permanent Journal*

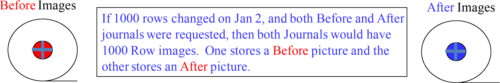


Transient Journal – Each AMP has a distinct Transient Journal for database integrity. If an AMP has a row that is about to change (Insert, Update, Delete), it will take a before picture and store it in its Transient Journal. If the transaction fails, then the AMP can Rollback the before image. If, however, the transaction is successful, the Transient Journal throws away the image! The Transient Journal lives for the life of a transaction.

Permanent Journal – All AMPs copy any changes to the database and store them in the Permanent Journal permanently. The DBA or Table Creator must request a Permanent Journal. Each day, the DBA usually does a “Check Point with Save”, and that days Permanent Journal is stored permanently on tape or disk off of the Teradata system.

Here is something they both have in common: They never store anything on a SELECT query because nothing in the database is going to change!

*Difference Between the Before and After Permanent Journal*



After Journal – This is by far the most popular Permanent Journal. It is always used in conjunction with Disaster Recovery and serves as a Backup of row changes to Teradata. The After Journal is called the After Journal because it takes a picture after a change to a row has been made. By capturing the After image of all row changes daily, we can restore a database to a point-in-time (using our Full System Backup) and then apply the additional daily changes from the After Journal for a full recovery.

Before Journal – This journal is not used that much. The Before Journal takes a Before picture before a change to a row has been made. Imagine that a programmer was told to give everyone a 1% raise. Instead, the programmer made a query mistake and gave everyone a 10% raise. The Before Journal could be used to go back in time on the Employee\_Table and change the rows back to the way they were before the mistake. Then, the programmer could re-run their update query and give everyone a 1% raise.

*Full System Backup compared to an After Journal*

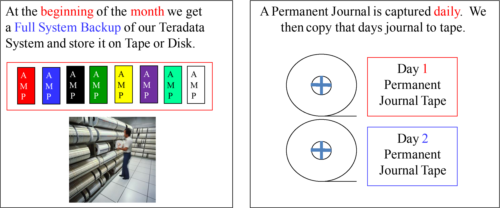


Full System Backup – This backup is designed to capture every database, table, and row and store it on external media in case of a disaster. If there are a trillion rows in total on your Teradata system, the Full System Backup would make a copy of all trillion rows. This helps the person responsible for the data sleep at night knowing all is safe.

After Journal – Think of this journal as a daily journal that only captures After images of the rows that change. If there were a trillion rows in total on your Teradata system, the After Journal would only take After pictures of the rows that changed that day. It would only do so if the database or table had requested an After Journal.

*How Full System Backups work with the After Journal*

A Permanent Journal tracks all Inserts, Updates, and Deletes and keeps them permanently. A Permanent Journal is used in conjunction with Monthly Backups.



Let’s assume we did a Full System Backup of our Teradata system on January 1 and stored the backup in our tape library. We then capture all daily changes and store them in our Permanent Journal. Daily, we store the Permanent Journal to tape. If a disaster occurs on our Teradata system, we can restore to January 1 with the Full System Backup tapes and then add the Daily Permanent After Journal tapes for a Full System Recovery.

*The Many Different Permanent Journal Options*

After Journal – Takes an After picture of any Insert, Update or Deleted row.

Before Journal – Takes a Before picture of any new or changing row.

Journal – Takes a Before and an After picture of any Insert, Update or Deleted row.

Dual Before Journal – Takes two Before pictures of any new or changing row and stores them on a buddy AMP for extra security.

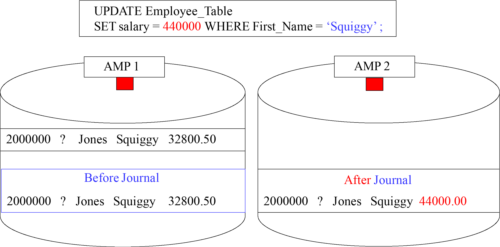
Dual After Journal – Takes two After pictures of any Insert, Update or Deleted row and stores them on a buddy AMP for extra security.

