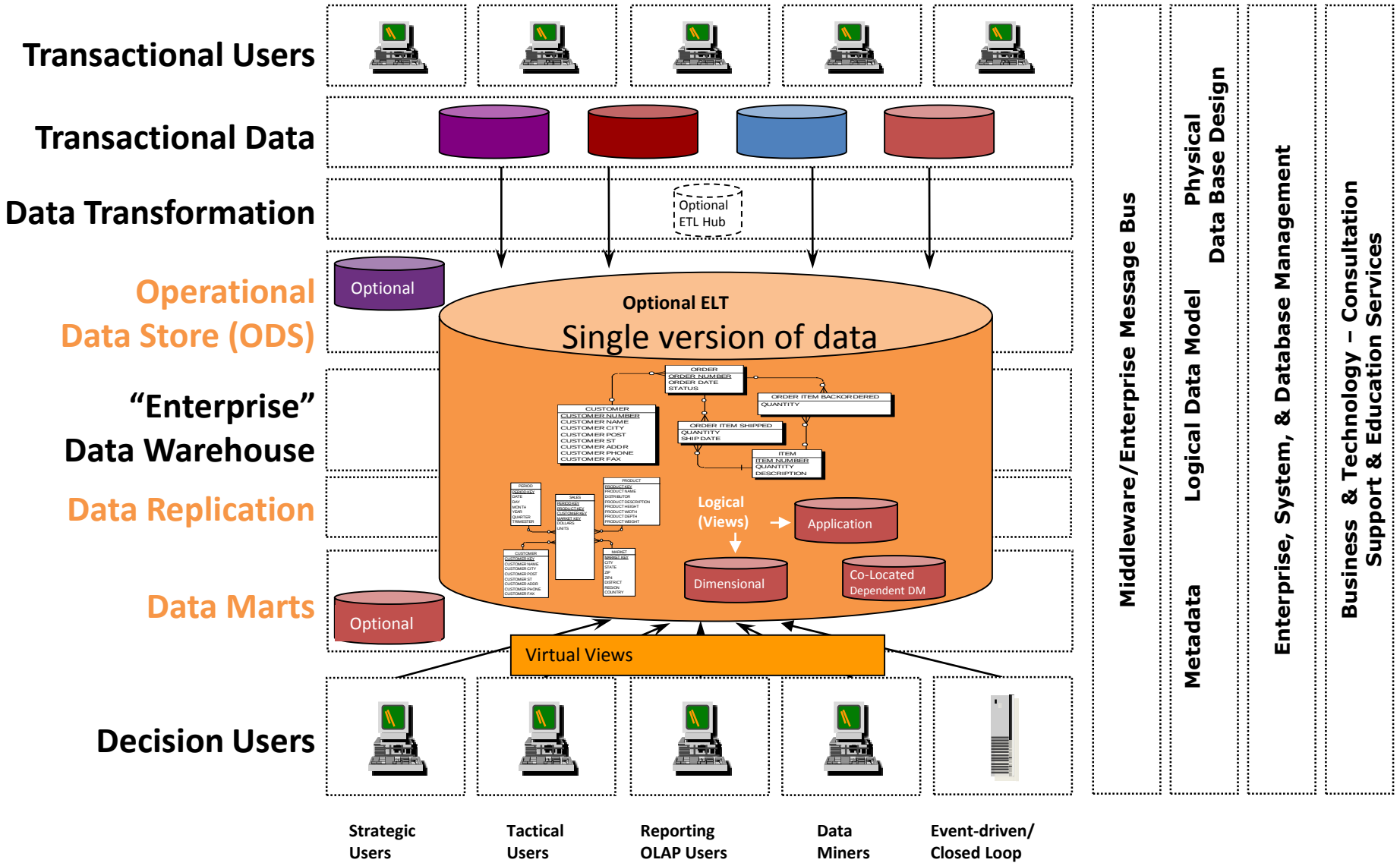
Teradata Training

Agenda

- What is Teradata
- Teradata EDW
- Teradata Database Manageability
- Teradata Database Competitive Advantage
- Teradata Architecture
- Teradata Storage Architecture
- Teradata Retrieval Architecture
- Teradata High Availability
- Teradata Objects
- Permanent, Spool and Temporary space
- Create Table Example
- Data Types
- Views
- Select
- Join Example
- Macros
- Help Command
- Show Command
- Explain Facility
- Locks
- Indexes

- Teradata is RDBMS founded in 1979 designed to run largest databases.
- It is massively parallel processing system.
- It works on UNIX-MP-RAS, Windows 2003, Linux
- Teradata DBMS inearly and predictably scalable in all dimensions of database system workload.
- Teradata is offered on Intel servers interconnected by the BYNET messaging fabric
- Teradata systems offered with either Engenio or EMC disk arrays for database storage.
- It has a parallel aware optimizer that allows multiple complex queries to run concurrently.
- It uses a Shared Nothing architecture.

Teradata EDW



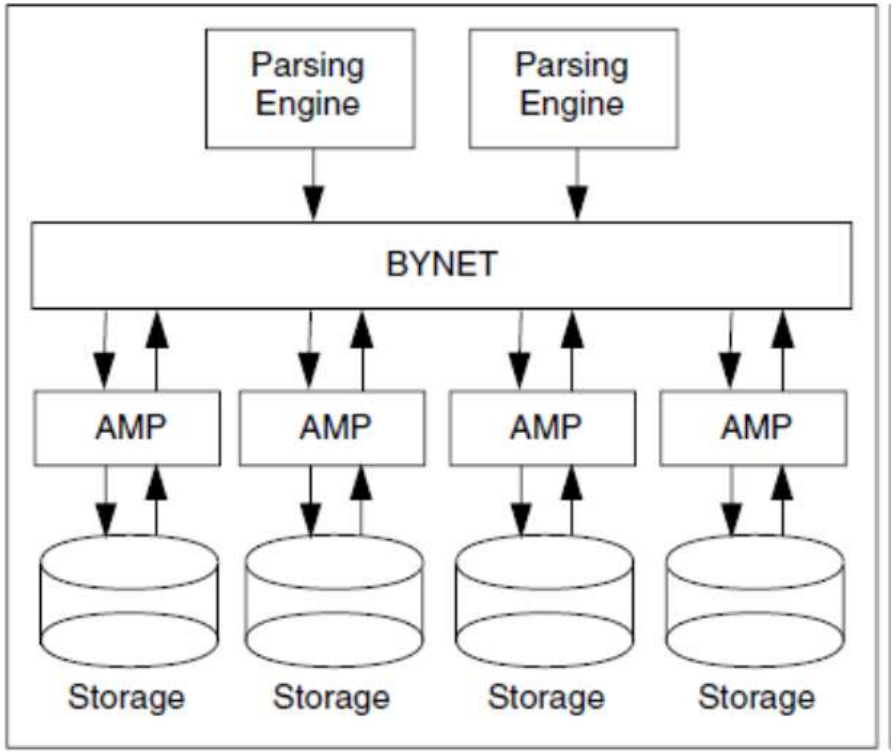
Things a Teradata Database DBA never has to do!

