



CW 9118-9000-9250

**Version 2**

# **Teradata Basics**

## **Instructor Guide**

## Trademarks

AT&T and AT&T Globe are registered trademarks of AT&T Corporation.  
CICS, CICS/ESA, CICS/MVS, Data Base 2, DB2, IBM, MVS/ESA, MVS/XA, QMS, RACF, SQL/DS, VM/XA and VTAM are trademarks or registered trademarks of International Business Machines Corporation.  
DEC, VAX, MicroVax, and VMS are registered trademarks of Digital Equipment Corporation.  
Excelan is a trademark of Excelan, Incorporated  
EMPATH is an NCR trademark registered in the U.S. Patent and Trademark Office.  
Hewlett-Packard is a registered trademark of the Hewlett-Packard Company.  
INTELLECT and KBMS are trademarks of Trinzic Corporation.  
MICROSOFT, MS-DOS, DOS/V, and WINDOWS are registered trademarks of Microsoft Corporation.  
NCR is the name and mark of NCR Corporation.  
Sabre is the trademark of Seagate Technology, Inc.  
SAS and SAS/C are registered trademarks of the SAS Institute, Inc.  
Sun and SunOS are trademarks of Sun Microsystems, Inc.  
Teradata, Ynet, and DBC/1012 are registered trademarks of NCR Corporation.  
UNIX is a registered trademark of The Open Group.  
X and X/Open are registered trademarks of X/Open Company Limited.

---

The materials included in this book are a  
licensed product of NCR Corporation.

Copyright ©2000  
By NCR Corporation  
Rancho Bernardo, CA U.S.A.  
All Rights Reserved

**Printed in U.S.A.**

---

## **Introduction to Teradata Basics**

## **Course Objectives**

The facing page describes the objectives for this course.

## **Course Objectives**

**After completing this course, you will be able to:**

- **Describe the purpose and function of the Teradata Relational Database Management System (RDBMS).**
- **Navigate relational tables using primary and foreign keys.**
- **List the principal components of the RDBMS and describe their functions.**
- **Describe Teradata features that provide data protection and fault tolerance.**
- **Distinguish among various types of table indexes.**
- **List the hardware and software platforms that provide the foundation for the Teradata database.**

## Course Description

This one-day class delivers information about the architecture, capabilities, and purpose of the Teradata Relational Database Management system.

Topics include:

- System components and their functions
- Hardware and software platforms available
- Relational database concepts
- Features of Teradata RBDMS

The audience and format of the class is described on the facing page.

# **Course Description**

## **Who Should Attend**

**This course is designed for anyone who will be working with the Teradata database, including programmers, administrators, designers, support personnel, and end users.**

## **Class Format**

**This course consists of:**

- **One day of classroom instruction**
- **Review exercises following each module**
- **A course handbook in facing-page format**

## **Recommended Prerequisite Knowledge**

**There are no formal prerequisites for this class.**

## **Course Modules**

The modules in this course are listed on the facing page.



## **Course Modules**

<b>Module 1</b>	<b>Teradata Product Overview</b>
<b>Module 2</b>	<b>Relational Database Concepts</b>
<b>Module 3</b>	<b>Teradata Basics—Components and Architecture</b>
<b>Module 4</b>	<b>Teradata and the Data Warehouse</b>
<b>Module 5</b>	<b>Teradata Configurations</b>
<b>Module 6</b>	<b>Creating a Teradata Database</b>
<b>Module 7</b>	<b>Storing and Accessing Data Rows</b>
<b>Module 8</b>	<b>Primary Index Mechanics</b>
<b>Module 9</b>	<b>Secondary Indexed and Table Scans</b>
<b>Module 10</b>	<b>Data Protection</b>
<b>Module 11</b>	<b>Request Processing in a Parallel Environment</b>
<b>Module 12</b>	<b>Teradata Tools</b>

## References

The references for this course are listed on the facing page.

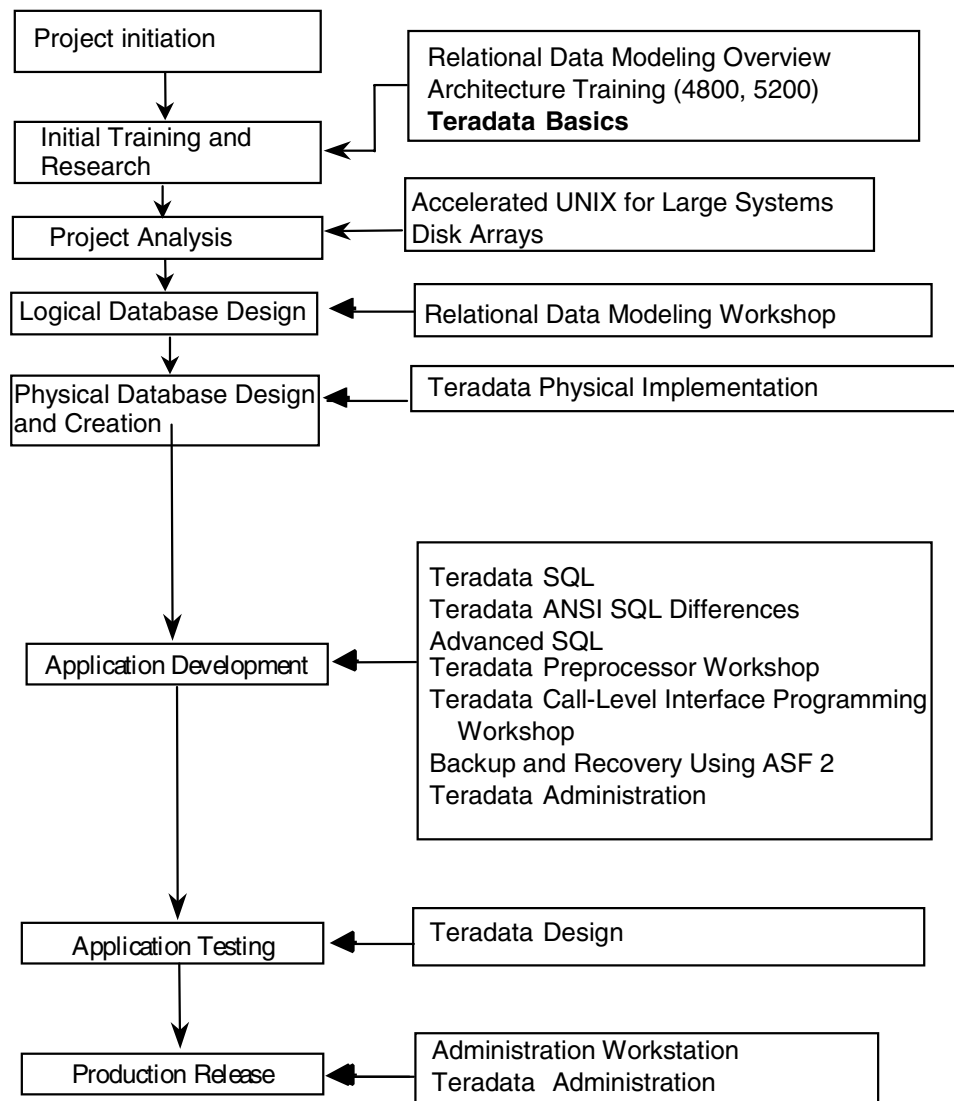
# References

<b>Book Number</b>	<b>Book Title</b>
<b>B035-1098-060A</b>	<b>Teradata RDBMS Release Summary Version 2 Release 4.0</b>
<b>B035-1091-060A</b>	<b>Introduction to Teradata RDBMS</b>
<b>B035-1093-060A</b>	<b>Teradata RDBMS Database Administration</b>
<b>B035-1092-060A</b>	<b>Teradata RDBMS Data Dictionary</b>
<b>B035-1100-060A</b>	<b>Teradata RDBMS Security Administration</b>
<b>B035-1101-060A</b>	<b>Teradata RDBMS SQL Reference: Volume 1, Fundamentals Volume 2, Statement and Transaction Processing Volume 3, Data Types Volume 4, Data Definition Statements Volume 5, Functions Volume 6, Data Manipulation Statements</b>
<b>B035-1102-060A</b>	<b>Teradata RDBMS Utilities, Volume 1, A-K Teradata RDBMS Utilities, Volume 2, L-Z</b>
<b>B035-1008-108A</b>	<b>V2R3.0.0 Teradata® RDBMS Database Design</b>

## **Project Phases and Training**

The facing page provides an overview of Teradata training.

# Project Phases and Training



## Notes

# Table of Contents

---

## ***Module 1***

### **Teradata Product Overview**

What is Teradata?.....	1-4
Teradata—A Brief History .....	1-6
How Large is a Trillion? .....	1-8
The Teradata Charter.....	1-10
Review Questions .....	1-12

## ***Module 2***

### **Relational Database Concepts**

What is a Database?.....	2-4
Relational Database.....	2-6
Primary Key.....	2-8
Foreign Key.....	2-10
Exercise—Answering Questions with a Relational Database .....	2-12
Advantages of a Relational Database Approach .....	2-14
Review Questions .....	2-16

## ***Module 3***

### **Teradata Basics—Components and Architecture**

Major Components of a Teradata System .....	3-4
The Parsing Engine.....	3-6
BYNET .....	3-8
The Access Module Processor (AMP) .....	3-10
Disk Arrays.....	3-12
Disk Array RAID Protection.....	3-14
Teradata Storage Process .....	3-16
Teradata Retrieval Process.....	3-18
Multiple Tables on Multiple AMPs .....	3-20
Linear Growth and Expandability .....	3-22
Teradata Parallelism .....	3-24
Teradata Functional Overview .....	3-26
Channel-Attached Client Software Overview .....	3-28
Network-Attached Client Software Overview.....	3-30
Teradata Objects.....	3-32
The Data Dictionary (DD).....	3-34
Structured Query Language (SQL) .....	3-36
Getting Information about Tables.....	3-38
The SELECT Statement .....	3-40
The Join Operation.....	3-42
Views.....	3-44
Single-table View .....	3-44
Multi-table Views .....	3-46
Macros .....	3-48
Macro Related Commands .....	3-50
The EXPLAIN Facility.....	3-52
Review: Teradata Features.....	3-54
Review: Teradata Objects.....	3-56
Review Questions .....	3-58

## ***Module 4***

### **Teradata and the Data Warehouse**

Evolution of Data Processing .....	4-4
Evolution in Data Capture and Storage.....	4-6
The Advantage of Using Detail Data.....	4-8
The Data Warehouse.....	4-10
Data Marts.....	4-12
Data Mart Pros and Cons.....	4-14
Teradata and the Data Warehouse.....	4-16
Review Questions .....	4-18



## ***Module 5***

### **Teradata Platforms**

Single-Node SMP .....	5-4
MPP System .....	5-6
Multi-Node Cliques .....	5-8
BYNET (for MPP) .....	5-10
Platforms .....	5-12
Review Questions .....	5-14

## ***Module 6***

### **Creating a Teradata Database**

A Teradata Database .....	6-4
A Teradata User .....	6-6
The Hierarchy of Databases .....	6-8
Creating Tables .....	6-10
Data Types .....	6-12
Access Rights and Privileges .....	6-14
Review Questions .....	6-16

## ***Module 7***

### **Storing and Accessing Rows**

Efficient Data Storage and Access .....	7-4
Storing Rows .....	7-6
Creating a Primary Index .....	7-8
Primary Index Values .....	7-10
Accessing Via a Unique Primary Index .....	7-12
Accessing Via a Non-Unique Primary Index .....	7-14
Primary Keys and Primary Indexes .....	7-16
Duplicate Rows .....	7-18
Row Distribution Using a Unique Primary Index (UPI) (Case 1) .....	7-20
Row Distribution Using a Non-Unique Primary Index (NUPI) (Case 2) .....	7-22
Row Distribution Using a Highly Non-Unique Index (NUPI) (Case 3) .....	7-24
Review Questions .....	7-26
Reference .....	7-28

## ***Module 8***

### **Primary Index Mechanics**

Hashing Down to the AMPs .....	8-4
A Hashing Example.....	8-6
The Hash Map.....	8-8
Identifying Rows.....	8-10
The Row-ID .....	8-12
Physical Data Block Layout.....	8-14
Physical Row Layout.....	8-16
Review Questions .....	8-18

## ***Module 9***

### **Secondary Index and Table Scans**

Secondary Indexes .....	9-4
Choosing a Secondary Index .....	9-6
Unique Secondary Index (USI) Access .....	9-8
Non-Unique Secondary Index (NUSI) Access.....	9-10
Comparison of Primary and Secondary Indexes.....	9-12
Full-Table Scans .....	9-14
Review Questions .....	9-16
Reference .....	9-18

## ***Module 10***

### **Data Protection**

Locks.....	10-4
Ruling on Locking.....	10-6
Access Locks .....	10-8
Fallback.....	10-10
Fallback Clustering (1 of 2) .....	10-12
Fallback Clustering (2 of 2) .....	10-14
Fallback vs. Non-Fallback Tables .....	10-16
Fallback and Disk Arrays.....	10-18
Recovery Journals for Down AMPs.....	10-20
Cliques (1 of 2).....	10-22
Cliques (2 of 2).....	10-24
Cliques and Clusters (1 of 4).....	10-26
Cliques and Clusters (2 of 4).....	10-28
Cliques and Clusters (3 of 4).....	10-30
Cliques and Clusters (4 of 4).....	10-32
Transient Journal .....	10-34
Archiving and Recovering Data.....	10-36
Permanent Journal.....	10-38
Using the Permanent Journal for Recovery .....	10-40
Review Questions .....	10-42

## ***Module 11***

### **Request Processing in a Parallel Environment**

Physical View .....	11-4
Logical View .....	11-6
AMP Queries.....	11-8
Single-AMP .....	11-8
Multi-AMP .....	11-8
All-AMP request .....	11-8
Single-AMP Query .....	11-10
Application to PE .....	11-12
PE to AMP .....	11-14
AMP to Vdisk.....	11-16
AMP to PE.....	11-18
PE to Application .....	11-20
All-AMP Query with a Sort.....	11-22
Application to PE .....	11-24
PE to AMPs .....	11-26
PE to AMPs .....	11-28
AMPs to Merge Process.....	11-30
AMP to Merge .....	11-32
PE to Application.....	11-34
Review Questions .....	11-36

## ***Module 12***

### **Teradata Tools**

Query Submitting Tools.....	12-4
Application Utilities .....	12-6
Administrative Tools.....	12-8
Programming Tools.....	12-10
Platform Support .....	12-12
Teradata SQL .....	12-14
Why Teradata? .....	12-16

***Appendix A***      **Review Questions/Solutions**

***Appendix B***      **Module 2 Exercise Solutions**

### Teradata Product Overview

After completing this module, you will be able to:

- Describe the purpose of the Teradata product.
- Give a brief history of the product.
- Identify support operating systems.

## Notes

*Attention Instructors! Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*

# Table of Contents

What is Teradata? .....	4
Teradata—A Brief History.....	6
How Large is a Trillion? .....	8
The Teradata Charter .....	10
Review Questions.....	12

# What is Teradata?

Teradata is a Relational Database Management System (RDBMS) for the world's largest commercial databases. Current technology permits databases that are hundreds of terabytes in size, making Teradata an obvious choice for large data warehousing applications. The Teradata system also may be as small as 10 gigabytes. With its parallelism and scalability, Teradata allows you to start small with a single node and grow large with many nodes through linear expandability.

Teradata has been available on UNIX based platforms since 1993. As of November 1998, it has also been available on Windows NT. Teradata is an open system and is compliant with industry ANSI standards.

Teradata is comparable to a large database server—with multiple client applications making inquiries against it concurrently.