The most popular Permanent Journal is the After Journal because this is used in conjunction with a company’s Full System Backups. A Before Journal is used to rollback mistakes the programmers made when maintaining data. This journal is not used very often. A Dual After journal is also used in conjunction with the Backup plans and having two After images of each change provides even better system protection.

*Where is the Permanent Journal Stored?*

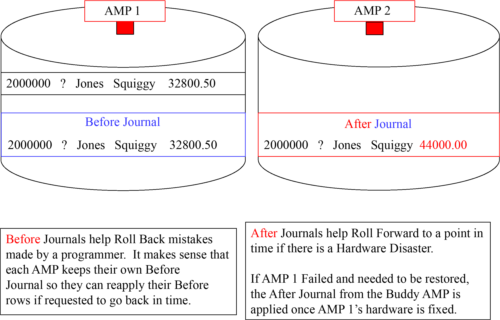
Before Journal rows are stored on the AMP where the row has changed.

After Journal rows are stored on the Buddy AMP.

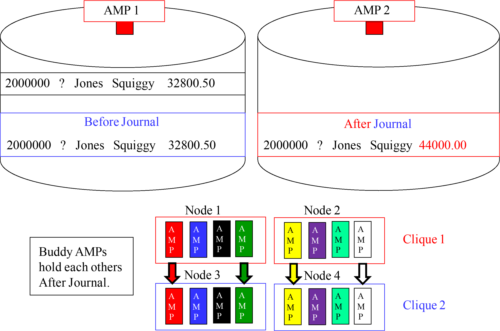


A Before Journal row image is stored on the AMP where the change was made. An After Journal row image is stored on the Buddy AMP within the cluster. These are the Fallback pairs, also called the “Buddy AMPs”. They back each other up.

*Using Common Sense about Journal Locations*

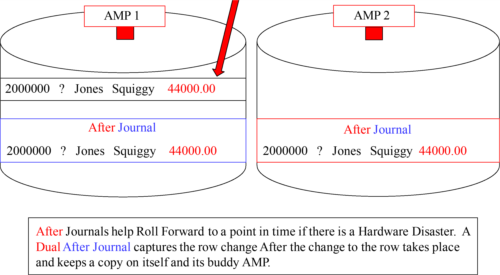


*After Journals are Never stored in the Same Node or Clique*



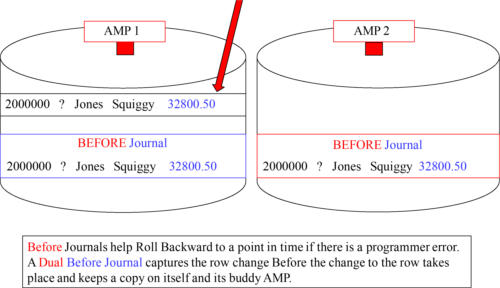
*What is a Dual After Journal?*

Squiggy Jones just received a raise and his row (on AMP 1) was updated from his old salary of 32800.50 to 44000.00.



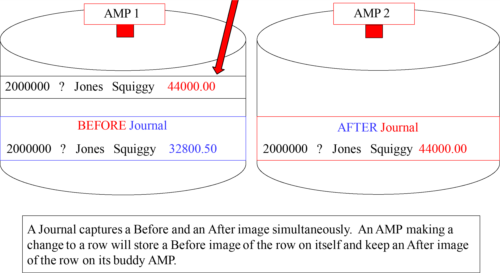
*What is a Dual Before Journal?*

Squiggy Jones just received a raise and his row (on AMP 1) is about to be updated from his old salary of 32800.50 to 44000.00.

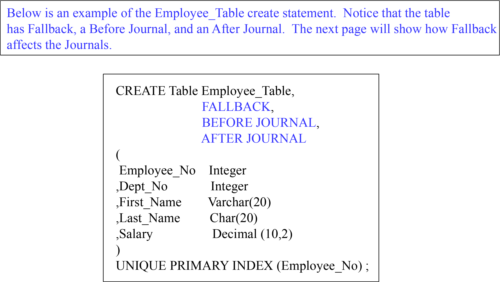


*What is a Journal?*

Squiggy Jones just received a raise and his row (on AMP 1) and we updated his old salary from 32800.50 to 44000.00.