- ✓ Reorganize data or index space
- ✓ Pre-allocate table/index space, format partitions
- ✓ Pre-prepare data for loading (convert, sort, split, etc.)
- ✓ Ensure that queries run in parallel
- ✓ Unload/reload data spaces due to expansion
- ✓ Design, implement and support partition schemes
- ✓ Write or run programs to split the input source files into partitions for loading.
- ✓ With Teradata the workload for creating table of 100 rows is the same as creating a table with 1,000,000,000 rows.
- ✓ Teradata DBA's know if the data doubles, the system can expand easily to accommodate it
- ✓ Teradata provide huge cost advantages, especially when it comes to staffing DB admins.

Teradata Database Competitive Advantages

- Unlimited, Proven Scalability – amount of data and number of users; allows for an enterprise wide model of the data.
- Unlimited Parallelism – parallel access, sorts, and aggregations.
- Mature Optimizer – handles complex queries, up to 64 joins per query, ad-hoc processing.
- Models the Business – normalized data (usually in 3NF), robust view processing, & provides star schema capabilities.
- Provides a “single version of the business”.
- Low TCO (Total Cost of Ownership) – ease of setup, maintenance, & administration; no re-orgs, lowest disk to data ratio, and robust expansion utility (reconfig).
- High Availability – no single point of failure.
- Parallel Load and Unload utilities – robust, parallel, and scalable load and unload utilities such as FastLoad, MultiLoad, TPT, TPump, and FastExport.

- Teradata acts as a single data store, with multiple client applications making inquiries against it concurrently.
- It provides the same connectivity for an entry-level system as it does for an EDW.
- A Teradata system contains one or more nodes
- A node is a term for a processing unit under the control of a single operating system.
- The node is where the processing occurs for the Teradata Database.

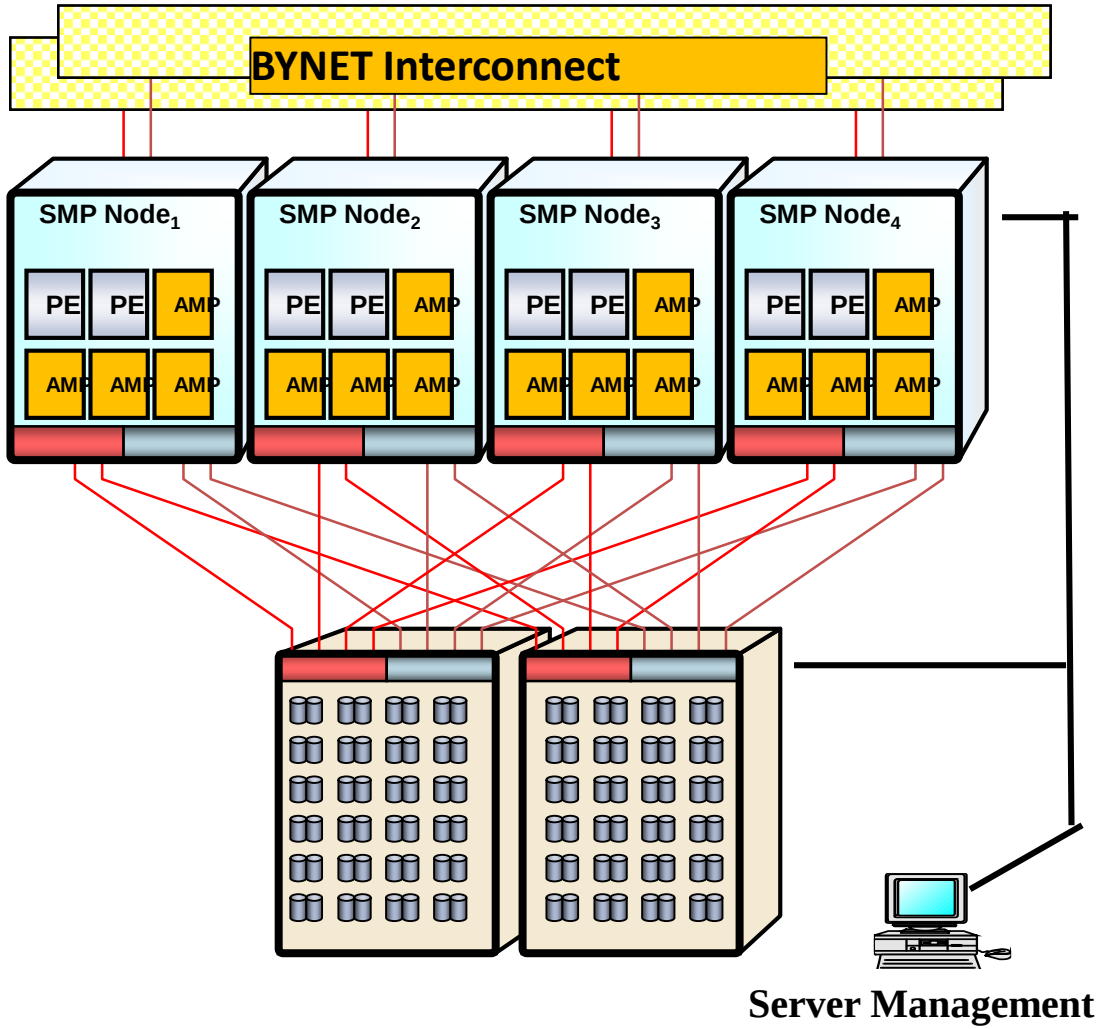


- **Symmetric multiprocessing (SMP)**: An SMP Teradata system has a single node that contains multiple CPU's sharing a memory pool
- **Massively parallel processing (MPP)**: Multiple SMP nodes working together comprise a larger, MPP implementation. The nodes are connected using BYNET, which allows multiple virtual processors on multiple nodes to communicate with each other.