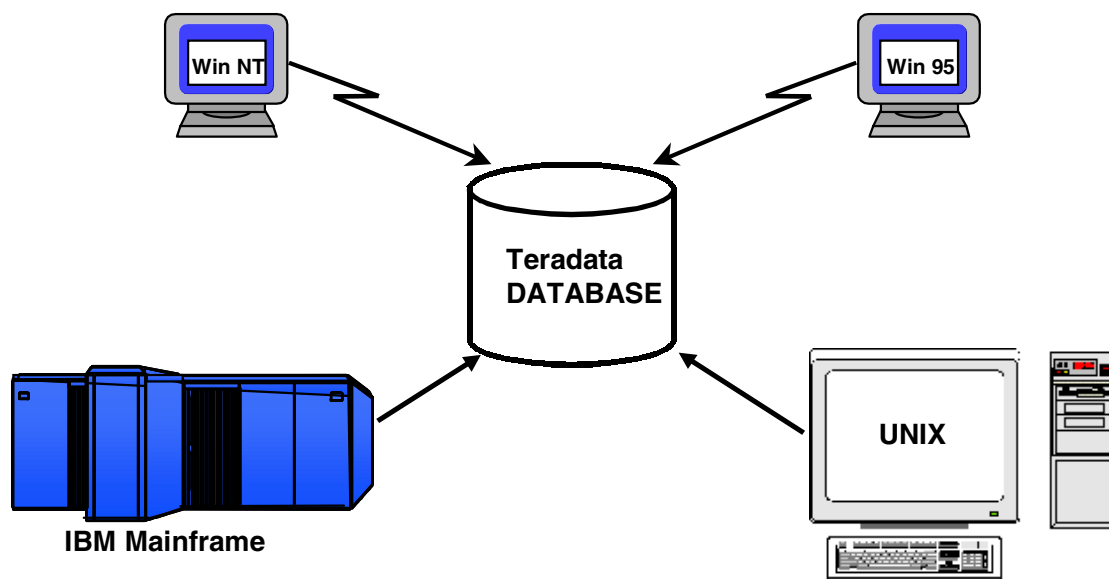
The ability to manage terabytes of data is accomplished using the concept of parallelism, wherein many individual processors perform smaller tasks concurrently to accomplish an operation against a huge repository of data. To date, only parallel architectures can handle databases of this size.



# What is Teradata?

**Teradata is a Relational Database Management System (RDBMS):**

- Designed to run the world's largest commercial databases
- Preferred solution for enterprise data warehousing
- Open, UNIX-based or NT-based system platforms
- Compliant with ANSI industry standards
- Runs on single or multiple nodes
- Acts as a database server to client applications throughout the enterprise
- Uses parallelism to manage terabytes of data



# Teradata—A Brief History

*Topics reflected on Certification Exam.*

Teradata Corporation was incorporated in 1979 in Los Angeles California. The corporate goal was the creation of a database computer that could handle billions of rows of data, up to and beyond a terabyte of data storage. It took five years of development before a product was shipped to the first customer in 1984. In 1982, YNET technology was patented for the parallelism that was at the heart of the architecture. The YNET was the interconnect that allowed hundreds of individual processors to share the same bandwidth.

In 1987, Teradata went public with its first stock offering. In 1989, Teradata partnered with NCR Corporation to build the next generation of database computers. Project 90 developed the advancements that made the 3600 system possible. Before either company could market its next generation product, NCR was purchased by AT&T corporation. Teradata was purchased and folded into the NCR structure. The new company was named AT&T GIS.

In 1996, AT&T spun off three separate companies; one of them was the former NCR, which then returned to its old name. By 1997, NCR had become the world leader in scalable data warehouse solutions, largely due to the strength of the Teradata database.

## **Teradata—A Brief History**

- 1979** Teradata Corp incorporated in Los Angeles, California. Development begins on a massively parallel database computer.
- 1982** YNET technology is patented.
- 1984** Teradata markets the first database computer, the DBC/1012. First system purchased by Wells Fargo Bank of California. Total revenue for year is \$3 million.
- 1987** First public offering of stock.
- 1989** Teradata and NCR partner on next generation of DBC.
- 1991** NCR Corp is acquired by AT&T. Teradata revenues at \$280 million.
- 1992** Teradata is merged into AT&T/NCR.
- 1996** AT&T spins off NCR Corporation with Teradata product.
- 1997** Teradata database becomes industry leader in scalable data warehousing.

# How Large is a Trillion?

*Topics reflected on Certification Exam.*

Teradata was the first commercial database system to support a trillion bytes of data.

“Tera” is a prefix used to indicate “trillion.” (This prefix is based on the Greek word *teras*, meaning “monster.”)

It is hard to imagine the size of a trillion. Most people are comfortable with kilobytes, megabytes, and even gigabytes, but terabytes are another 3 orders of magnitude. To put it in perspective, the life span of the average person is 2.5 Gigaseconds (or said differently 2,500,000,000 seconds). A tera-second is 31,688 years!

## How Large is a Trillion?

<b>1 Kilobyte</b>	<b>= <math>10^3</math> = 1000 bytes</b>
<b>1 Megabyte</b>	<b>= <math>10^6</math> = 1,000,000 bytes</b>
<b>1 Gigabyte</b>	<b>= <math>10^9</math> = 1,000,000,000 bytes</b>
<b>1 Terabyte</b>	<b>= <math>10^{12}</math> = 1,000,000,000,000 bytes</b>
<b>1 Petabyte</b>	<b>= <math>10^{15}</math> = 1,000,000,000,000,000 bytes</b>
<b>1 Exabyte</b>	<b>= <math>10^{18}</math> = 1,000,000,000,000,000,000 bytes</b>
<b>1 Zetabyte</b>	<b>= <math>10^{21}</math> = 1,000,000,000,000,000,000,000 bytes</b>
<b>1 Yottabyte</b>	<b>= <math>10^{24}</math> = 1,000,000,000,000,000,000,000,000 bytes</b>

**1 million seconds = 11.57 days**  
**1 billion seconds = 31.6 years**  
**1 trillion seconds = 31,688 years**

**1 million inches = 15.7 miles**  
**1 trillion inches = 15,700,000 miles (30 roundtrips to the moon)**

**1 million square inches = .16 acres = .0002 square miles**  
**1 trillion square inches = 249 square miles (larger than Singapore)**

**\$1 million = < \$ .01 for every person in U.S.**  
**\$1 billion = \$ 3.64 for every person in U.S.**  
**\$1 trillion = \$ 3,636 for every person in U.S.**

# The Teradata Charter

*Topics reflected on Certification Exam.*

Teradata founders surveyed potential business partners/customers to determine their needs and requirements that included the following:

- **Relational:** the relational model has become the accepted standard for database design. It provides the flexibility needed to leverage business information.
- **Large capacity database machine:** data storage requirements are large as a result of the need for processing the large amounts of detail data for decision support.
  - Billions of rows
  - Terabytes of data
  - Hundreds of processing engines
- **Performance:** as table size increased, early relational systems suffered severe performance limitations. Teradata addresses performance issues of large databases.
- **Single data store for multiple client host:** instead of replicating a database for different hosts, store it once, and use it for all clients.
- **Network connectivity:** connect easily to network-attached host systems.
- **Standard access language (SQL):** SQL has been adapted as the industry standard for relational databases.
- **Manageable growth:** businesses want a system that is linearly expandable to allow for growth without performance drop-off.
- **Fault tolerance:** automatically detect and recover from one or more hardware failures.
- **Data integrity:** to guarantee the integrity of the data, transactions are either completed or, if a fault occurs, rolled back.

# **The Teradata Charter**

- **Relational database**
- **Enormous capacity**
  - Billions of rows
  - Terabytes of data
- **High-performance parallel processing**
- **Single database server for multiple clients**
- **Network and mainframe connectivity**
- **Industry standard access language (SQL)**
- **Manageable growth via modularity**
- **Fault tolerance at all levels of hardware and software**
- **Data integrity and reliability**

## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.



## **Review Questions**

- 1. When was the first Teradata product sold?**
- 2. One trillion is written as a 1 (one) followed by how many zeroes?**
- 3. Which feature of Teradata permits high performance even against enormous databases?**
- 4. Which two operating systems can be the platform for Teradata?**
- 5. Is Teradata a client or a server?**
- 6. What operating systems can be used with Teradata?**
- 7. Name at least 3 features of Teradata.**

# Notes

### Relational Database Concepts

**After completing this module, you will be able to:**

- **Describe the general characteristics of a database.**
- **Define the terms associated with relational databases: column, row, entity, primary key, foreign key, join, null.**
- **List the advantages of a relational database.**

## Notes

*Attention Instructors! Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*

## Table of Contents

What is a Database? .....	4
Relational Database .....	6
Primary Key .....	8
Foreign Key .....	10
Exercise—Answering Questions with a Relational Database .....	12
Advantages of a Relational Database Approach .....	14
Review Questions.....	16

# What is a Database?

A **database** is a collection of permanently stored data used by an application or enterprise.

A database contains **logically related data**, which means that the database was created with a purpose in mind. A database supports **shared access** by many users. One characteristic of a database is that many people use it, often for many different purposes. A database also is **protected** to control access and **managed** to retain its value and integrity.

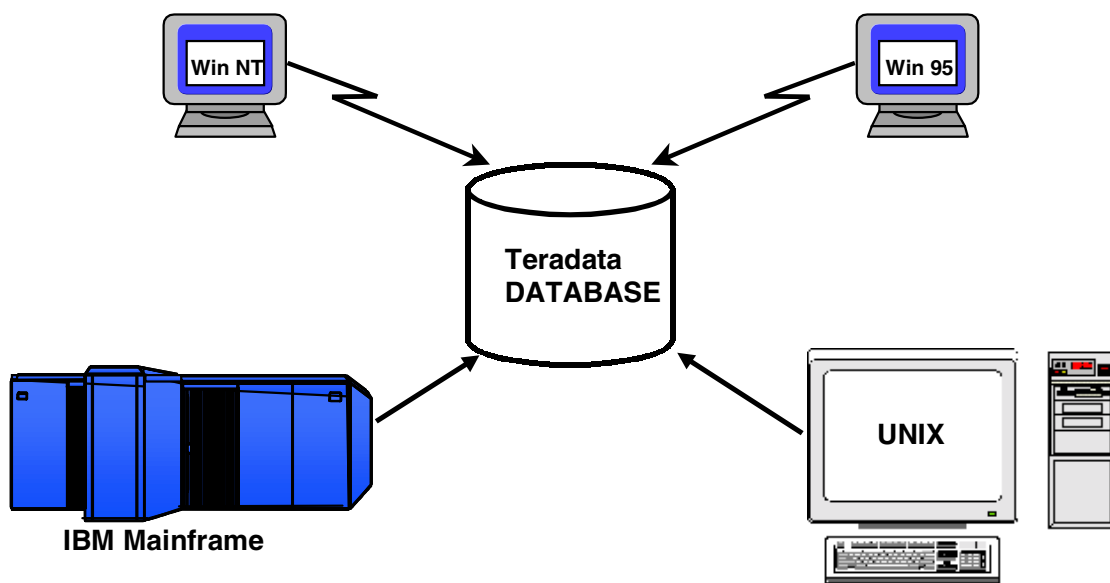
One example of a database is payroll data that includes the names of the employees, their employee numbers, and their salary history. This database is logically related—it's all about payroll. It must have shared access, since it will be used by the payroll department to generate checks, and also by management to make decisions. This database must be protected; much of the information is confidential and managed to ensure the accuracy of the records.

Later in this course we will provide another definition of database that is specific to Teradata.

# What is a Database?

**Database—A collection of permanently stored data that is:**

- **Logically related—data relates to other data.**
- **Shared—many users may access data.**
- **Protected—access to data is controlled.**
- **Managed—data has integrity and value.**



# Relational Database

*Topics reflected on Certification Exam.*

The key to understanding relational databases is the concept of the **table**, which is made up of **rows** and **columns**.

A **column** always contains like data. In the example on the following page, the column named LAST NAME contains last name and never anything else. Column position in the table is arbitrary.

A **row** is one instance of all the columns of a table. In our example, all information about a single employee is in one row. The sequence of the rows in a table is arbitrary.

In a relational database, tables are defined as a named collection of one or more named columns that can have zero or many rows of related information.

Notice that the information in each row of the example table refers to only one person. There are no rows with data on two people, nor are there rows with information on anything other than people. This may seem obvious, but the concept underlying it is very important.

A missing data value is referred to as a **null** value.

Each row represents an occurrence of an **entity** defined by the table. An **entity** is defined as a person, place, thing, or event about which the table contains information. In this case, the entity is the *employee*.



# Relational Databases

**A Relational Database consists of a set of logically related tables.**

**A table is a two dimensional representation of data consisting of rows and columns.**

Column  
↓

EMPLOYEE

EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT 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	Kanieski	Carol	870201	680517	3925000
1005	0801	403	431100	Ryan	Loretta	861015	650910	4120000
1004	1003	401	412101	Johnson	Darlene	861015	560423	4630000
1007				Villegas	Arnando	870102	470131	5970000
1003	0801	401	411100	Trader	James	860731	570619	4785000

Row →

**The employee table has:**

- **Nine columns of data**
- **Six rows of data—one per employee**
- **No prescribed order for the rows of the table**
- **Only one row format for the entire table**
- **Missing data values represented by nulls**

# Primary Key

Tables, made up of rows and columns, represent **entities** or relationships. Entities are the people, places, things, or events that the Entity Tables model. Each table holds only one kind of row, and each row is uniquely identified within a table by a **Primary Key (PK)**.

## Primary Key Rules

A Primary Key **uniquely identifies** each row in a table. A Primary Key is **required**, because each row within a table must be able to be uniquely identified

**No duplicate values are allowed.** The Primary Key for the EMPLOYEE table is the employee number, because no two employees can have the same number.

Because it is used for identification, the Primary Key **cannot be null.** There must be something in that field to uniquely identify each occurrence.

Primary Key values **cannot be changed.** Historical information, as well as relationships with other entities, may be lost if a PK value is changed or re-used.

A Primary Key **can include more than one column.**

**Only one** Primary Key is allowed per table.

## Selecting a Primary Key

Each column of a row is called an **attribute** of that row. A database designer can select any attribute to be a Primary Key but, as a result of the rules listed above, many attributes will not qualify as a Primary Key candidate. For example, we could have selected **Last Name** as the PK of the EMPLOYEES table, but as soon as the company hires two people with the same last name, the PK is no longer unique. Even if we made the PK **Last Name and First Name** (possible since PKs can be made up of more than one column) we could still have two employees with the same name. Moreover, some employees may choose to change their last names.

Many data modelers recommend using system-assigned sequential integers for Primary Keys. This assures uniqueness and gives users control over Primary Key values.

## Primary Key

**Primary Key (PK) values uniquely identify each row in a table.**

**EMPLOYEE**

EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT NUMBER	JOB CODE	LAST NAME	FIRST NAME	HIRE DATE	BIRTH DATE	SALARY AMOUNT
PK								
1006	1019	301	312101	Stein	John	861015	631015	3945000
1008	1019	301	312102	Kanieski	Carol	870201	680517	3925000
1005	0801	403	431100	Ryan	Loretta	861015	650910	4120000
1004	1003	401	412101	Johnson	Darlene	861015	560423	4630000
1007				Villegas	Arnando	870102	470131	5970000
1003	0801	401	411100	Trader	James	860731	570619	4785000

### Primary Key Rules

- **A Primary Key is required for every table.**
- **Only one Primary Key is allowed in a table.**
- **Primary Keys may consist of one or more columns.**
- **Primary Keys cannot have duplicate values.**
- **Primary Keys cannot be null.**
- **Primary Keys are considered “non-changing” values.**

## Foreign Key

Relational databases permit data values to associate across more than one table. A **Foreign Key (FK)** value identifies table relationships.