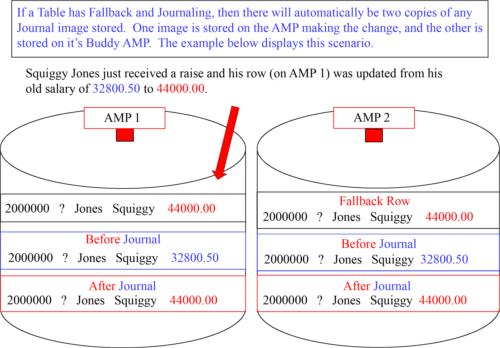


*Creating a Table with Fallback and a Before and After Journal*



The next page will show this table in action. We will Update a row on a table with Fallback and a Before and After Journal.

*Does Fallback Affect a Permanent Journal?*



*Permanent Journal Rules*

Permanent Journals are created with a CREATE or MODIFY DATABASE or USER statement. That is why there can be only one Journal per USER or DATABASE.

Tables within a DATABASE or USER will be assigned to the PERMANENT JOURNAL within their USER or DATABASE by default (only if one exists) but can also be specifically defined in another DATABASE or USER.

If a JOURNAL Table in another DATABASE or USER is specified when a table is created, then the other JOURNAL Table must already exist.

Tables that have FALLBACK and also specify Journaling will automatically receive DUAL Image Journaling. Tables without FALLBACK can request Dual Image or Single Image Journals.

Permanent Journals take up PERM Space. If a USER or DATABASE holding the Journal runs out of PERM Space, then all transactions will be aborted.

A Permanent Journal will no longer work or be applicable if the system is Reconfigured with more AMPs, or if a MODIFY USER or DATABASE command changes the object.

*Example 1: Permanent Journal Scenarios to Test the Rules*

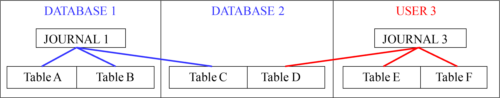
This first example follows all the rules. Each Database or User has their own Journal.



Permanent Journaling in this example is pure Vanilla. This example shows each Database or User with their own Permanent Journal. The tables by default go to the Journal in their Database or User.

*Example 2: Permanent Journal Scenarios to Test the Rules*

Some of the table here use the Journal in their own Database or User, but not all!



Permanent Journaling in this example is a little more interesting. This example shows Database 1 and User 3 with their distinct Permanent Journal. The tables in these databases or Users by default go to the Journal in their home Database or User. What is interesting here is that in Database 2 Table C uses Journal 1 in Database 1 and Table D uses Journal 3 in User 3. Teradata believes a Database and User is the same thing. The only minor difference is that a User can login and run queries, but a database can’t. Both can have Perm and Spool space assignments and a Permanent Journal.

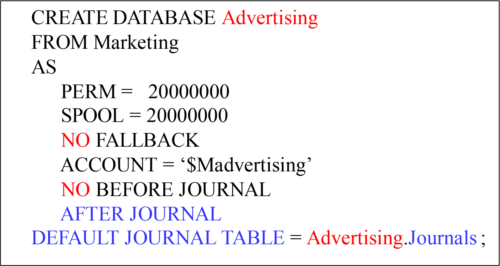
*Example 3: Permanent Journal Scenarios to Test the Rules*

In this example, there is only one Journal, and each Database and User that contain tables are using the Journal 1 in Database 2.



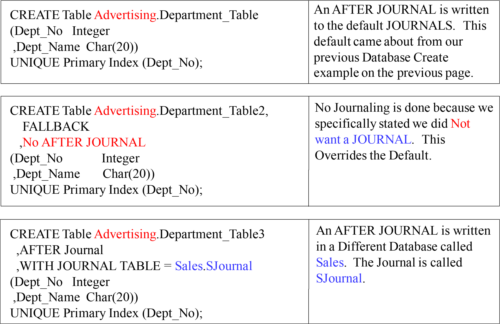
I like this example because there is only one Journal in the system. Every table uses this Journal 1 for their Permanent Journal. No rules have been broken here!