The two major components of the Teradata Database are implemented as (virtual processors) vprocs

- Parsing Engine (PE)
- Access Module Processor (AMP)

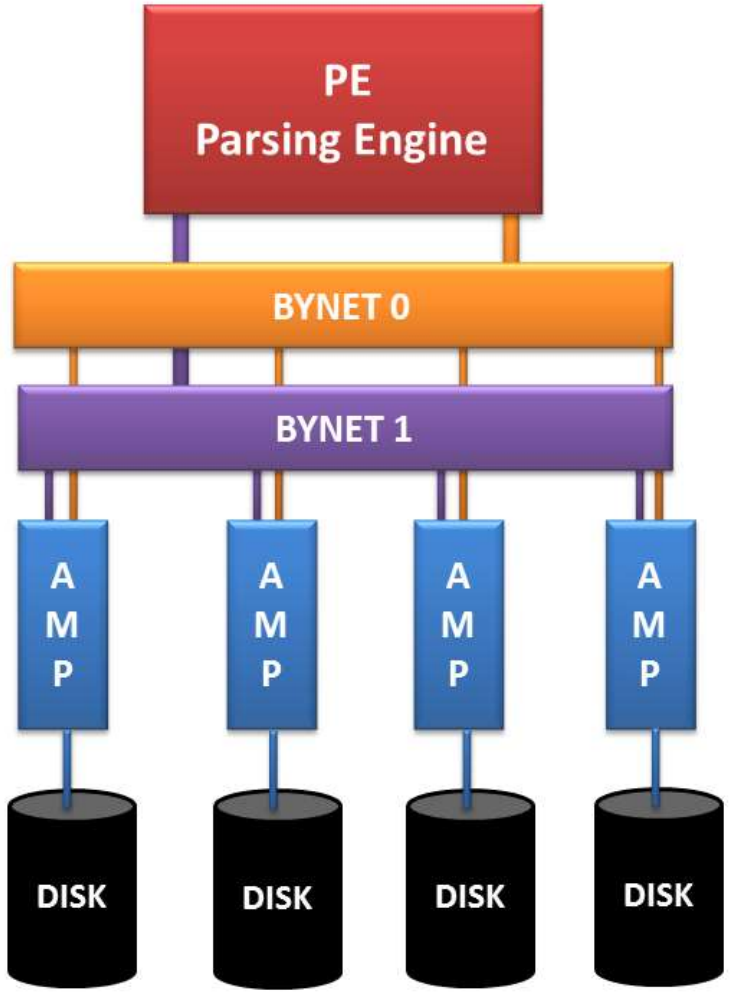
- SMP Nodes
  - Latest Intel SMP CPUs
  - Configured in 2 to 8 node cliques
  - Windows, Unix or Linux
- BYNET Interconnect
  - Fully scalable bandwidth
  - 1 to 1024 nodes
- Connectivity
  - Fully scalable
  - Channel - ESCON
  - LAN, WAN
- Storage
  - Independent I/O
  - Scales per node
- Server Management
  - One console to view the entire system



The Parsing Engine

The Parsing Engine is responsible for:

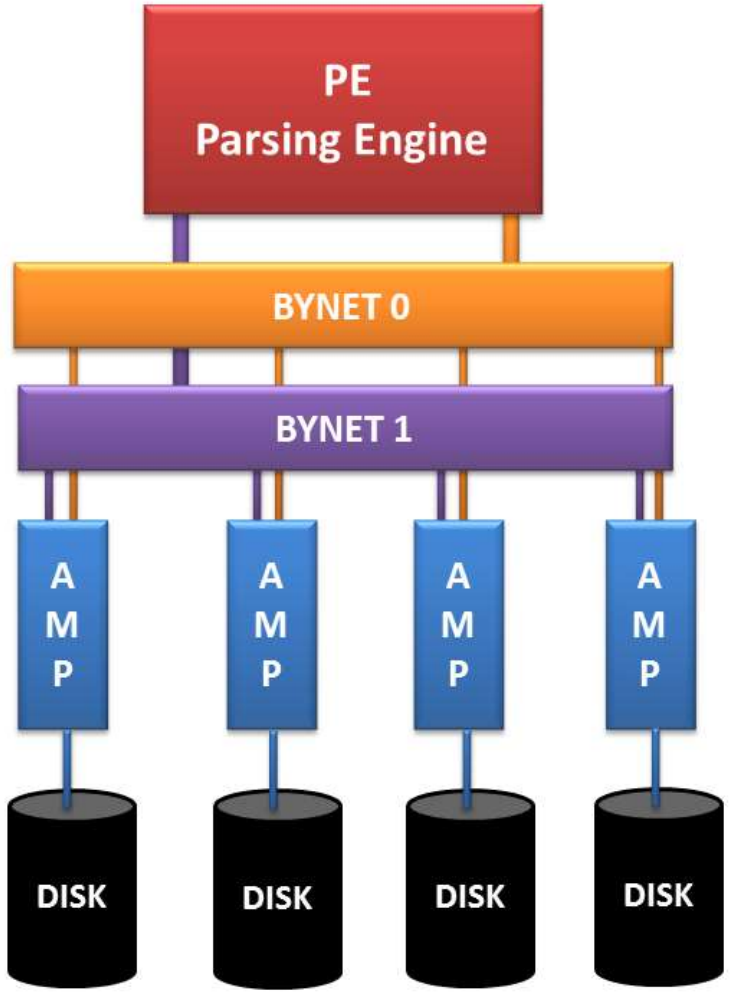
- Managing individual sessions (up to 120)
- Parsing and Optimizing your SQL requests
- Dispatching the optimized plan to the AMPs
- Input conversion (EBCDIC / ASCII) - if necessary
- Sending the answer set response back to the requesting client



Message parsing Layer (BYNET)
BYNET (**B**an**Y**an **Net**work) provides high performance networking Capabilities for MPP systems.