On the facing page you will see that the employee table has three FK columns, one of which models the relationship between employees and their departments. A second FK column models the relationship between employees and their jobs.

A third FK column is used to model the relationship between employees and each other. This is called a recursive relationship.

### Foreign Key Rules

- Duplicate values are allowed in a FK column.
- Missing values are allowed in a FK column.
- Values may be changed in a FK column.
- Each FK value must exist as a Primary Key.

Note that Department\_Number is the **Primary Key** for the DEPARTMENT table.

# Foreign Key

Foreign Key (FK) values identify table relationships.

EMPLOYEE (partial listing)

EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT 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	Kanieski	Carol	870201	680517	3925000
1005	0801	403	431100	Ryan	Loretta	861015	650910	4120000
1004	1003	401	412101	Johnson	Darlene	861015	560423	4630000
1007				Villegas	Arnando	870102	470131	5970000
1003	0801	401	411100	Trader	James	860731	570619	4785000

DEPARTMENT

DEPARTMENT NUMBER	DEPARTMENT NAME	BUDGET AMOUNT	MANAGER EMPLOYEE NUMBER
PK			FK
501	marketing sales	80050000	1017
301	research and development	46560000	1019
302	product planning	22600000	1016
403	education	93200000	1005
402	software support	30800000	1011
401	customer support	98230000	1003
201	technical operations	29380000	1025

- FKs are optional—not all tables have them.
- More than one FK is allowed per table.
- FKs can be made up of more than one column.
- Duplicate values may be allowed.
- Missing (null) FK values may be allowed.
- Changes to FKs are allowed.
- Each FK value must exist somewhere as a PK value.

## Exercise—Answering Questions with a Relational Database

A relational database is a collection of **relational tables** stored in a single installation of a **Relational Database Management System (RDBMS)**.

The words, “management system,” indicate that this is a relational database that indicates underlying software that provides additional expected functions such as transaction integrity, security, and journaling. The Teradata Database is a Relational Database Management System.

Relational databases do not use access paths to locate data; data **connections are made by data values**. Data connections are made by matching values in one column with the values in a corresponding column in another table. In relational terminology, this connection is referred to as a **JOIN**.

The diagrams on the facing page show how the values in one table may be matched to values in another table.

Both tables have a column named “Department Number,” which allows the database to answer questions like, “What is the name of the department in which an employee works?”

One reason relational databases are so powerful is that, unlike other databases, they are based on a mathematical model and implement a query language solidly founded in **set theory**. (Dr. Edgar Codd developed this model.)

To sum up, a relational database is a **collection of tables**. The data contained in the tables can be associated using data values—**columns with matching data values**.

## Exercise—Answering Questions with a Relational Database

**EMPLOYEE** (partial listing)

EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT 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	Kanieski	Carol	870201	680517	3925000
1005	0801	403	431100	Ryan	Loretta	861015	650910	4120000
1004	1003	401	412101	Johnson	Darlene	861015	560423	4630000
1007			432101	Villegas	Arnando	870102	470131	5970000
1003	0801	401	411100	Trader	James	860731	570619	4785000

**DEPARTMENT**

DEPARTMENT NUMBER	DEPARTMENT NAME	BUDGET AMOUNT	MANAGER EMPLOYEE NUMBER
PK			FK
501	marketing sales	80050000	1017
301	research and development	46560000	1019
302	product planning	22600000	1016
403	education	93200000	1005
402	software support	30800000	1011
401	customer support	98230000	1003
201	technical operations	29380000	1025

1. Name the department in which James Trader works.
2. Who manages the Education Department?
3. Identify by name an employee who works for James Trader.
4. James Trader manages which department?

# Advantages of a Relational Database Approach

## Flexible

Flexibility provides substantial benefits:

- The user does not need to know the access path of the data; the RDBMS keeps track of where everything is.
- Relational databases use atomic data —breaking data down into its smallest parts to allow maximum flexibility in selecting and using data.

## Responds quickly

In a traditional database, adding a field means that all programs that use the database must be rewritten to become aware of the new data structure. In a relational RDBMS, programs do not need to be re-written when a field is added.

## Data-driven

Relational databases are designed to represent a business and its practices not the application or the computer system.

## Business-oriented

The two tables we have looked at, EMPLOYEE and DEPARTMENT, are organized to reflect the way the business really works.

## Simple and easy to use

Simplicity is useful not only to the people who ask the questions, but also to the people who have to figure out how to retrieve information from the database.

## Easier to build applications

Relational databases make the data do more work. Programs and transactions are simpler, which makes it easier to build applications.

## Easy to understand

Understanding how a RDBMS functions is not necessary, any more than it's necessary to know the actual gear ratios of your automatic transmission. All the user needs to know is what it can do.

## Support this trend toward end-user computing

The trend is moving away from organizations funneling all data requests through a few people who know how the system works. As systems get easier to use, more people have access to them. This is called the "democratization of data."



# Advantages of a Relational Database Approach

Advantages to a Relational Database compared to other database methodologies include:

- Allowing businesses to quickly respond to changing conditions
- More flexible than other types
- Being data-driven vs. application driven
- Modeling the business, not the processes
- Being easy to use
- Makes applications easier to build
- Being easy to understand
- Supporting trend toward end-user computing

## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.

## **Review Questions**

**Match each term with its definition:**

\_\_\_1. Database

\_\_\_2. Table

\_\_\_3. Relational database

\_\_\_4. Primary Key

\_\_\_5. Null

\_\_\_6. Foreign Key

- a. A set of columns that uniquely identify a row
- b. A set of logically related tables
- c. One or more columns that are a PK somewhere in the database
- d. The absence of a value
- e. A two-dimensional array of rows and columns
- f. A collection of permanently stored data

## Notes

### **Teradata Basics— Components and Architecture**

**After completing this module, you will be able to:**

- **List and describe the major components of the Teradata architecture.**
- **Describe how the components interact to manage incoming and outgoing data.**
- **Provide a general description of parallelism for the Teradata RDBMS.**
- **Describe software requirements for Teradata clients.**
- **Define the three Teradata database objects: tables, views, and macros.**
- **Describe the Data Dictionary**

## Notes

*Attention Instructors!*

*This module describes many fundamental concepts essential to understanding the subsequent modules. Ensure that participants have a clear understanding of these concepts.*

*Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*

## Table of Contents

Major Components of a Teradata System .....	4
The Parsing Engine .....	6
BYNET .....	8
The Access Module Processor (AMP) .....	10
Disk Arrays .....	12
Disk Array RAID Protection .....	14
Teradata Storage Process .....	16
Teradata Retrieval Process .....	18
Multiple Tables on Multiple AMPs .....	20
Linear Growth and Expandability .....	22
Teradata Parallelism .....	24
Teradata Functional Overview .....	26
Channel-Attached Client Software Overview .....	28
Network-Attached Client Software Overview .....	30
Teradata Objects .....	32
The Data Dictionary (DD) .....	34
Structured Query Language (SQL) .....	36
Getting Information about Tables .....	38
The SELECT Statement .....	40
The Join Operation .....	42
Views .....	44
Single-table View .....	44
Multi-table Views .....	46
Macros .....	48
Macro Related Commands .....	50
The EXPLAIN Facility .....	52
Review: Teradata Features .....	54
Review: Teradata Objects .....	56
Review Questions .....	58

# Major Components of a Teradata System

*Topics reflected on Certification Exam.*

Until now we have discussed relational databases in terms of how the user perceives them—as a collection of tables that relate to one another. It's time to describe the physical components of the system.

## Parsing Engine

The Parsing Engine (PE) is a virtual processor that interprets SQL requests, receives input records, and passes data. To do that it sends the messages through the BYNET to the AMPs.

## BYNET

The BYNET can be thought of as a highly sophisticated bus. It is implemented either as hardware or software, depending on the platform used. It determines which AMP(s) (Access Module Processor) should receive a message.

## Access Module Processor (AMP)

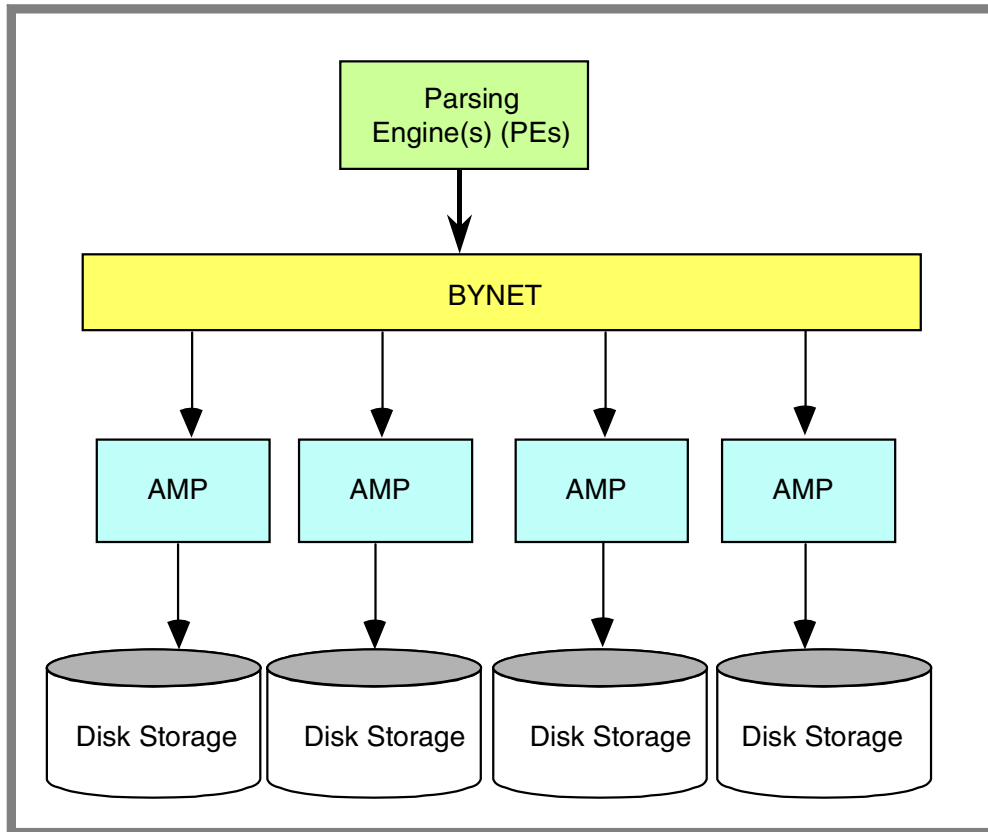
The AMP is a virtual processor designed for and dedicated to managing a portion of the entire database. It performs all database management functions such as sorting, aggregating, and formatting data. The AMP receives data from the PE, formats rows, and then distributes them to the disk storage units it controls. The AMP also retrieves the rows requested by the parsing engine.

## Disks

Disks are disk drives associated with an AMP that store the data rows. On current systems, they are usually implemented using a disk array.



## Major Components of a Teradata System



# The Parsing Engine

*Topics reflected on Certification Exam.*

A **Parsing Engine (PE)** is a virtual processor<sup>1</sup> (vproc) made up of the following software components: **Session Control, the Parser** (including the Optimizer), and the **Dispatcher**.

Once a valid session has been established, the PE is the component that manages the dialogue between the client application and the RDBMS.

## Session Control

Provides user session management (logon and logoff). Logon takes a textual request for session authorization, verifies it, and returns a yes or no answer. Logoff terminates any ongoing activity and deletes the session's context. When connected to an EBCDIC host, the PE converts incoming data to the internal 8-bit ASCII used by the Teradata RDBMS, thus allowing input values to be properly evaluated against the database data.

## Parser

When a PE receives an SQL request from a client application, the Parser interprets the statement, checks it for proper SQL syntax and evaluates it semantically. The PE also consults the Data Dictionary to ensure that all objects and columns exist and that the user has authority to access these objects.

The Parser has an **Optimizer** whose role is to develop the least expensive plan to return the requested response set. Processing alternatives are evaluated and the fastest alternative is chosen. This alternative is converted to executable steps, to be performed by the AMPs, which are then passed to the dispatcher.

## Dispatcher

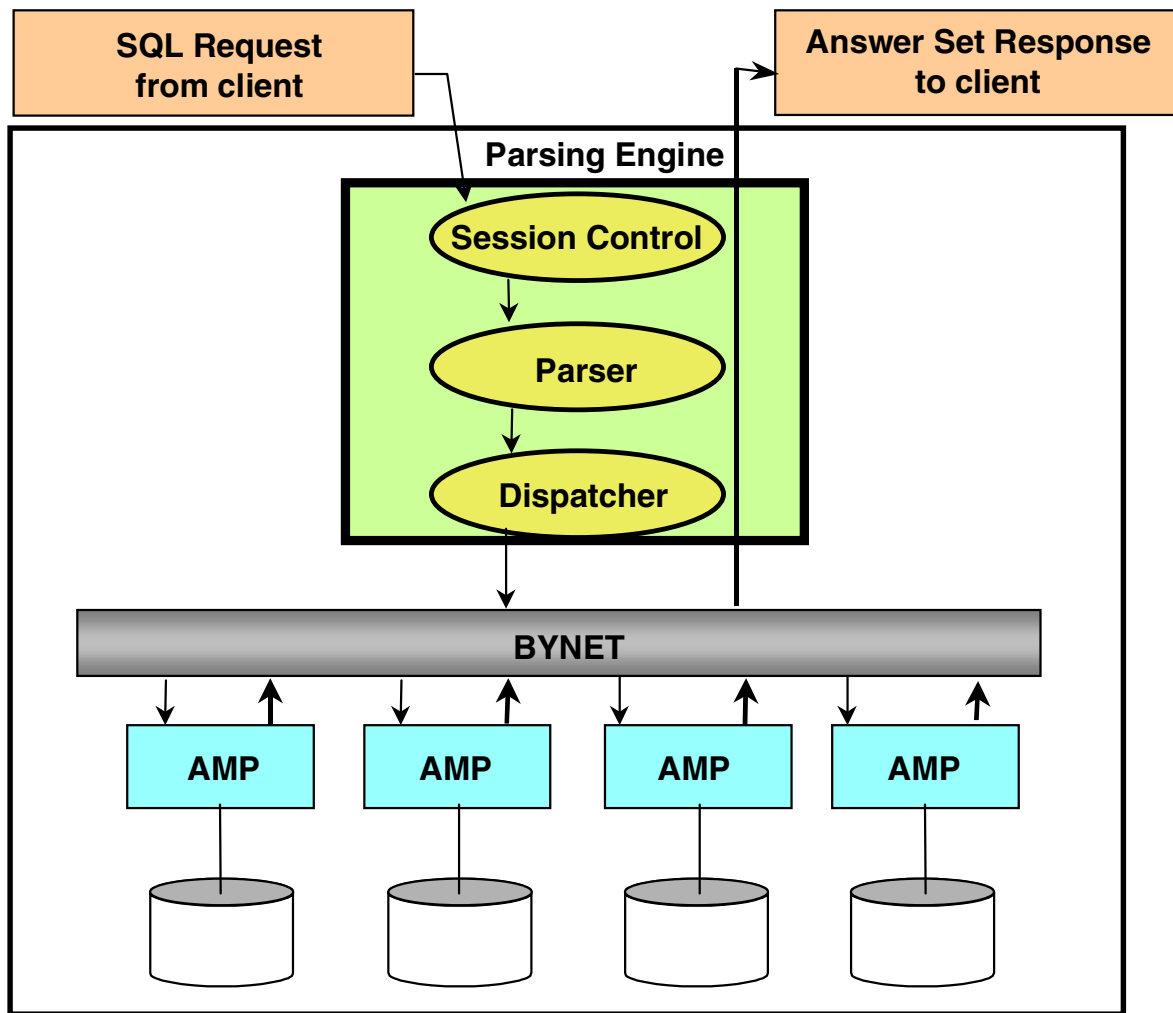
The Dispatcher controls the sequence in which the steps are executed and passes the steps on to the BYNET. It is composed of **execution-control and response-control tasks**. Execution control receives the step definitions from the Parser and transmits them to the appropriate AMP(s) for processing, receives status reports from the AMPs as they process the steps, and passes the results on to response control once the AMPs have completed processing. Response control returns the results to the user. The Dispatcher sees that all AMPs have finished a step before the next step is dispatched.