*How to Create Database with a Permanent Journal*

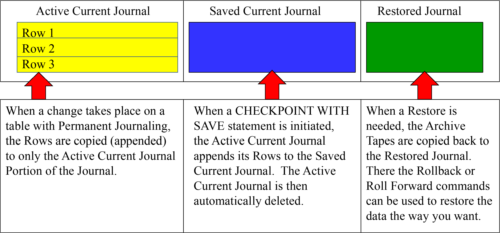


All tables created in the database Advertising will by default keep a Single After Journal in the Permanent Journal named Journals!

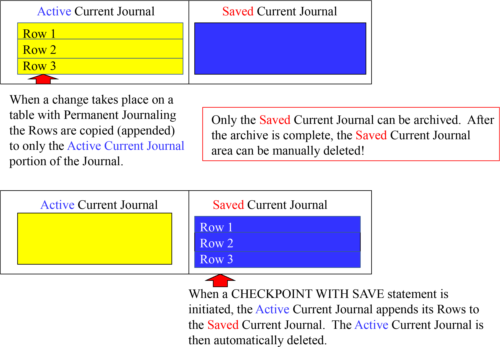
*Creating Tables under different Journal Circumstances*



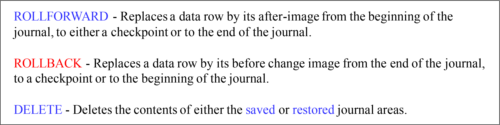
*Permanent Journal’s Three Main Areas*



*The Current Journal consists of the Active and Saved Areas*

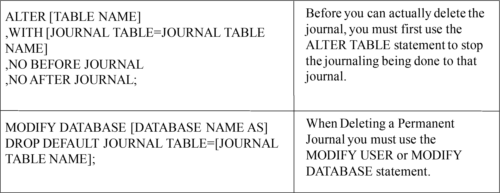


*Permanent Journal Commands*



There are two utilities that will utilize the Permanent Journal to backup and restore data. They are the ARC (Archive and Recovery) Console utility on a channel-attached host (mainframe) and ASF2 (Archive Storage Facility 2) utility on a network-attached system. Additionally, ASF2 can backup your channel-attached system to SMP-attached tape drives. So, take a look at the commands above and use them in ARC or ASF2.

*Deleting a Permanent Journal*



When Deleting a Permanent Journal, you must use the MODIFY USER or MODIFY DATABASE statement. Here is the catch! Before you can actually delete the journal, you must first use the ALTER TABLE statement to stop the journaling being done to that journal.

*Some Great Advice for Maintaining the Permanent Journals*

Checkpoint all JOURNALS with SAVE

Archive the JOURNAL Tables

DELETE the SAVED JOURNALS

ARCHIVE each day

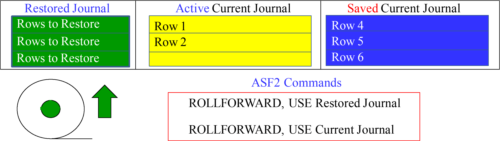
All-AMP Archives Weekly or Monthly

*Recovery Using the Permanent Journals*

1. Submit an SQL Checkpoint statement as the first statement of a batch job, with or without a Checkpoint name. A Checkpoint Name makes things easier later if you need to recover, but it is not mandatory.

2. If required, ROLLBACK to the Checkpoint using either the checkpoint name or the event number supplied by the DBC when you executed the Checkpoint command. Later changes are also handled with a back out.

3. The data table is now in its original condition.



Remember that a permanent journal is time-oriented, not transaction-oriented. Journal Tables or databases are restored first. Next, archived journals are restored, one at a time, and then the restored journal is rolled forward.

*The Journals View in DBC (DBC.Journals)*

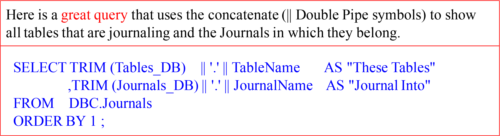
The DBC.Journals view has four different columns.