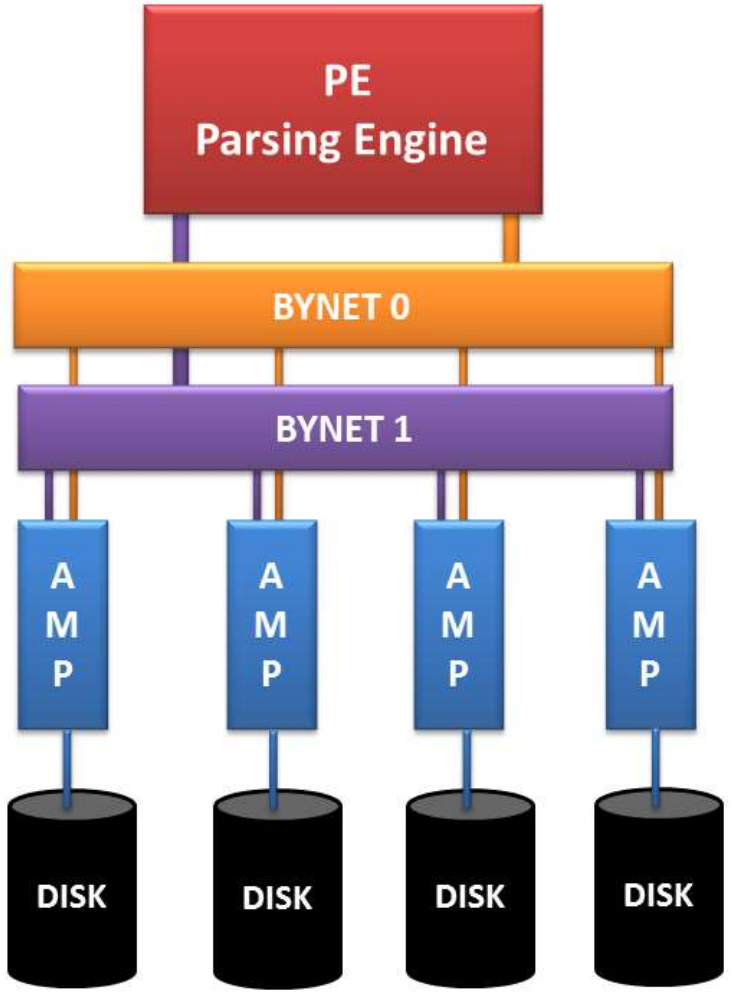
The Message Passing Layer is responsible for:
- Carrying messages between the AMPs and PEs
- Point-to-Point, Multi-Cast, and Broadcast communications
- Merging answer sets back to the PE
- Making Teradata parallelism possible

The Message Passing Layer is a combination of:
- Parallel Database Extensions (PDE) Software
- BYNET Software
- BYNET Hardware for MPP systems

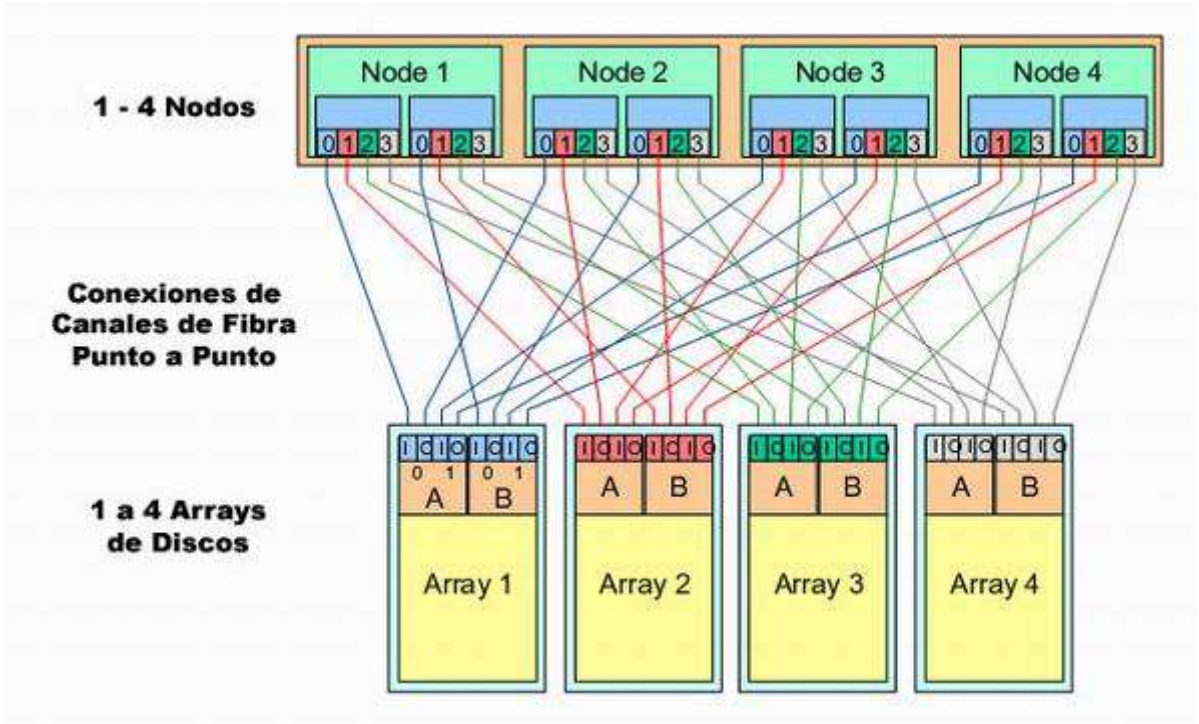


AMPs store and retrieve rows to and from disk
The AMPs are responsible for:

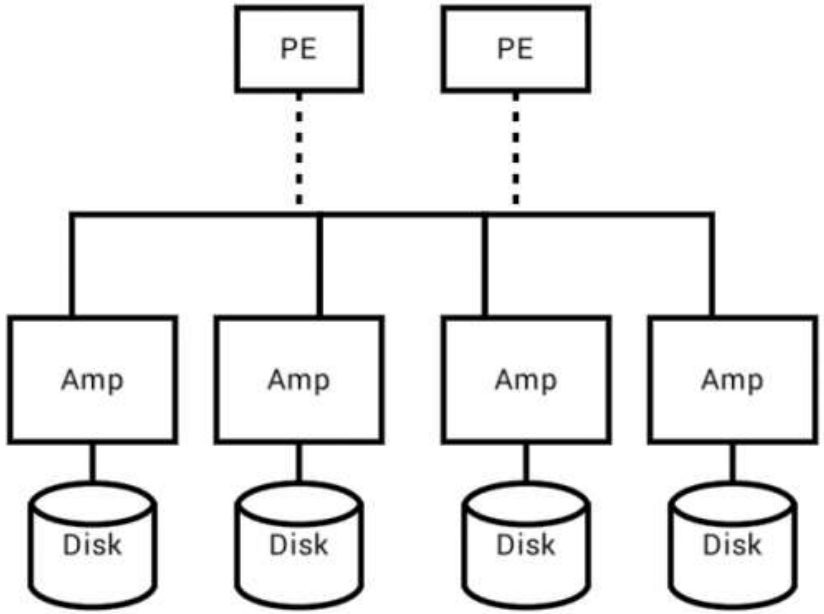
- Finding the rows requested
- Lock management
- Sorting rows
- Aggregating columns
- Join processing
- Output conversion and formatting
- Creating answer set for client
- Disk space management
- Accounting
- Special utility protocols
- Recovery processing