Depending on the nature of the SQL request, a step will be sent to one AMP, a few AMPs, or all AMPs.

---

<sup>1</sup> The versatility of Teradata RDBMS is based on virtual processors (vprocs) that eliminate dependency on specialized physical processors. These vprocs are a set of software processes that run on a node under the Teradata Parallel Database Extensions (PDE) and the multitasking environment of the operating system.

## The Parsing Engine



The Parsing Engine is responsible for:

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

# BYNET

*Topics reflected on Certification Exam.*

The **BYNET** handles the internal communication of the Teradata RDBMS.

All communication between PEs and AMPs is done via the BYNET.

When the PE dispatches the steps for the AMPs to perform, they are dispatched onto the BYNET. The messages are routed to the appropriate AMP(s) where results sets and status information are generated. This response information is also routed back to the requesting PE via the BYNET.

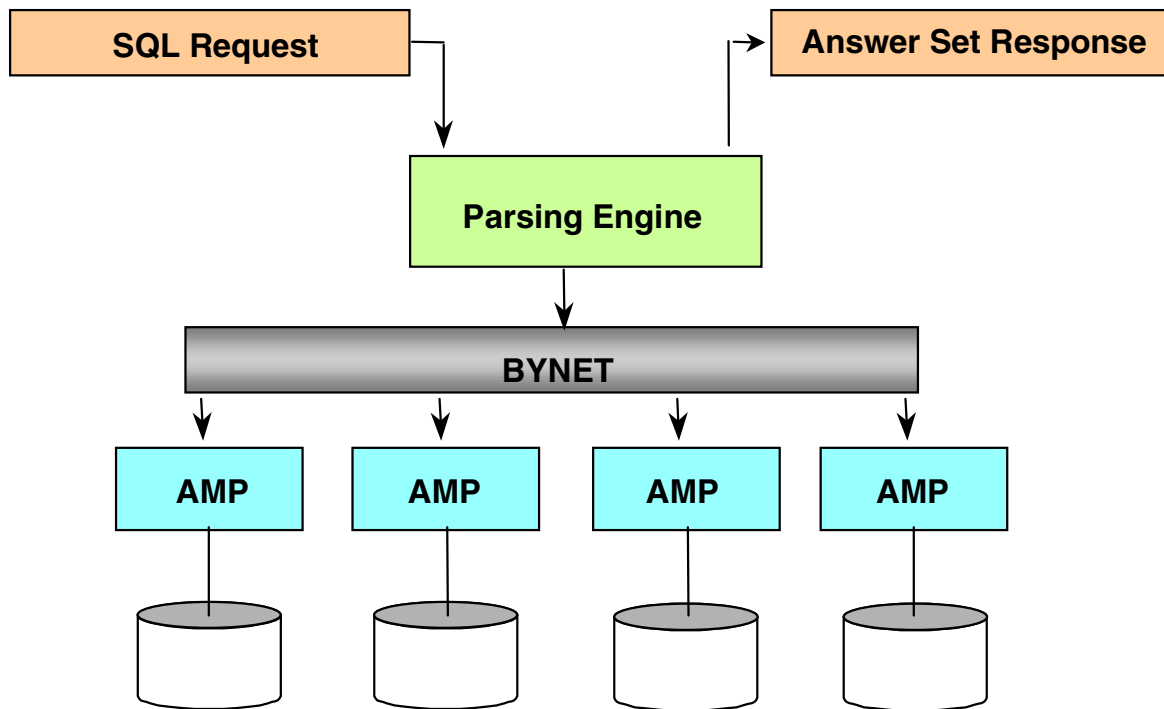
Depending on the nature of the dispatch request, the communication may be a:

- **Broadcast**—message is routed to all AMPs and PEs on the system.
- **Multipoint**—message is routed to specific AMPs on the system.
- **Point-to-point**—message is routed to one specific AMP or PE on the system.

The technology of the BYNET is what makes the parallelism of the Teradata RDBMS possible.

The BYNET is implemented for multi-node or single-node systems.

# BYNET



**The BYNET is responsible for:**

- Carrying messages between the AMPs and PEs.
- Broadcasting, point-to-point, or point-to-multipoint communications.
- Merging answer sets back to the PE.
- Making Teradata parallelism possible.

**The BYNET is implemented for multi-node or single-node systems.**

# The Access Module Processor (AMP)

*Topics reflected on Certification Exam.*

The **Access Module Processor (AMP)** is the **virtual processor (vproc)** responsible for **managing the database**. An individual AMP will control some portion of each table on the system. **AMPs do the physical work** associated with generating an answer set including sorting, aggregating, formatting, and converting. An AMP can control up to 64 physical disks.

An AMP responds to Parser/Optimizer steps transmitted across the BYNET by selecting data from or storing data to its disks. For some requests, the AMPs may redistribute a copy of the data to other AMPs.

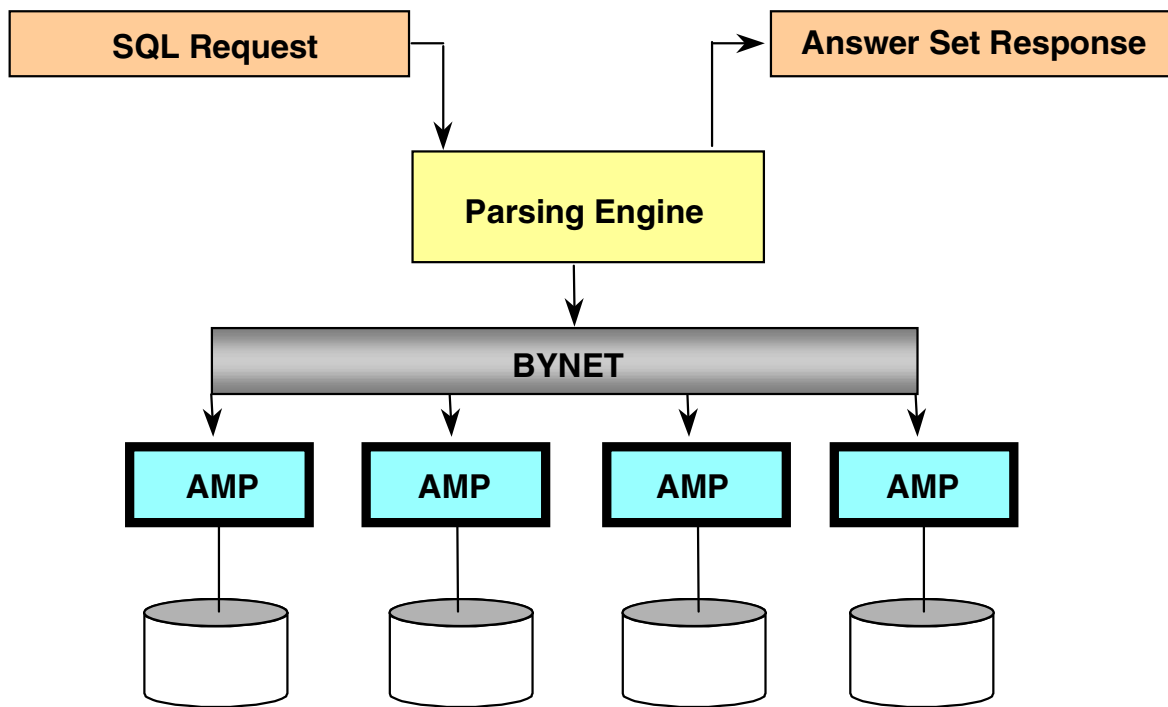
The **Database Manager** subsystem resides on each AMP. The Database Manager:

- Receives the steps from the Dispatcher and processes the steps. It has the ability to:
  - **Lock** databases and tables.
  - **Create, modify, or delete definitions** of tables.
  - **Insert, delete, or modify rows** within the tables.
  - **Retrieve information** from definitions and tables.
- Collects **accounting** statistics, recording accesses by session so users can be billed appropriately.
- Returns responses to the Dispatcher.

The **Database Manager** provides a bridge between that logical organization and the physical organization of the data on disks. The Database Manager performs a space-management function that controls the use and allocation of space.

AMPs also perform **output data conversion, checking the session** and **changing the internal**, 8-bit ASCII used by Teradata to the data format of the requester. (This is the reverse of the process performed by the PE when it converts the incoming data into internal ASCII.)

## The Access Module Processor (AMP)



**AMPs store and retrieve rows to and from the disks.  
An AMP can control up to 64 physical disks.**

**AMPs are responsible for:**

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

## Disk Arrays

A **disk array** is a configuration of disk drives that utilizes specialized controllers to manage and distribute data and parity across the disks while providing fast access and data integrity.

Each **AMP vproc** must have access to an array controller that in turn accesses the physical disks. AMP vprocs are associated with one or more ranks of data. The total disk space associated with an AMP vproc is called a **vdisk**. A vdisk may have up to three ranks.

Disks are organized as a **Redundant Array of Independent Disks (RAID)**. There are several levels:

- RAID Level 5—Data and parity protection striped across multiple disks.
- RAID Level 1—Each disk has a physical mirror replicating the data.
- RAID Level S—Data and parity protection similar to RAID 5 for EMC disk arrays.

The disk array controllers are referred to as **dual active array controllers**, which means that both controllers are actively used in addition to serving as backup for each other.

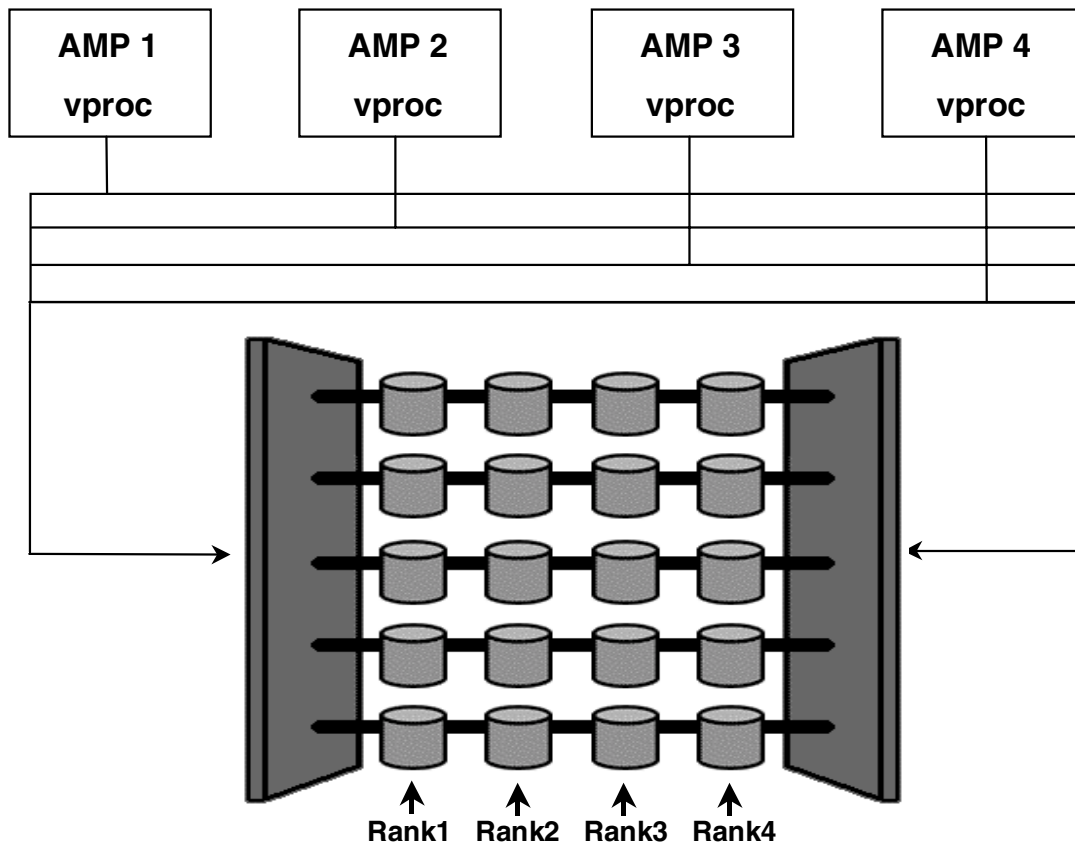
### Maximum amount of disk space per AMP

V2R2 - 46 GB

V2R3/V2R4 - 119 GB



## Disk Arrays



- Each AMP vproc is assigned to a *vdisk*.
- A vdisk may contain 119 GB of disk space.

## Disk Array RAID Protection

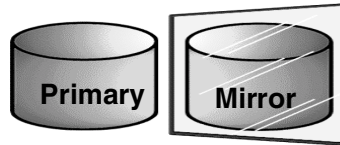
There are several forms of disk array protection in Teradata. RAID 1 and RAID 5 are commonly used and are discussed here. The disk array controllers manage both.

**RAID 1 is a disk-mirroring technique.** Each physical disk is mirrored elsewhere in the array. This requires the array controllers to write all data to two separate locations, which means data can be read from two locations. In the event of a disk failure, the mirror disk becomes the primary disk to the array controller and performance is unchanged. RAID 1 may be configured as RAID 1 + 0 that uses mirrored striping.

**RAID 5 is a parity-checking technique.** For every three blocks of data (spread over three disks), there is a fourth block on a fourth disk that contains parity information. This allows any one of the four blocks to be reconstructed by using the information on the other three. If two of the disks fail, the rank becomes unavailable. The array controller does the recalculation of the information for the missing block. Recalculation will have some impact on performance, but at a much lower cost in terms of disk space.

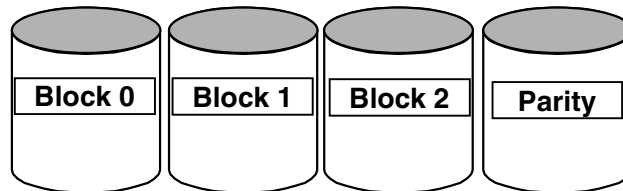
# Disk Array RAID Protection

## RAID-1 (Mirroring)



- Each physical disk in the array has an exact copy in the same array.
- The array controller can read from either disk and write to both.
- When one disk of the pair fails, there is no change in performance.
- Mirroring reduces available disk space by 50%.
- Array controller reconstructs failed disks quickly.

## RAID-5 (Parity)



- For every 3 blocks of data, there is a parity block on a 4th disk.
- If a disk fails, any missing block may be reconstructed using the other three disks.
- Parity reduces available disk space by 25% in a 4-disk rank.
- Array controller reconstruction of failed disks is longer than RAID 1.