1) Tables\_DB - Displays the name of a database where a data table resides that has the journal option activated.

2) TableName - Displays the name of a data table that records changed images in a journal table.

3) Journals\_DB - Displays the name of a database where a journal table resides.

4) JournalName - Displays the name of a journal table associated with a listed data table.



Teradata provides a system view called DBC.Journals that displays associations between journal tables and the data tables that journal to them.

*Archive Recovery Console (ARC)*

The ARC utility is utilized for four reasons:

|  |  |
| --- | --- |
| Archive | Backs up information from the Teradata system onto some type of portable storage media. |
| Restore | The restore function reverses the archive process and moves the data from the storage media back to the database. |
| Copy | Allows you to copy data from one system onto another. |
| Recovery | Utilizes information stored in permanent journals to rollback or roll forward row information |

Archive and Recovery (ARC) Console backs up and restore databases and their associated objects. Remember, to place archived data onto a **different system** , you would use the **Copy command**.

*Reasons You Might Utilize ARC*

Archive a database, individual table, or selected partitions of a PPI table directly from Teradata to external media such as tape or disk.

Restore a database, individual table, or selected partitions of a PPI table back to Teradata from external media.

Copy an archived database, table, or selected partitions of a PPI table to Teradata on a different Teradata system than the one from which the system was archived.

Place checkpoint entries in a Teradata journal table.

Recover a database to checkpoint in time by using a Rollback or Roll forward, using the Before or After images from a Permanent journal table.

Delete rows from a journal table.

The ARC utility has come a long way, and above are some creative ideas to using ARC.

*ARC raising the BAR (Backup Archive Restore)*

The ARC facility is required to archive/restore/copy the Teradata Database and it can be run in batch or interactively. But, Teradata uses 3rd party products in what they term BAR (Backup Archive Restore).

The 3rd Party BAR products include:

1. NetBackup

2. Tivoli Storage Manager

3. NetVault

4. Command Line ARCMain Execution

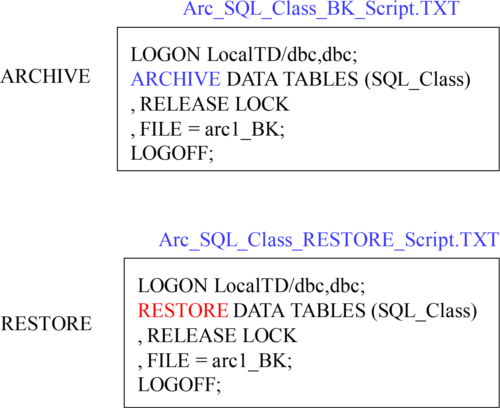
NetBackup and Tivoli use a design called TARA, which stands for TARA (Tiered Archive Restore Architecture)

The ARC utility is required to archive to Teradata. So ARCMAIN is the executable, but Teradata also relies on 3rd party products. They are listed above.

*ARC Commands in Alphabetical Order*

|  |  |
| --- | --- |
| ANALYZE | Reads an archive tape for information about it’s contents. |
| ARCHIVE | Archives a copy of a database or table. |
| BUILD | Builds Fallback on a table plus it builds all indexes for a Table. |
| CHECKPOINT | Places a Checkpoint in a Permanent Journal. |
| COPY | Restores a copy of an archive to ant Teradata system of choice. |
| DELETE DATABASE | Deletes a Teradata Database. |
| DELETE JOURNAL | Deletes SAVED or RESTORED Journal rows. |
| LOGON/LOGOFF | Provides a Logon or Logoff. |
| RESTORE | Restores a database/table from an archive file to specific AMPs. |
| REVALIDATE REFERENCES | Revalidate referential integrity. Acts as a cleanup function. |
| ROLLBACK | Recovers a database and tables to a BEFORE previous state. |
| ROLLFORWARD | Recovers a database or table to an AFTER state. |

*An ARC Example of an Archive and then a Restore*



In our example, we first ARCHIVE the SQL\_Class Database and store the archive on disk. Then, we deleted all the tables in the SQL\_Class Database. Then, using our second example we restored the SQL\_Class tables.