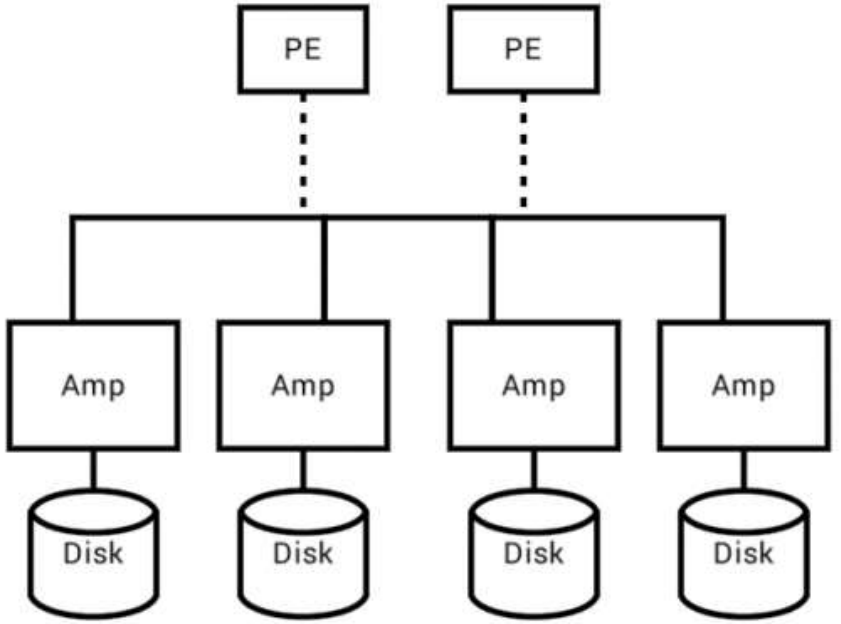
- A clique is a defined set of nodes that share a common set of disk arrays.
- All nodes in a clique must be able to access all Vdisks for all AMPs in the clique.
- A clique provides protection from a node failure.
- If a node fails, all vprocs will migrate to the remaining nodes in the clique (Vproc Migration).



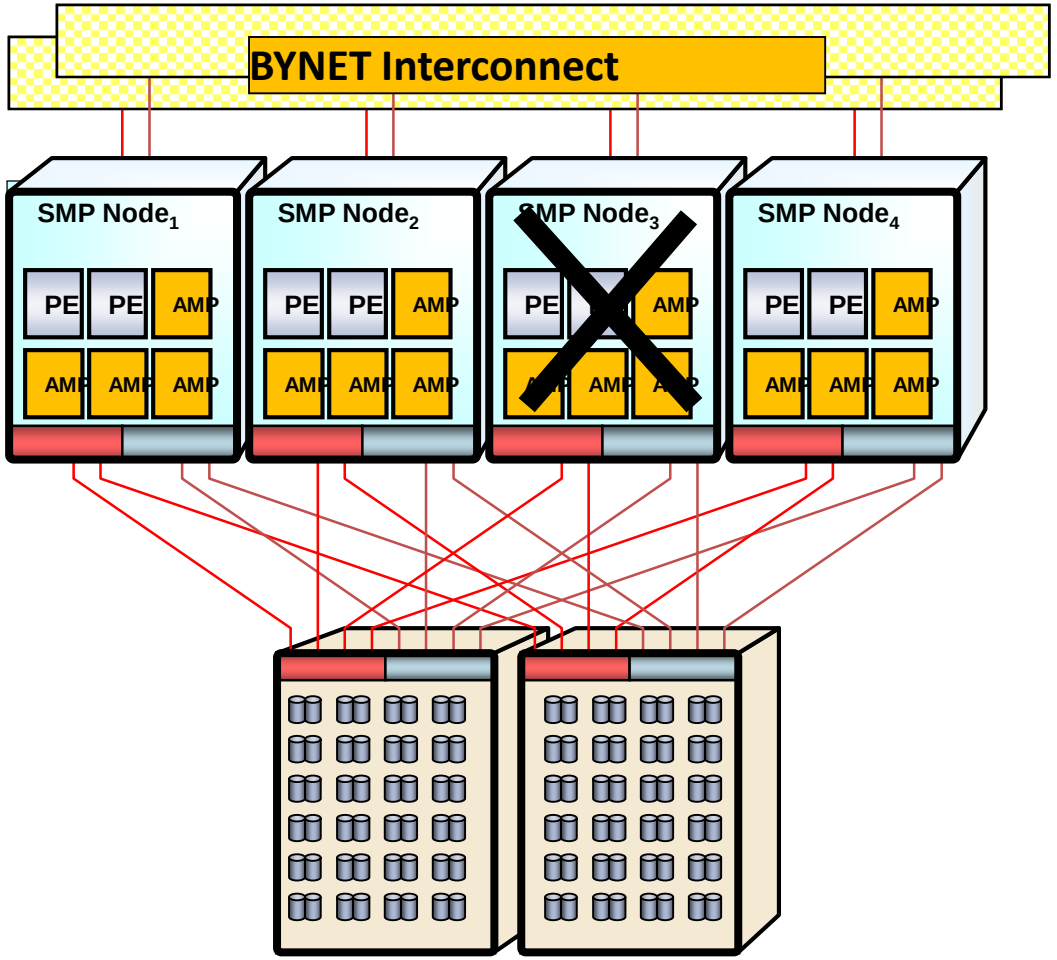
- The Parsing Engine dispatches request to insert a row.
- The Message Passing Layer insures that a row gets to the appropriate AMP (Access Module Processor).
- The AMP stores the row on its associated (logical) disk.
- An AMP manages a logical or virtual disk which is mapped to multiple physical disks in a disk array.



- The Parsing Engine dispatches a request to retrieve one or more rows
- The Message Passing Layer insures that the appropriate AMP(s) are activated.
- The AMP(s) locate and retrieve desired row(s) in parallel access.
- Message Passing Layer returns the retrieved rows to PE.
- The PE returns row(s) to requesting client application



- Teradata software provides high availability beyond other databases
  - Compensates for hardware failures:
    - Automatic failover for dynamic workload rebalancing (migrating VPROCS)
    - Online, continuous backup (Fallback)
  - Recycles before the operating system completes its reboot (multi-node system)



Teradata Database:

There are nine types of objects which may be found in a Teradata database/user.

- Tables – rows and columns of data
- Views – predefined subsets of existing tables
- Macros – predefined, stored SQL statements
- Triggers – SQL statements associated with a table
- Stored Procedures – program stored within Teradata
- User-Defined Function – function (C program) to provide additional SQL functionality
- Join and Hash Indexes – separate index structures stored as objects within a database Permanent Journals – table used to store before and/or after images for recovery

These objects are created, maintained and deleted using SQL

Objects definitions are stored in Database Dictionary.