## Summary

**RAID-1 - Good performance with disk failures.  
Higher cost in terms of disk space.**

**RAID-5 - Reduced performance with disk failures.  
Lower cost in terms of disk space.**

## Teradata Storage Process

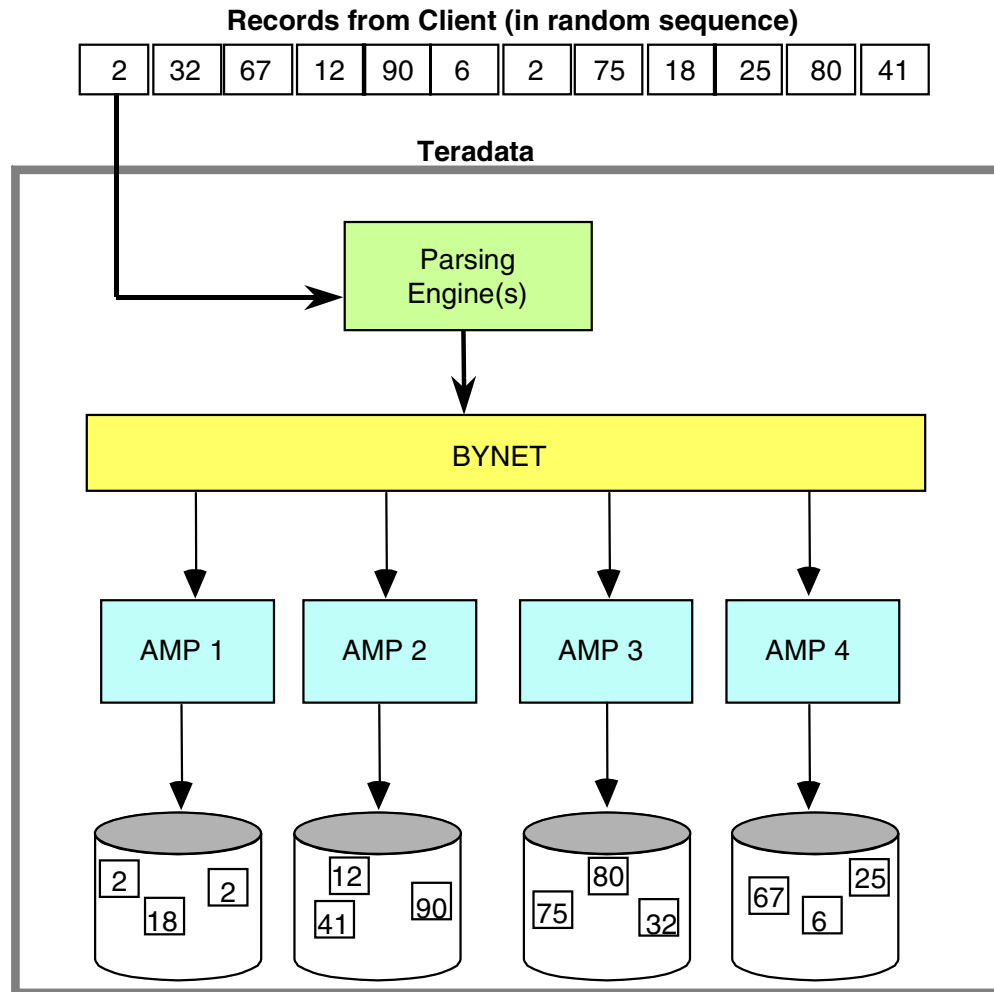
On the facing page, you can see a simplified view of how the physical components of a Teradata database work to insert a row of data.

- The **Parsing Engine** interprets the SQL command and converts the data record from the host into an AMP message.
- The **BYNET** distributes the row to the appropriate AMP.
- The **AMP** formats the row and writes it to its associated disks.
- The **disk** holds the row for subsequent access.

The **host** or **client system** supplies the records. These records are the raw data from which the database will be constructed.

Since Teradata has no concept of pre-allocated table space, not only are the rows of the tables distributed randomly across all of the AMPs, but they are stored randomly within the available disk space on that AMP.

# Teradata Storage Process



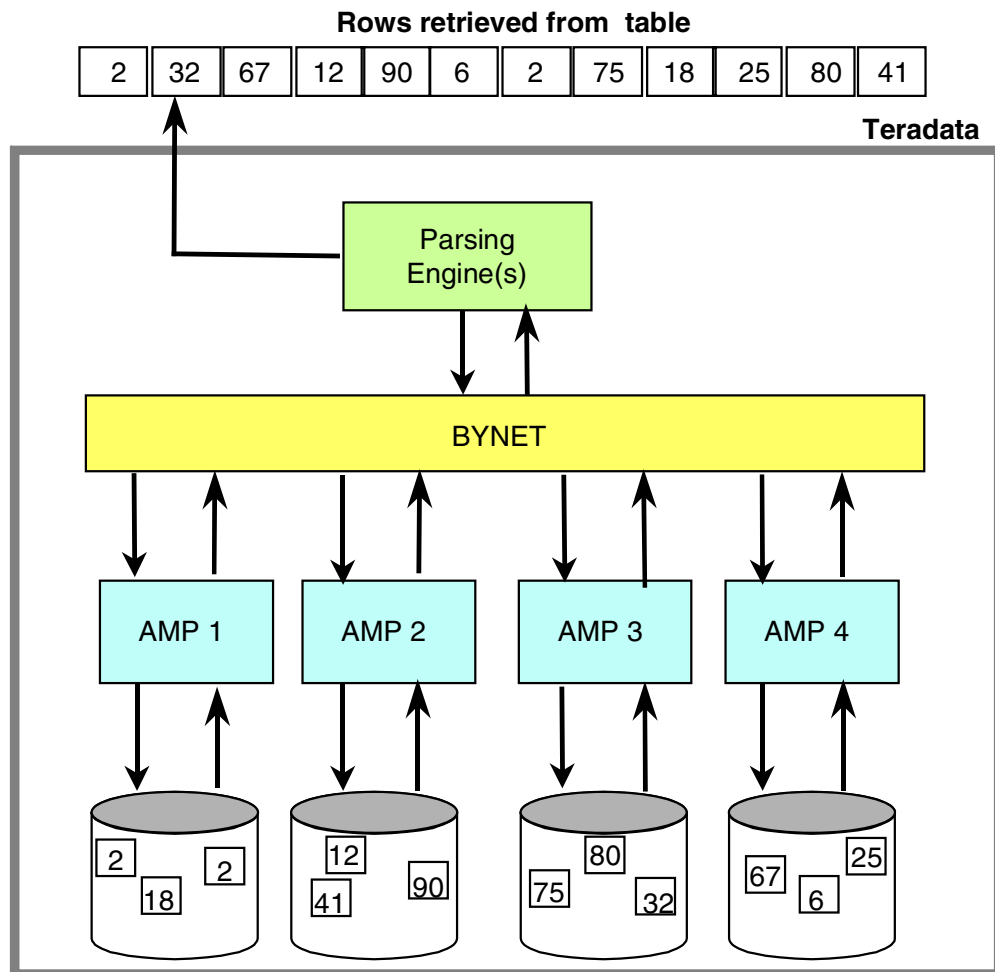
- The Parsing Engine dispatches a request to insert a row.
- The BYNET ensures that the row gets to the appropriate AMP (Access Module Processor).
- The AMP stores the row on its associated disk.
- Each AMP can have multiple physical disks associated with it.

## Teradata Retrieval Process

Retrieving data from the Teradata RDBMS simply reverses the storage model process. A request made for data is passed on to a Parsing Engine (PE). The PE optimizes the request for efficient processing and creates tasks for the AMPs to perform, which results in the request being satisfied. Tasks are then dispatched to the AMPs via the BYNET. Often, all AMPs must participate in creating the answer set, such as returning all rows of a table to a client application. Other times, only one or a few AMPs need participate. The PE will ensure that only needed AMPs will be assigned tasks.

Once the AMPs have been given their assignments, they retrieve the desired rows from their respective disks. The AMPs will sort, aggregate, or format if needed. The rows are then returned to the requesting PE via the BYNET. The PE takes the returned answer set and returns it to the requesting client application.

# Teradata Retrieval Process



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

# Multiple Tables on Multiple AMPs

*Topics reflected on Certification Exam.*

You might think that the RDBMS would assign each table to a particular AMP, and that the AMP would put that table on a single disk. However, as you see on the diagram on the facing page, that's not what happens. The system takes the rows that compose a table and divides them up among all available AMPs.

## How it works

- Tables are distributed across all AMPs. The distribution of rows should be even across all AMPs to ensure that the workload is being evenly distributed across the AMPs.
- Each table has some rows distributed to **each AMP**.
- Each AMP controls one **logical storage unit**, which may consist of several physical disks.
- Each AMP places, **maintains**, and **manages** the rows on its own disks.
- Large configurations may have **hundreds of AMPs**.
- **Full table scans**, operations, which require looking at all the rows of a table, access all AMPs *in parallel*. Parallelism is what makes possible the accessing of enormous amounts of data.

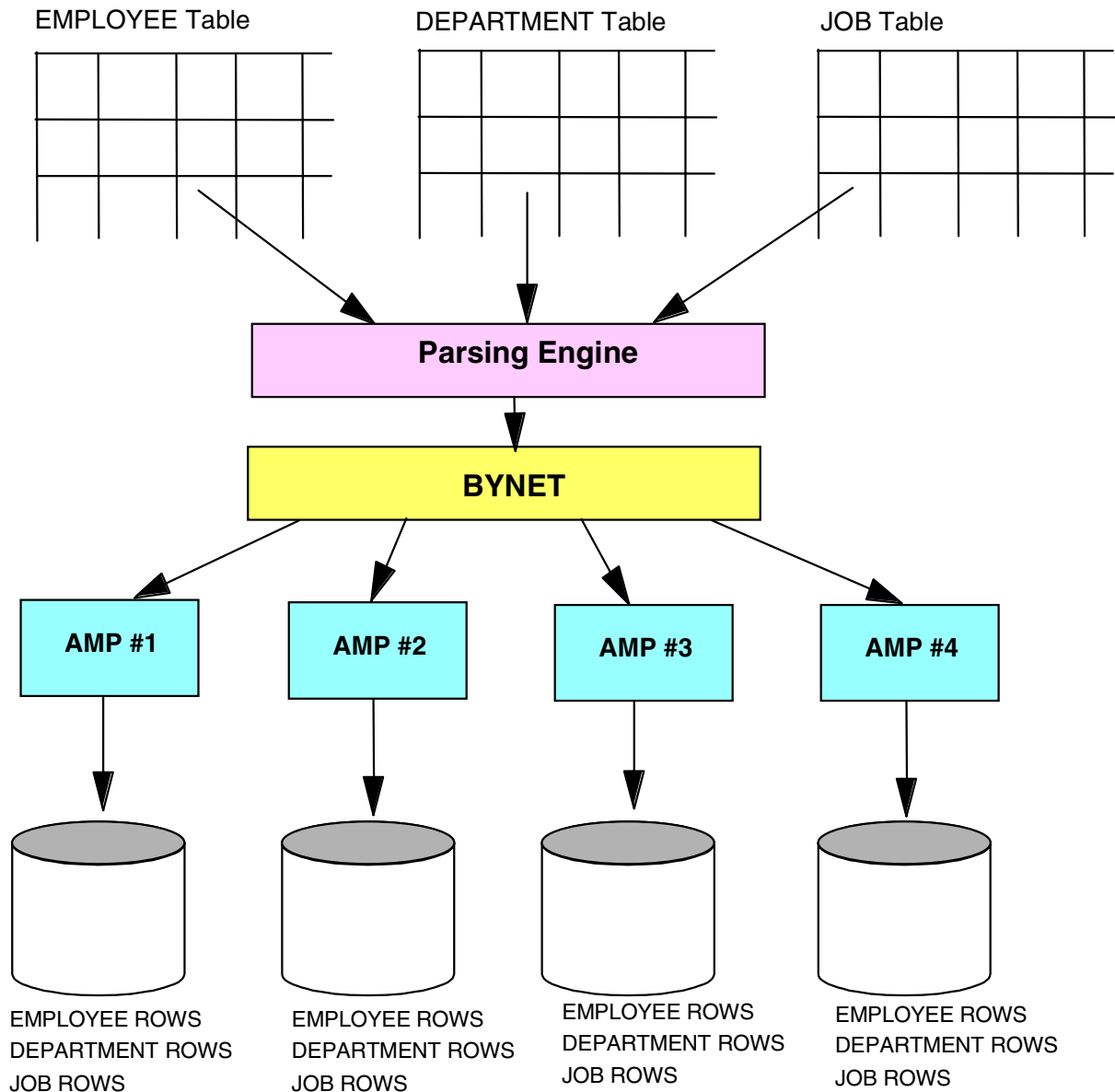
Consider the following three tables: EMPLOYEE, DEPARTMENT, and JOB.

The Teradata RDBMS takes the rows from each of the tables and divides them up among all the AMPs. The AMPs divide the rows up among their disks. **Each** AMP gets part of **each** table. Dividing up the tables means that all AMPs and their associated disks will be activated in a full-table scan, thus speeding up requests against these tables.

In our example, if you have four AMPs, theoretically each AMP would get approximately **25% of each table**. If AMP #1 were to get 90% of the rows from the EMPLOYEE table, that would be called lumpy data distribution. Lumpy data distribution slows the system down because any request that required scanning all the rows of EMPLOYEE would have three AMPs sitting idle while AMP #1 finished its work. It is better to divide all the tables up evenly among all the available AMPs. In a later chapter, you will see how this distribution is controlled.



## Multiple Tables on Multiple AMPs



- Some rows from each table may be found on each AMP.
- Each AMP may have rows from all tables.
- Ideally, each AMP will hold roughly the same amount of data.

## Linear Growth and Expandability

The Teradata RDBMS is the first commercial database system to offer true parallelism and the performance increase that goes with it. Think back to the example we just discussed of how rows are divided up among AMPs. Assume that our three tables, EMPLOYEE, DEPARTMENT, and JOB total 100,000 rows and 50 users.

What happens if you **double the number of AMPs** and the number of users stays the same? Performance **doubles**. Each AMP processes half as many rows as it used to.

Now think of that system in a situation where both the number of **users** and the number of AMPs are **doubled**. We now have 100 users, but we also have twice as many AMPs. What happens to performance? It **stays the same**. There is no drop-off in the speed with which requests are executed because the system is **modular** and the workload is easily partitioned into independent pieces. In the last example, each AMP still does the same amount of work as when there were half the AMPs and half the users.

This feature—that the amount of time (or money) required to do a task is directly proportional to the size of the system—is unique to the Teradata RDBMS. Traditional databases show a sharp drop in performance when the system approaches a critical size.

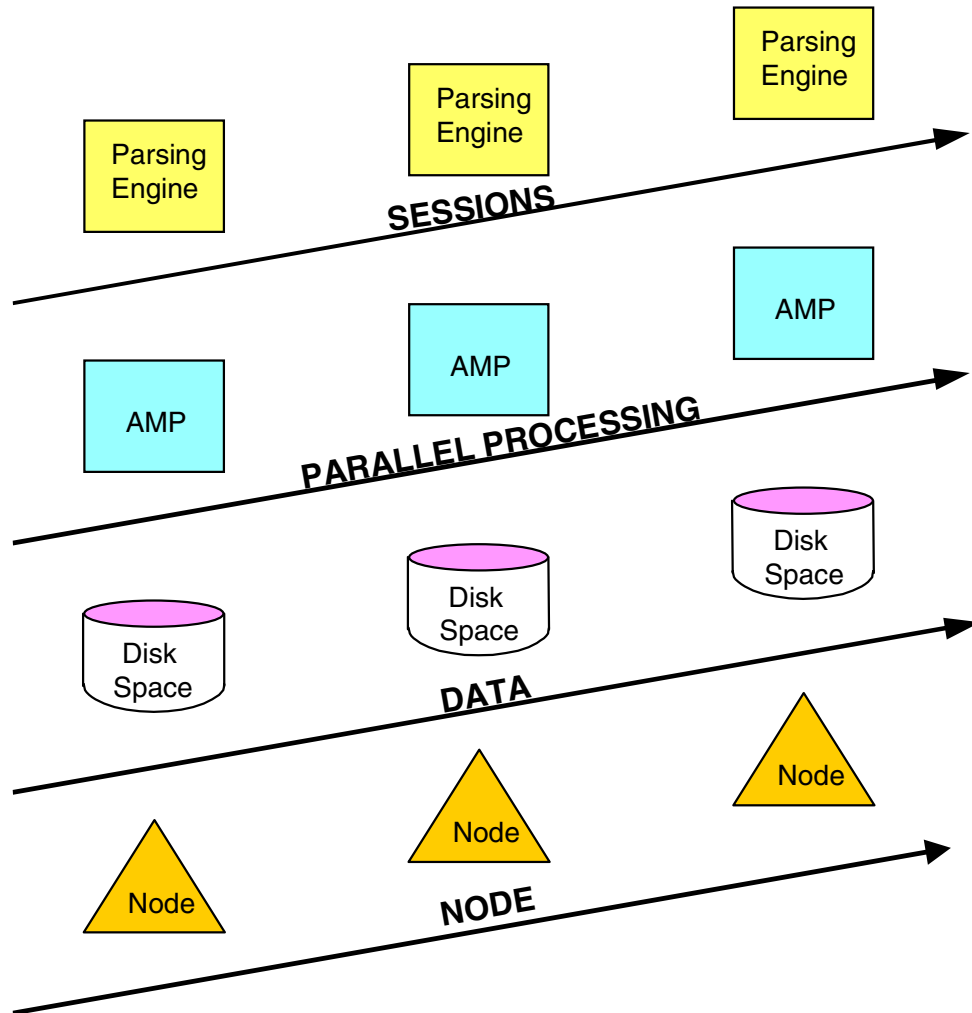
Look at the diagram on the facing page. As the number of **Parsing Engines** increases, the number of SQL requests that can be supported increases.

As you add **AMPs**, data is spread out more evenly as you add processing power to handle it.

As you add **disks**, you add space for each AMP to store and process more information. **All AMPs must have the same number of disks.**

**Note:** Performance assumptions on this page are approximations, and assume appropriate hardware increases as well as added vproc software.

## Linear Growth and Expandability



**Teradata is a *linearly expandable* RDBMS.**

Components may be added as requirements grow.

# Teradata Parallelism

*Topics reflected on Certification Exam.*

**Parallelism** is at the very heart of the Teradata RDBMS. Virtually every part of the system has parallelism built in. Without parallelism, managing enormous amounts of data would either not be possible or would be prohibitively expensive and inefficient.

Each PE can support up to 120 user sessions in parallel. This could be 120 distinct users or a single user harnessing the power of all 120 sessions for a one application.

Each session may handle multiple requests concurrently. While only one request at a time may be active on behalf of a session, the session itself can manage the activities of 16 requests and their associated answer sets.

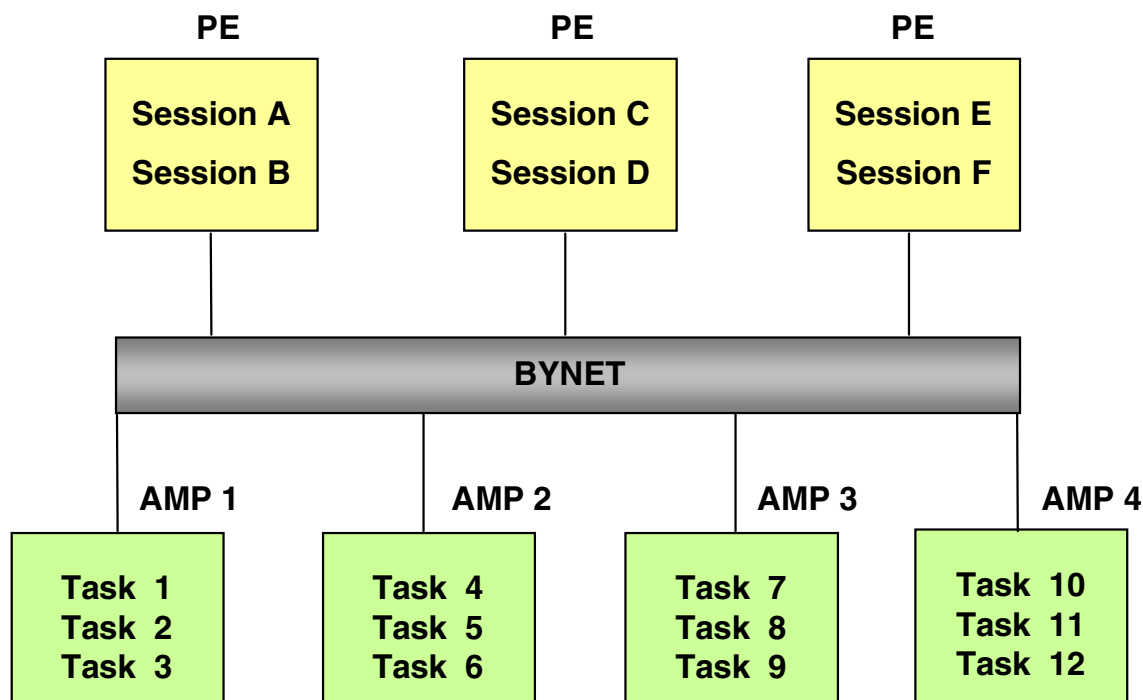
The BYNET is designed so that it can never be a bottleneck for the system. Because the BYNET is implemented differently for different platforms, it will always be within the needed bandwidth for each particular platform's maximum throughput.

Each AMP can perform up to 80 tasks in parallel. AMPs are not dedicated at any moment in time to the servicing of only one request, but concurrently multi-thread multiple requests.

Because AMPs are designed to operate on only one portion of the database, they must operate in parallel to accomplish their intended results. In addition, the Optimizer may direct the AMPs to perform certain steps in parallel if there are no contingencies between the steps. This means that an AMP might be concurrently performing more than one step on behalf of the same request.

Parallel CLI enables parallel processing of the client application, which is particularly useful for multi-session applications and is accomplished by setting a few environmental variables. It requires no changes to the application code.

## Teradata Parallelism



- Each PE can handle up to 120 sessions in parallel.
- Each session can handle multiple requests.
- The BYNET can handle all message activity in parallel.
- Each AMP can perform up to 80 tasks in parallel.
- All AMPs can work together in parallel to service any request.
- Each AMP can work on several requests in parallel.

**PARALLELISM IS BUILT INTO THE TERADATA  
RDBMS FROM THE GROUND UP!**

# Teradata Functional Overview

The Teradata database requires three distinct pieces of software: TPA, PDE, and OS.

An OS (UNIX or Windows NT) and a Teradata software license are necessary for each node.

A Trusted Parallel Application (TPA) implements virtual processors and runs on the OS with PDE. The Teradata RDBMS is classified as a TPA. The components of the Teradata RDBMS include:

- Channel Driver
- Teradata Gateway
- AMP
- PE

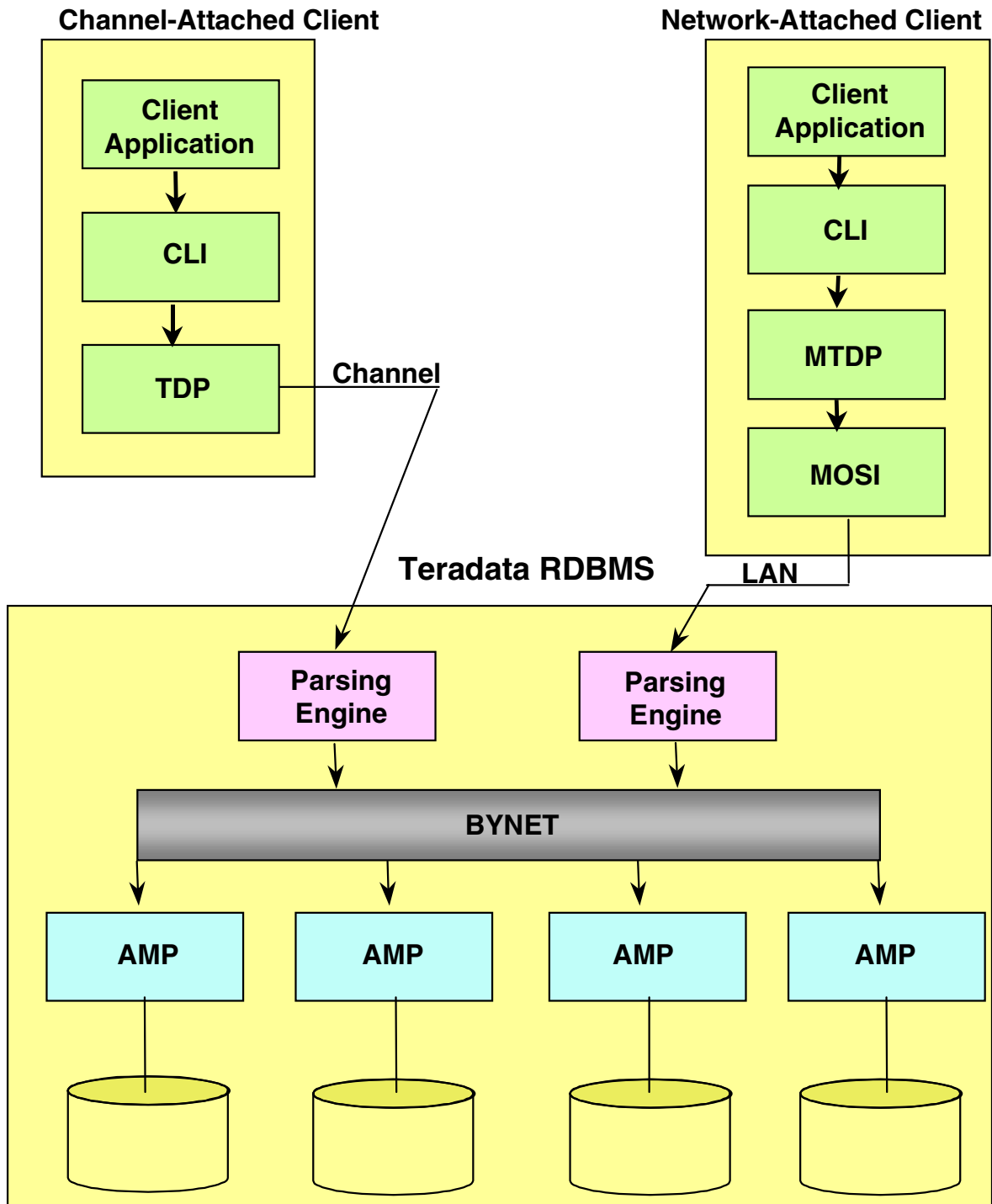
The Parallel Database Extensions (PDE) software is an interface layer on top of the operating system. PDE supports the parallel software environment.

The client may be:

- A mainframe system, such as IBM or Unisys that is channel-attached to the Teradata RDBMS.
- A PC or UNIX-based system that is LAN or network-attached.

The client application submits an SQL request to the RDBMS, receives the response, and submits the response to the user.

# Teradata Functional Overview



# Channel-Attached Client Software Overview

*Topics reflected on Certification Exam.*

In **channel-attached systems**, there are three major software components that play important roles in getting the requests to and from the Teradata RDBMS.

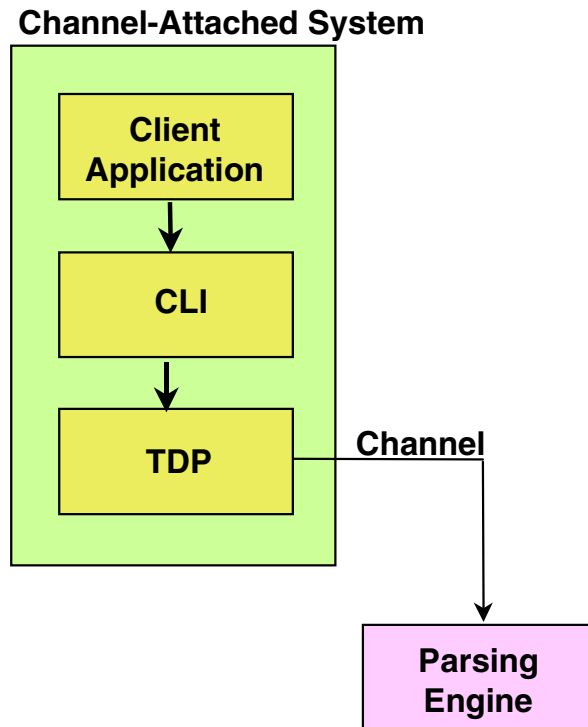
A programmer either writes the client application or it is one of Teradata's provided utility programs. Many client applications are written as front ends for SQL submission, but they also are written for file maintenance and report generation. Any client-supported language may be used provided it can interface to the Call Level Interface.

The **Call Level Interface (CLI)** is the lowest-level interface to the Teradata RDBMS. It consists of system calls that create sessions, allocate request and response buffers, package queries into uniform blocks, unpackage responses into a results table.

The **Teradata Director Program (TDP)** is a Teradata-supplied program that must run on any client system that will be channel-attached to the Teradata RDBMS. The TDP manages session traffic between the Call-Level Interface and the RDBMS. Its functions include session initiation and termination, logging, verification, recovery, restart, physical input to and output from the PEs (including session balancing), and the maintenance of queues. The TDP may also handle system security.



## Channel-Attached Client Software Overview



### Client Application

- Your own application(s)
- Teradata utilities (BTEQ, etc.)

### CLI (Call-Level Interface) Service Routines

- Request and response control
- Package SQL requests and unpackage results
- Buffer allocation and initialization

### TDP (Teradata Director Program)

- Session balancing across multiple PEs
- Ensures proper message routing to/from RDBMS
- Failure notification (application failure, Teradata restart)

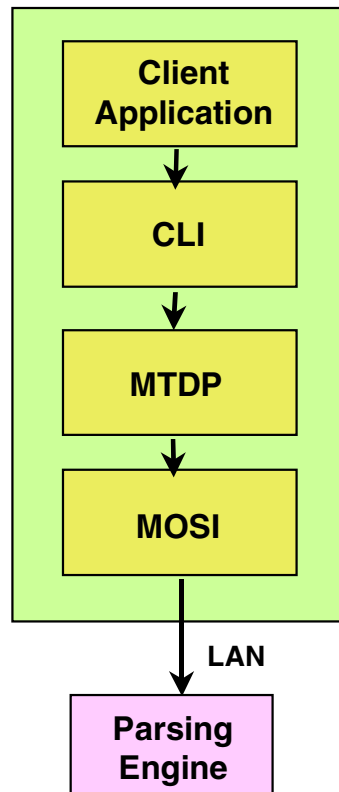
# Network-Attached Client Software Overview

In **network-attached systems**, there are four major software components that play important roles in getting requests to and from the Teradata RDBMS. They include:

- The **programmer using a client-supported language writes the client application**. The application's purpose is to submit SQL statements to the RDBMS and perform processing on the result sets.
- The **Call Level Interface (CLI)** is **a library of routines that resides on the client**. Client application programs use these routines to perform operations such as logging on and off, submitting SQL queries, and receiving responses which contain the answer set. These routines are nearly the same in both network-attached and channel-attached environments.
- The **Micro Teradata Director Program (MTDP)** is a Teradata-supplied program that must be linked to applications that will be network-attached to the Teradata RDBMS. **The MTDP performs many of the same functions as the channel-based TDP including session management**. The MTDP does not control session balancing across PEs. Connect and Assign servers that run on the Teradata system handle session balancing.
- The **Micro Operating System Interface (MOSI)** is a library of routines that provide operating system independence for clients accessing the RDBMS. By using MOSI, only one version of the MTDP is needed to run all network-attached platforms.