Data Dictionary is entirely managed by Teradata database. These are accessed via Teradata supplied views.

Examples of data dictionary views are

- DBC.Tables – information about all tables
- DBC.Users – information about all users
- DBC.AllRights – information about access rights
- DBC.AllSpace – information about space utilization

Attributes that may be specified for a database:

- Perm Space – max amount of space available for tables, stored procedures, and UDFs
- Spool Space – max amount of work space available for requests
- Temp Space – max amount of temporary table space

A Teradata database is created with the CREATE DATABASE command.

Example

```
CREATE DATABASE Database_2 FROM Sysdba AS PERMANENT = 20e9, SPOOL = 500e6;
```

Notes: "Database_2" is owned by "Sysdba". A database is empty until objects are created within it.

A Teradata user is a database with an assigned password.
A Teradata user may logon to Teradata and access objects within:

- itself
- other databases for which it has access rights

Examples of attributes that may be specified for a user:

- Perm Space – max amount of space available for tables, stored procedures, and UDFs
- Spool Space – max amount of work space available for requests
- Temp Space – max amount of temporary table space

A user is an active repository while a database is a passive repository.

A user is created with the CREATE USER command.

Example

```
CREATE USER User_C FROM User_A AS PERMANENT = 100e6 ,SPOOL = 500e6 ,TEMPORARY = 150e6
,PASSWORD = lucky_day ;
```

"User_C" is owned by "User_A". A user is empty until objects are created within it.

User	Database
Unique Name	Unique Name
Password = Value	
Define and use Perm space	Define and use Perm space
Define and use Spool space	Define Spool space
Define and use Temporary space	Define Temporary space
Set Fallback protection default	Set Fallback protection default
Set Permanent Journal defaults	Set Permanent Journal defaults
Multiple Account strings	One Account string
Logon and Establish a session with priority	
May have a startup string	
Default database, dateform, timezone, and default character set Collation Sequence	

- You can only LOGON as a known User to establish a session with Teradata.
- Tables, Join/Hash Indexes, Stored Procedures, and UDFs require Perm Space.
- Views, Macros, and Triggers are definitions in the DD/D and require Perm Space.
- A database (or user) with zero Perm Space may have views, macros, and triggers, but cannot have tables, join/hash indexes, stored procedures, or user-defined functions.

SQL is a non procedural language and consists of

DDL (Data Definition Language) : Defines database structure(tables, users, views, macros, triggers etc.)
CREATE DROP ALTER

DML(Data manipulation Language) : Manipulates rows and data values
SELECT UPDATE INSERT DELETE

DCL (Data Control Language) : Grants and revokes access rights
GRANT REVOKE

Teradata SQL also includes Teradata Extensions to SQL
HELP SHOW EXPLAIN CREATE MACRO

Permanent Space

CREATE DATABASE CS_Tables FROM Customer_Service AS PERMANENT = 100e9 BYTES, ...

- Table rows, index subtable rows, join indexes, hash indexes, stored procedures, and UDFs use Perm space.
- Fallback protection uses twice the Perm space of No Fallback.
- Perm space is deducted from the owner's database space.
- Disk space is not reserved ahead of time, but is available on demand.
- Perm space is defined globally for a database.
- Perm space can be dynamically modified.
- The global limit divided by the number of AMPs is the per/AMP limit.
- The per/AMP limit cannot be exceeded.
- Good data distribution is crucial to space management

If we have 10 AMPs and permanent space of 100 GB, permanent space per AMP will be 10 GB

Sppol Space

CREATE USER Susan FROM CS_Users AS PERMANENT = 100e6 BYTES, SPOOL = 500e6 BYTES, PASSWORD = secret …

Spool space is work space acquired automatically by the system for intermediate query results or answer sets. – SELECT statements generally use Spool space. – Only INSERT, UPDATE, and DELETE statements affect table contents.

- The Spool limit cannot exceed the Spool limit of the original owner.
- The Spool limit is divided by the number of AMPS in the system, giving a perAMP limit that cannot be exceeded.

– "Insufficient Spool" errors often result from poorly distributed data or joins on columns with large numbers of non-unique values.
– Keeping Spool rows small and few in number reduces Spool I/O.

If we have 10 AMPs and spool space of 500 MB, Spool space per AMP will be 50 MB

Temporary Space

CREATE USER Susan FROM CS_Users AS PERMANENT = 100e6 BYTES, SPOOL = 500e6 BYTES, TEMPORARY = 150e6 BYTES, PASSWORD = secret

- Temporary space is space acquired automatically by the system when a "Global Temporary" table is used and materialized.
- The Temporary limit cannot exceed the Temporary limit of the original owner.
- The Temporary limit is divided by the number of AMPS in the system, giving a per-AMP limit that cannot be exceeded. – "Insufficient Temporary" errors often result from poorly distributed data or joins on columns with large numbers of non-unique values.

- Note: Volatile Temporary tables and derived tables utilize Spool space

If we have 10 AMPs and spool space of 150 MB, temporary space per AMP will be 15 MB

Create Table Example

```
CREATE TABLE Employee
(employee_number INTEGER NOT NULL ,
manager_emp_number INTEGER ,
dept_number SMALLINT ,
job_code INTEGER COMPRESS ,
last_name CHAR(20) NOT NULL ,
first_name VARCHAR (20) ,
hire_date DATE    FORMAT 'YYYY-MM-DD' ,
birth_date DATE   FORMAT 'YYYY-MM-DD' ,
salary_amount DECIMAL (10,2)
)
UNIQUE PRIMARY INDEX (employee_number) ,
INDEX (dept_number);
```

Other DDL Examples

```
CREATE INDEX (job_code) ON Employee ;
DROP INDEX (job_code) ON Employee ;
DROP TABLE Employee ;
```

Data Types

TYPE	Name	Bytes	Description
Date/Time	DATE	4	YYYYMMDD
	TIME (WITH ZONE)	6/8	HHMMSSZZ
	TIMESTAMP(WITH ZONE)	10/12	YYYYMMDDHHMMSSZZ
Numeric	DECIMAL or NUMERIC (n, m)	2, 4, 8	+ OR – (up to 18 digits V2R6.1 and prior) or 16 (up to 38 digits is V2R6.2 feature)
	BYTEINT	1	-128 to +127
	SMALLINT	2	-32,768 to +32,767
	INTEGER	4	-2,147,483,648 to +2,147,483,647
	BIGINT	8	-263 to +263 - 1 (+9,223,372,036,854,775,807)
Byte	BYTE(n)	0 – 64,000	
	VARBYTE (n)	0 – 64,000	
	BLOB	0 – 2 GB	
Character	CHAR (n)	0 – 64,000	
	VARCHAR (n)	0 – 64,000	
	LONG VARCHAR		same as VARCHAR(64,000)
	GRAPHIC	0 – 32,000	
	VARGRAPHIC	0 – 32,000	
	LONG VARGRAPHIC	same as VARGRAPHIC(32,000)	
	CLOB	0 – 2 GB Character Large Object (V2R5.1)	