# Network-Attached Client Software Overview

## Network-Attached System



### CLI (Call Level Interface)

- Package SQL requests and unpackage results

### MTDP (Micro Teradata Director Program)

- Library of session management routines
- Data communications manager for network clients

### MOSI (Micro Operating System Interface)

- Library of routines providing OS-independent interface

# Teradata Objects

*Topics reflected on Certification Exam.*

A **database** in Teradata database systems is a collection of objects known as **tables, views, and macros**.

Databases provide a **logical grouping for information**. They are also the foundation for space allocation and access control.

## Tables

A table is the logical structure of data in an RDBMS. It is a **two-dimensional structure made up of columns and rows**. A user defines a table by giving it a table name that refers to the type of data that will be stored in the table.

A **column** represents attributes of the table. Column names are given to each column of the table. All information in a column is the same type. For example, a column named date of birth would only hold date of birth information.

Each occurrence of an entity is stored in the table as a **row**. Entities are the people, things, or events that the table is describing.

## Views

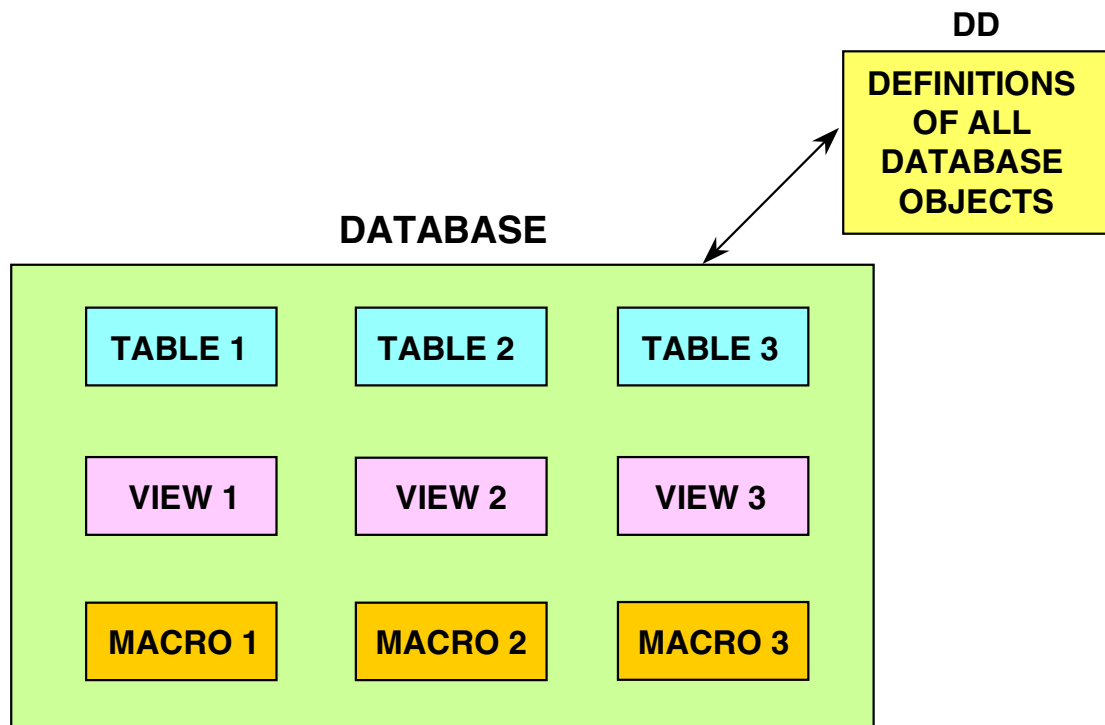
A view is a **pre-defined subset of one or more tables or other views**. It does not exist as a real table, but serves as a reference to existing tables or views. One way to think of a view is as **a virtual table**. Views have definitions in the data dictionary but do not contain any physical rows. Views can be used by the database administrator to control access to the underlying tables. Views can be used to hide columns from users, to insulate applications from database changes, and to simplify or standardize access techniques.

## Macros

A macro is a predefined, stored set of one or more SQL commands, and optionally, report-formatting commands. Macros are used to simplify the execution of **frequently used SQL commands**.

## Teradata Objects

- Three fundamental objects may be found in a Teradata database:
  - Tables—rows and columns of data
  - Views—predefined subsets of existing tables
  - Macros—predefined, stored SQL statements
- These objects are created, maintained, and deleted using Structured Query Language (SQL).
- Object definitions are stored in the Data Dictionary (DD).



# The Data Dictionary (DD)

*Topics reflected on Certification Exam.*

The Data Dictionary (DD) is an integrated set of system tables that:

- Stores database **object definitions** and accumulates **information** about **users, databases, resource usage, data demographics, and security rules.**
- **Records specifications** about **tables, views, and macros.**
- **Contains information** about **ownership, space allocation, accounting, and access rights (privileges)** for these objects.

Data Dictionary information is updated automatically during the processing of Teradata SQL data **definition language (DDL) statements**. It is used by the Parser to obtain information needed to process all Teradata SQL statements.

Users may access the DD through Teradata-supplied views, if permitted by the system administrator.

# The Data Dictionary (DD)

## The Data Dictionary (DD)

- Is an integrated set of system tables.
- Contains definitions of and information about all objects in the system.
- Is entirely maintained by the RDBMS.
- Is “data about the data” or “metadata.”
- Is distributed across all AMPs.
- May be queried by administrators or support staff.
- Is **accessed via Teradata-supplied views.**

## Examples of views:

- DBC.Tables—information about all tables
- DBC.Users—information about all users
- DBC.AllRights—information about access rights
- DBC.DiskSpace—information about space utilization

# Structured Query Language (SQL)

*Topics reflected on Certification Exam.*

Structured Query Language (SQL) is the language of relational databases. It is sometimes referred to as a Fourth Generation Language (4GL) to differentiate it from Third Generation Languages such as FORTRAN and COBOL, though it is quite different from other 4GLs. It acts as an intermediary between the user and the database.

SQL is different from other computer languages. Its statements resemble English-like structures. It provides powerful, set-oriented database manipulation including **structural modification, data retrieval, modification, and security functions.**

SQL is a **non-procedural language**. Because of its set orientation, it does not require IF, GOTO, DO, FOR NEXT, OPEN, or PERFORM statements.

We will describe three important subsets of SQL: the Data Definition Language, the Data Manipulation Language, and the Data Control Language.

## Data Definition Language (DDL)

The DDL allows a user to define the **database objects and the relationships that exist among them**. Examples of DDL uses are creating or modifying tables and views.

## Data Manipulation Language (DML)

The DML consists of the **statements that manipulate, change, or retrieve the data rows** of the database. If the DDL defines the database, the DML lets the user change the information contained in the database. The DML is the most commonly used subset of SQL. It is used to select, update, delete, and insert rows.

## Data Control Language (DCL)

The Data Control Language is used to **restrict or permit a user's access**. It can selectively limit a user's ability to retrieve, add, or modify data. It is used to grant and revoke access privileges on tables and views.

## Teradata Extensions for User Assistance

Teradata User Assistance commands are extensions that allow you to list the objects in a database or the characteristics of a table, see how a query will execute, or show the details of your system. The types of commands vary widely from vendor to vendor.



# Structured Query Language (SQL)

**SQL is a query language for Relational Database Systems.**

- **A fourth-generation language**
- **A non-procedural language**
- **A set-oriented language**

**SQL consists of:**

- **Data Definition Language (DDL)**
  - **Defines database structures (tables, users, views, macros)**  
**CREATE**  
**DROP**  
**ALTER**
- **Data Manipulation Language (DML)**
  - **Manipulates rows and data values**  
**INSERT**  
**SELECT**  
**UPDATE**  
**DELETE**
- **Data Control Language (DCL)**
  - **Grants and revokes access rights**  
**GRANT**  
**REVOKE**
- **Teradata SQL includes Teradata Extensions to SQL that include User Assistance commands:**  
**HELP**  
**SHOW**  
**EXPLAIN**  
**CREATE MACRO**

# Getting Information about Tables

*Topics reflected on Certification Exam.*

To perform a query, you need to know something about the table structure. Two SQL commands that can help are:

- **HELP TABLE**—displays information about database objects (users/databases, tables, views, and macros) and session characteristics used to create a table.
- **SHOW TABLE**—displays the data definition language (DDL) associated with database objects (tables, views, or macros) used to create a table.

**Note:** There are other useful HELP commands. For example, HELP SESSION provide information about the current session.

For more information on HELP commands, refer to *Teradata RDBMS SQL, Vol.4, Data Definition Statements*.

## Getting Information about Tables

**HELP TABLE customer\_service.employee;**

<u>ColumnName</u>	<u>Type</u>	<u>Comment</u>
employee_number	I	System assigned identification
manager_employee_number	I	
department_number	I	Department employee works in
job_code	I	Job classification designation
last_name	CF	Employee surname
first_name	CV	Employee given name
hire_date	DA	Date employee was hired
birthdate	DA	
salary_amount	D	Annual compensation amount

**SHOW TABLE employee;**

```
CREATE SET TABLE CUSTOMER_SERVICE.employee
,FALLBACK ,
  NO BEFORE JOURNAL,
  NO AFTER JOURNAL
(
  employee_number INTEGER,
  manager_employee_number INTEGER,
  department_number INTEGER,
  job_code INTEGER,
  last_name CHAR(20) NOT CASESPECIFIC NOT NULL,
  first_name VARCHAR(30) NOT CASESPECIFIC NOT NULL,
  hire_date DATE NOT NULL,
  birthdate DATE NOT NULL,
  salary_amount DECIMAL(10,2) NOT NULL)
UNIQUE PRIMARY INDEX ( employee_number );
```

# The SELECT Statement

**SELECT** is the most commonly used SQL statement. It is a data manipulation statement that:

- **Retrieves information** from the database
- **Organizes information for presentation** to the user or to the application program.

The SELECT statement allows you to retrieve data from tables **for display or processing**.

The SELECT statement on the facing page is an example of selecting data from a table. It retrieves the answer to the question: "Who was hired on 10/15/86?"

This is an example of a SELECT on one table. What if you want information from more than one table? That's our next topic.

# The SELECT Statement

Use the SELECT statement to retrieve data from tables.

Who was hired on October 15, 1986?

EMPLOYEE

EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT 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	Kanieski	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	432101	Villegas	Arnando	870102	470131	5970000
1003	0801	401	411100	Trader	James	860731	570619	4785000

```
SELECT Last_Name
       ,First_Name
FROM   Employee
WHERE  Hire_Date = 861015
;
```

<u>LAST NAME</u>	<u>FIRST NAME</u>
Stein	John
Ryan	Loretta
Johnson	Darlene

← Answer

# The Join Operation

A **Join** operation joins rows of multiple tables and creates virtual rows. These are rows that contain data from more than one table but are not maintained anywhere in permanent storage. Rows are matched up based on Primary and Foreign Key relationships.

**Joins** are an essential technique for accessing data in multiple tables.

Let's look at the sample tables on the facing page and see how you can use the **Join** operation to answer the question: "Who works in Research and Development?"

Example of a JOIN statement:

```
SELECT    first_name, last_name
FROM      employee INNER JOIN department
ON        employee.department_number =
          department.department_number
AND       department.department_name = 'research and
          development';
```

**Note:** Join operations are described in detail in the Teradata SQL course.

# The Join Operation

A join operation is used when the SQL query requires information from multiple tables.

Who works in Research and Development?

EMPLOYEE

EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT 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	Kanieski	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	432101	Villegas	Arnando	870102	470131	5970000
1003	0801	401	411100	Trader	James	860731	570619	4785000

DEPARTMENT

DEPARTMENT NUMBER	DEPARTMENT NAME	BUDGET AMOUNT	MANAGER EMPLOYEE NUMBER
PK			FK
501	marketing sales	80050000	1017
301	research & development	46560000	1019
302	product planning	22600000	1016
403	education	93200000	1005
402	software support	30800000	1011
401	customer support	98230000	1003
201	technical operations	29380000	1025

<u>FIRST NAME</u>	<u>LAST NAME</u>
John	Stein
Carol	Kanieski

Answer

# Views

A **view** is a pre-defined subset of one or more tables. Think of a view as a window that accesses selected portions of a database. Authorized users may use views to **read data** specified in the view and/or to **update** data specified in the view.

Use views to **simplify query requests**, to **limit access to data**, and to **allow different users to** look at the same data from different perspectives.

Views can show parts of one table (single-table view), more than one table (multi-table view), or a combination of tables and other views.

To the user, views look just like tables since they have both rows and columns. However, the rows and columns of a view are not stored directly; they are derived from the rows and columns of tables whenever the view is referenced. Since a view has no data of its own, it takes up no storage space except for its definition.

## Single-table View

A **single-table view** takes specified columns and/or rows from one table and makes them available in a way that resembles a table. An example might be an employee table from which you select only certain columns for employees in a particular department number, e.g., department 403, and present them in a view.

Example of a CREATE VIEW statement:

```
CREATE VIEW emp_403 AS
    SELECT      employee_number AS EMP NO
               ,department_number AS DEPT NO
               ,last_name AS LAST NAME
               ,first_name AS FIRST NAME
               ,hire_date AS HIRE DATE
    FROM        employee
    WHERE       department_number = 403;
```



# Views

- Pre-defined subsets of existing tables
- Consist of specified columns and/or rows from the table(s)
- Act as a window into an underlying table
- Have no data of their own

## Single table view

- Subset of columns from one table
- Allows users to read and update a subset of the underlying table

Employee (TABLE)

EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT 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	Kanieski	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	432101	Villegas	Arnando	870102	470131	5970000
1003	0801	401	411100	Trader	James	860731	570619	4785000

Emp\_403 (VIEW)

EMP NO	DEPT NO	LAST NAME	FIRST NAME	HIRE DATE
1005 801	403 403	Villegas Ryan	Arnando Loretta	870102 861015

## Multi-table Views

A **multi-table view** combines data from more than one table into one pre-defined view. These views are called **join views** because more than one table is involved. An example might be a view that shows employees and the names of their departments, information that comes from two different tables.

**Note:** Unlike single-table views, **multi-table views are read only.** You cannot alter the data via the view.

One might wish to create a view containing the last name and department name for all employees where this information is stored in different tables.

Example of SQL to create a join view:

```
CREATE VIEW EmpDept AS
SELECT      last_name, department_name
FROM        employee INNER JOIN department
ON          employee.department_number =
            department.department_number ;
```

An example of reading via this view is:

```
SELECT      last_name, department_name
FROM        EmpDept;
```

## Multi-table Views

- A multi-table view allows users to access data from multiple tables as if it were in a single table.
- Multi-table views are also called join views.
- Join views are used for reading only, not updating.

Employee (TABLE)

EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT 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	Kanieski	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	432101	Villegas	Arnando	870102	470131	5970000
1003	0801	401	411100	Trader	James	860731	570619	4785000

Department (TABLE)

DEPARTMENT NUMBER	DEPARTMENT NAME	BUDGET AMOUNT	MANAGER EMPLOYEE NUMBER
PK			FK
501	marketing sales	80050000	1017
301	research & development	46560000	1019
302	product planning	22600000	1016
403	education	93200000	1005
402	software support	30800000	1011
401	customer support	98230000	1003
201	technical operations	29380000	1025

EmpDept (VIEW)

LAST NAME	DEPARTMENT NAME
Stein	research & development
Kanieski	research & development
Ryan	education
Johnson	customer support
Villegas	education
Trader	customer support

# Macros

*Topics reflected on Certification Exam.*

You can create a **macro** that defines a sequence of Teradata SQL statements (and, optionally, Teradata report-formatting statements). When you execute the macro, the statements execute as a single transaction. Macros reduce the number of keystrokes needed to perform a complex task. This saves you time, reduces the chance of errors, reduces the communication volume to Teradata, and allows efficiencies internal to Teradata.