Views

Views are pre-defined filters of existing tables consisting of specified columns and/or rows from the table(s).
A single table view:
– is a window into an underlying table
– allows users to read and update a subset of the underlying table
– has no data of its own
Employee(Table)

Employee Number	Manager Emp Number	Dept Number	Job Code	Last Name	First Name	Hire Date	Birth Date	Salary Amount
PK	FK	FK	FK					
1006	1019	301	312101	Stein	John	861015	631015	3945000
1008	1019	301	312102	Kanish	Carol	870201	680517	3925000
1005	0801	403	431100	Ryan	Loretta	861015	650910	4120000
1004	1003	401	412101	Johnson	Darlene	861015	560423	4630000
1007	1005	403	431101	Crusie	Tom	870102	470131	5970000

Emp_403
View

Emplouee Number	Manager Emp Number	Dept Number	Job Code	Last Name	First Name	Hire Date	Birth Date	Salary Amount
1005	0801	403	431100	Ryan	Loretta	861015	650910	4120000
1007	1005	403	431101	Crusie	Tom	870102	470131	5970000

Select

Employee Number	Manager Emp Number	Dept Number	Job Code	Last Name	First Name	Hire Date	Birth Date	Salary Amount
PK	FK	FK	FK					
1006	1019	301	312101	Stein	John	861015	631015	3945000
1008	1019	301	312102	Kanish	Carol	870201	680517	3925000
1005	0801	403	431100	Ryan	Loretta	861015	650910	4120000
1004	1003	401	412101	Johnson	Darlene	861015	560423	4630000
1007	1005	403	431101	Crusie	Tom	870102	470131	5970000

```
SELECT    Last_Name
         ,First_Name
FROM Employee
WHERE     Hire_Date = '1986-10-15' ;
```

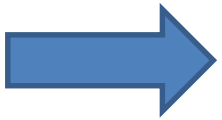
Last Name	First Name	Hire Date	Birth Date	Salary Amount
Stein	John	861015	631015	3945000
Ryan	Loretta	861015	650910	
Johnson	Darlene	861015	560423	4630000

Join Example

```
SELECT E.first_name, E.last_name
FROM  Employee E INNER JOIN Department D
ON   E.department_number = D.department_number
AND  D.department_name = 'R&D'
```

Employee Number	Manager Emp Number	Dept Number	Job Code	Last Name	First Name	Hire Date	Birth Date	Salary Amount
1006	1019	301	312101	Stein	John	861015	631015	3945000
1008	1019	301	312102	Kanish	Carol	870201	680517	3925000
1005	0801	403	431100	Ryan	Loretta	861015	650910	4120000
1004	1003	401	412101	Johnson	Darlene	861015	560423	4630000
1007	1005	403	431101	Crusie	Tom	870102	470131	5970000

Dept Number	Dept Name	Manager Emp Number
301	Marketing	1017
401	Sales	1019
403	R&D	1018



First Name	Ryan
Loretta	Ryan
Tom	Crusie

Macros

Macros allows a user to define a series of Teradata statements that they execute as a single transaction.

Features of Macros

- Macros are source code stored on the DBC.
- They can be modified and executed at will.
- They are re-optimized at execution time.
- They can be executed by interactive or batch applications.
- They are executed by one EXECUTE command.
- They can accept user-provided parameter values.

Benefits of Macros

- Macros simplify and control access to the system.
- They enhance system security.
- They provide an easy way of installing referential integrity.
- They reduce the amount of source code transmitted from the client application.
- They are stored in the Teradata DD/D and are available to all connected hosts.

To create a macro: CREATE MACRO Customer_List AS (SELECT customer_name FROM Customer;);
To execute a macro: EXEC Customer_List;
To Drop a macro : DROP MACRO Customer_List;
To Replace a macro : REPLACE MACRO Customer_List AS (SELECT customer_name, customer_number FROM Customer;);

Help Command

Databases and Users:

```
HELP DATABASE Customer_Service;
HELP USER Dave_Jones;
```

Tables, Views, Macros, etc.:

```
HELP TABLE Employee;
HELP VIEW Emp;
HELP MACRO Payroll_3;
HELP COLUMN Employee.*;
                Employee.last_name;
                Emp.*;
                Emp.last;
HELP INDEX Employee;
HELP TRIGGER Raise_Trigger;
HELP STATISTICS Employee;
HELP CONSTRAINT Employee.over_21;
HELP JOIN INDEX Cust_Order_JI;
HELP HASH INDEX Orders_HI;
HELP SESSION;
```

Show Command

SHOW commands display how an object was created.

Command	Returns Statement
SHOW TABLE table_name;	CREATE TABLE statement
SHOW VIEW view_name;	CREATE VIEW ...
SHOW MACRO macro_name;	CREATE MACRO ...
SHOW TRIGGER trigger_name;	CREATE TRIGGER …
SHOW PROCEDURE procedure_name;	CREATE PROCEDURE …
SHOW JOIN INDEX join_index_name;	CREATE JOIN INDEX …
SHOW HASH INDEX hash_index_name;	CREATE HASH INDEX …

```
SHOW TABLE Employee;
CREATE SET TABLE CUSTOMER_SERVICE.Employee, FALLBACK,
NO BEFORE JOURNAL,
NO AFTER JOURNAL,
CHECKSUM = DEFAULT
( employee_number INTEGER,
manager_employee_number INTEGER,
department_number INTEGER,
:
salary_amount DECIMAL(10,2) NOT NULL)
UNIQUE PRIMARY INDEX ( employee_number)
```

Explain facility

The EXPLAIN modifier in front of any SQL statement generates an English translation of the Parser's plan.
The request is fully parsed and optimized, but not actually executed.

EXPLAIN returns:
- Text showing how a statement will be processed (a plan)
- An estimate of how many rows will be involved
- A relative cost of the request (in units of time) This information is useful for:
- predicting row counts • predicting performance
- testing queries before production
- analyzing various approaches to a problem

EXPLAIN SELECT last_name, department_number FROM Employee ;

Explanation (partial):

3) We do an all-AMPs RETRIEVE step from CUSTOMER_SERVICE.Employee by way of an all-rows scan with no residual conditions into Spool 1, which is built locally on the AMPs. The size of Spool 1 is estimated to be 24 rows. The estimated time for this step is 0.15 seconds.

Locks

Locking prevents multiple users who are trying to change the same data at the same time from violating the data's integrity..
Teradata has four types of locks

- Exclusive – prevents any other type of concurrent access
- Write – prevents other reads, writes, exclusives
- Read – prevents writes and exclusives
- Access – prevents exclusive only

Three levels of database locking are provided:
- Database - locks all objects in the database
- Table - locks all rows in the table or view
- Row Hash - locks all rows with the same row hash

Lock types are automatically applied based on sql command.
- SELECT – applies a Read lock
- UPDATE – applies a Write lock
- CREATE TABLE – applies an Exclusive lock

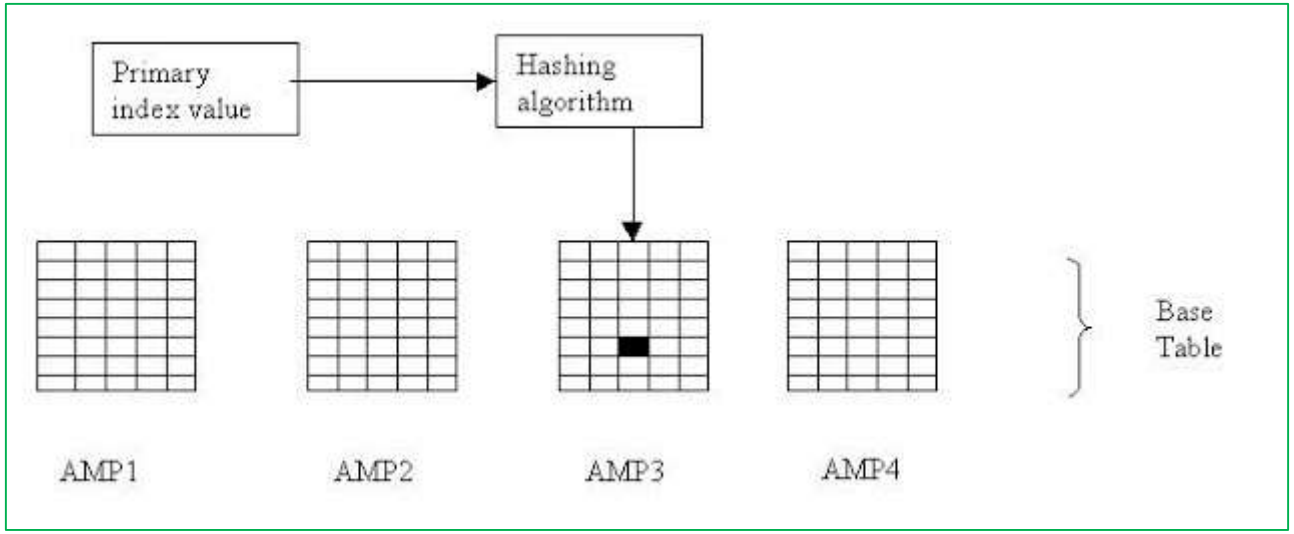
Indexes in Teradata

Teradata mainly support 5 type of indexes

* Unique Primary Index (UPI)
* Unique Secondary Index (USI)
* Non-Unique Primary Index (NUPI)
* Non-Unique Secondary Index (NUSI)
* Join Index

Primary Index : A primary index in Teradata RDBMS is required for row distribution and storage.

* Unique Primary Index (UPI))
* Non-Unique Primary Index (NUPI)



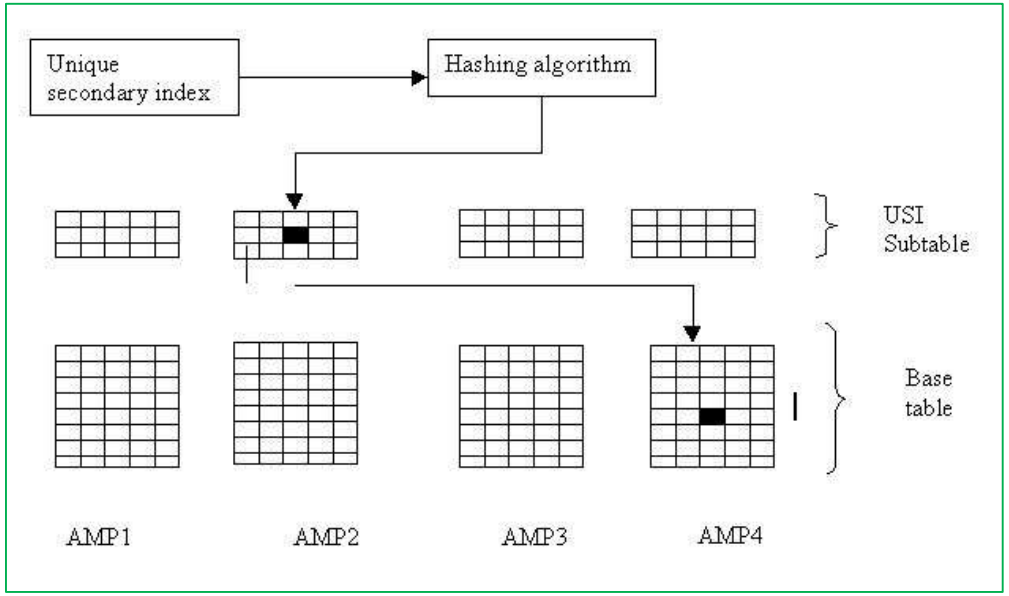
Indexes in Teradata (Secondary Index)

In addition to a primary index, up to 32 unique and non-unique secondary indexes can be defined to a table. Secondary indexes allow an alternate path to access the rows in the table which is less frequently used.

* Unique Secondary Index (USI)
* Non-Unique Secondary Index (NUSI)

A secondary index is a subtable that is stored in all AMPs separately from the primary table, It contains following information,

* RowIDs of the subtable rows
* Base table index column values
* RowIDs of the base table rows



Indexes in Teradata (Join Index)

A join index is an indexing structure which contains columns from multiple tables. It dramatically improves the performance ,

```
CREATE JOIN INDEX OrdCustIdx as
SELECT (c_custkey, c_name), (o_status, o_date, o_comment)
FROM  Orders LEFT JOIN Customer ON o_custkey = c_custkey
ORDER BY c_custkey
PRIMARY INDEX (c_name);
```

Any Questions