# Macros

- A macro is a predefined set of SQL statements.
- You can create macros for frequently occurring queries or sets of operations.
- To work with macros, you need **CREATE, DROP, and EXECUTE privileges.**

## Features

- Are source code stored on the database.
- Can be modified and executed at will.
- Are re-optimized at execution time.
- Can be executed interactively or by batch applications.
- Are executed by one EXECUTE command.
- Can accept user-provided parameter values.

## Benefits

- **Simplify and control access to the system.**
- **Enhance system security.**
- Reduce LAN/Channel traffic because they reduce the size of the query transmitted from the client application.
- Available to all connected hosts because they are stored in the Teradata DD.

To create macro:

```
CREATE MACRO customer_list AS  
(SELECT customer_name FROM customer);
```

To execute macro:

```
EXEC customer_list;
```

### Customer Name

ABC Corp.  
First National Bank  
Zundell's Retail  
Food 4 Less

:  
:

## Macro Related Commands

*Topics reflected on Certification Exam.*

Use the **REPLACE MACRO** command to apply changes a macro or create a new macro.

Use the **DROP MACRO** command to remove a macro definition from the Data Dictionary,.

The example on the facing page shows you how to use the REPLACE MACRO command to include a minor sort on last name in the birthday\_list macro.

## Macro Related Commands

<b>REPLACE MACRO</b> macroname <b>AS</b> (. . . );	Apply changes to a macro or create a new macro.
<b>DROP MACRO</b> macroname;	Remove a macro definition from the DD.

```
CREATE MACRO birthday_list AS
      (SELECT      last_name
                  ,first_name
                  ,birthdate
      FROM          employee
      WHERE         department_number = 201
      ORDER BY     birthdate; ) ;
```

Change "CREATE" to "REPLACE" and include the minor sort.

```
→ REPLACE MACRO      birthday_list AS
      (SELECT      last_name
                  ,first_name
                  ,birthdate
      FROM          employee
      WHERE         department_number = 201
      ORDER BY     birthdate
→                  ,last_name; ) ;
```

# The EXPLAIN Facility

*Topics reflected on Certification Exam.*

The **EXPLAIN** facility allows you to preview how Teradata will execute a requested query. It returns a summary of the steps the Teradata RDBMS would perform to execute the request. **EXPLAIN** also discloses the strategy and access method to be used, how many rows will be involved, and its cost in minutes and seconds. Use **EXPLAIN** to evaluate a query performance and to develop an alternative processing strategy that may be more efficient. **EXPLAIN** works on any SQL request. **The request is fully parsed and optimized, but not run.** The complete plan is returned to the user in readable English statements.

**EXPLAIN** provides information about locking, sorting, row selection criteria, join strategy and conditions, access method, and parallel step processing.

**EXPLAIN** is useful for performance tuning, debugging, pre-validation of requests, and for technical training.

The following is an example of an **EXPLAIN** on a very simple query:

```
EXPLAIN SELECT last_name, department_number FROM employee;
```

**Explanation (full)**

- 
- 1) First, we lock a distinct CUSTOMER\_SERVICE."pseudo table" for read on a RowHash to prevent global deadlock for CUSTOMER\_SERVICE.employee.
  - 2) Next, we lock CUSTOMER\_SERVICE.employee for read.
  - 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.
  - 4) Finally, we send out an END TRANSACTION step to all AMPs involved in processing the request.
- > The contents of Spool 1 are sent back to the user as the result of statement 1. The total estimated time is 0 hours and 0.15 seconds.

**Note:** With V2R4, the size of the EXPLAIN text is unlimited.



## The 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 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

## Review: Teradata Features

The Teradata system is a high-performance database system that processes enormous quantities of detail data that are beyond the capability of conventional systems.

The system is **specifically designed for large relational databases**. From the beginning, the Teradata system was created to do one thing—manage enormous amounts of data.

**Hundreds of terabyte s of on-line storage capacity** are currently available, making it an ideal solution for enterprise data warehouses or even smaller data marts.

**Parallel processing** distributes data across multiple processors uniformly. The system is designed in such a way that the component parts divide the work up into approximately equal pieces. This keeps all the parts busy all the time and enables the system to accommodate a larger number of users and/or more data.

**Open architecture** adapts readily to new technology. As higher-performance industry standard computer chips and disk drives are made available, they are easily incorporated into the architecture.

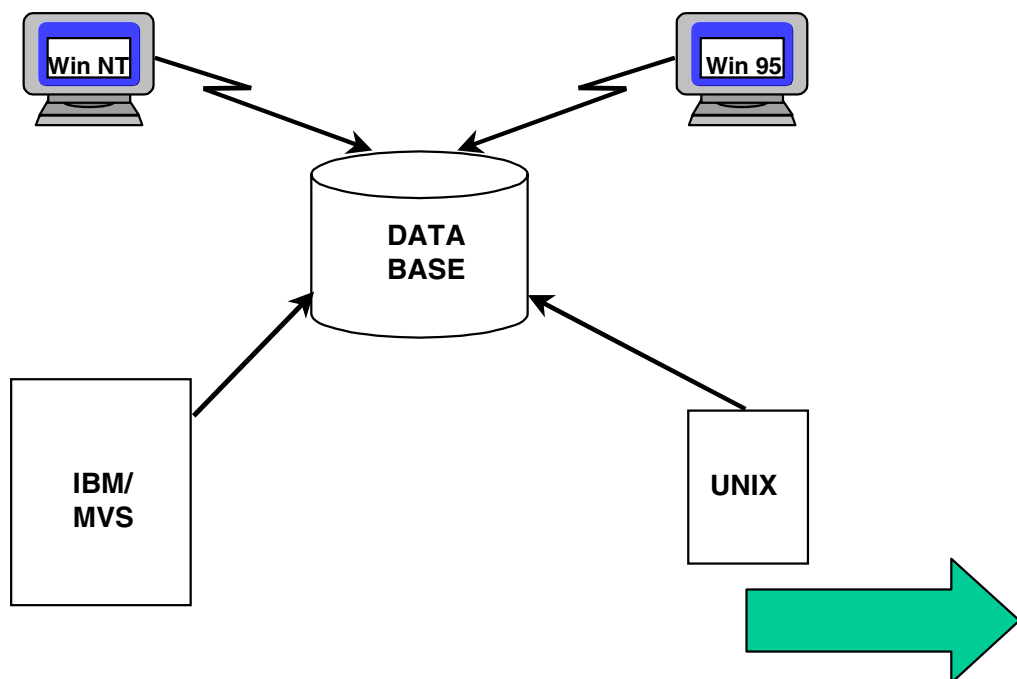
As the configuration grows, **performance increase is linear**. There is no performance drop-off until extremely large configurations are attempted.

**Structured Query Language (SQL)** is the industry standard for communicating with relational databases.

The Teradata RDBMS currently runs as a **database server on a variety of UNIX-based hardware platforms**, with Windows NT the most recent platform addition.

## Review: Teradata Features

- Designed to process large quantities of detail data
- Ideal for data warehouse and data mart applications
- Parallelism makes possible access to very large tables
- Open architecture—uses industry standard components
- Performance increase is linear as components are added
- Uses standard SQL
- Runs as a database server to client applications
- Runs on **multiple hardware platforms**



## **Review: Teradata Objects**

The facing page reviews key Teradata objects.

## **Review: Teradata Objects**

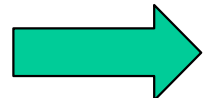
- **Tables—rows and columns of data**
- **Views—predefined subsets of existing tables**
- **Macros—predefined, stored SQL statements**
- **SQL—creates, maintains, and deletes objects**
- **Data Dictionary—stores object definitions**

## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.

## **Review Questions**

- 1. Name the three major Teradata components and their purposes?**
- 2. What are three Teradata database objects?**
- 3. What are views?**
- 4. What are macros? What privileges do you need to work with macros?**
- 5. How many sessions can a PE support?**
- 6. What happens to performance if you double the number of AMPs?**
- 7. Name the 2 types of virtual processors.**



## Review Questions—Continued



## Review Questions—Continued

### MATCH QUIZ 1

- |                 |       |   |
|-----------------|-------|---|
| 1. CLI          | _____ | a. Does locking, sorting, inserting, etc.                             |
| 2. MTDP         | _____ | b. Handles up to 120 sessions   |
| 3. MOSI         | _____ | c. Handles communication between AMPs and PEs                         |
| 4. PE           | _____ | d. Balances sessions across PEs for channel-attached systems          |
| 5. SQL          | _____ | e. Provides client-side OS independence                               |
| 6. AMP          | _____ | f. Manages data communication for network clients                     |
| 7. BYNET        | _____ | g. Plans most efficient steps to return response                      |
| 8. TDP          | _____ | h. At the client, packages queries into blocks and unpackages results |
| 9. Optimizer    | _____ | i. Library of Teradata Service Routing                                |
| 10. Dispatcher  | _____ | j. Foundation of Teradata architecture                                |
| 11. Parallelism | _____ | k. Language used to access data on tables                             |

### MATCH QUIZ 2

- |            |       |  |
|------------|-------|--|
| 1. HELP    | _____ | a. Displays the DDL for an object                  |
| 2. EXPLAIN | _____ | b. Displays information about objects              |
| 3. SHOW    | _____ | c. Displays path PE optimizer will use for a query |
| 4. DML     | _____ | d. Allows you to create an object                  |
| 5. DDL     | _____ | e. Allows you to retrieve a row (e.g. SELECT)      |

## Notes

### **Teradata and the Data Warehouse**

**After completing this module, you will be able to:**

- **Identify four types of enterprise data processing.**
- **Define a data warehouse and a data mart.**
- **List and define the different types of data marts.**
- **Explain the advantages of detail data over summary data.**

## Notes

*Attention Instructors! Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*

## Table of Contents

Evolution of Data Processing .....	4
Evolution in Data Capture and Storage .....	6
The Advantage of Using Detail Data .....	8
The Data Warehouse .....	10
Data Marts .....	12
Data Mart Pros and Cons .....	14
Teradata and the Data Warehouse .....	16
Review Questions.....	18

# Evolution of Data Processing

Traditionally, data processing has been divided into two categories: **on-line transaction processing (OLTP)** and **decision support systems (DSS)**. Let's look at the four types of enterprise data processing.

## On-line Transaction Processing (OLTP)

OLTP is typified by a small number of rows (or records) being accessed in a matter of seconds or less. This type of transaction takes place when we take out money at an ATM. Once our card is validated, a debit transaction takes place against our current balance to reflect the amount of cash withdrawn. We expect these transactions to be performed quickly. They must occur in real time.

## Decision Support Systems (DSS)

Decision support systems have always been with us, but were limited in earlier days. Batch reports, which roll-up numbers to give business the big picture were fairly typical of early DSS systems. Over time, this type of information requirement has evolved:

- Instead of pre-written scripts, users now require the ability to do ad hoc queries, which are unpredictable in their processing and which allow what if types of queries.
- DSS systems are now expected to read through huge volumes of data—volumes unthinkable even 10 or 15 years ago. DSS is an area where Teradata has traditionally excelled with its parallel architecture.

## On-line Complex Processing (OLCP)

OLCP is typified by a moderate number of rows processed against multiple databases with a response in minutes, not seconds. Filling out an application for instant credit in a store becomes an OLCP transaction. The information is used to do a quick check on your credit history, to approve or disapprove your application, and to determine an appropriate credit limit. People are usually more willing to wait ten or fifteen minutes to be approved for credit than they would be to withdraw their own money.

## On-line Analytical Processing (OLAP)

OLAP is the kind of processing which takes place in many data warehouses or data marts. Here the user may be looking for historical trends, sales rankings or seasonal inventory fluctuations for the entire corporation. Usually, this involves a great deal of detail data to be retrieved, processed and analyzed and, therefore response time can be in seconds or minutes. In the most sophisticated OLAP systems, systems themselves will make automated purchasing or inventory decisions without any human intervention.

## Evolution of Data Processing

<b>T R A D I T I O N A L</b>	Type	Example	Number of Rows Accessed	Response Time
	OLTP	Update a checking account to reflect a deposit.	Small	Seconds
	DSS	How many child size blue jeans were sold across all of our Eastern stores in the month of March?	Large	Seconds or minutes
<b>T O D A Y</b>	OLCP	Instant credit—How much credit can be extended to this person?	Small to moderate against multiple databases	Minutes
	OLAP	Show the top ten selling items across all stores for 1997.	Large of detail rows or moderate of summary rows	Seconds or minutes

# Evolution in Data Capture and Storage

There are four major media for capturing data—**paper, microfiche, magnetic tape, and direct access storage devices (DASD).**

There have been two strong trends in data storage over the history of computers.

The first is the **increase in the amount of data stored on DASD.**

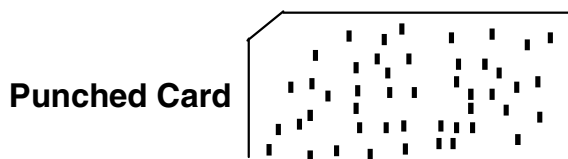
Electronic on-line storage has gotten cheaper, and probably will continue to do so. At the same time, businesses have recognized the value of detail data as opposed to summarized data. Databases, and the relational database in particular, provide a better way of storing data and making it more accessible, more useful, and, therefore, more valuable.

The second trend is the **increase in the amount of overall data** captured. The information explosion is real. Paper, fiche, and tape storage are simply not up to the task of making data available in a manner that can effect operation of the enterprise.



# Evolution in Data Capture and Storage

Early data capture was often slow and unreliable.

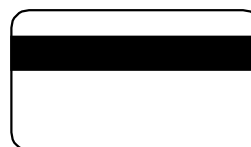


Today, more data is captured faster and more accurately.



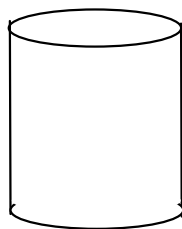
Bar Code

and

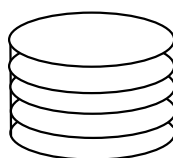


Magnetic Stripe

Disk technologies have become cheaper and more reliable.



1960's Drum - \$1,000s/MB



1980's Disk - \$26.00/MB



1990's Disk - < \$1.00/MB

## Conclusion

More data and more accurate data can now be stored.

## Problem

Turning large volumes of detail data into useful information.

# The Advantage of Using Detail Data

Until recently, most business decisions were based on summary data.

The problem is that **summarized data** is not as useful as **detail data** and cannot answer some questions with accuracy.

With summarized data, peaks and valleys are leveled when the peaks fall at the end of reporting period and are cut in half—as shown in the example on the facing page.

Here's another example. Think of your monthly bank statement that records checking account activity. If it only told you the total amount of deposits and withdrawals, would you be able to tell if a certain check had cleared? To answer that question you need a list of every check received by your bank. You need **detail data**.

Decision support—answering business questions—is the real purpose of database s. To answer business questions, decision-makers must have four things:

- The right data
- Enough detail data
- Proper data structure
- Enough computer power to access and produce reports on the data

Consider your own business and how it uses data. Is that data detailed or summarized? If it's summarized, are there questions it cannot answer?

# The Advantage of Using Detail Data

How effective was the national advertisement for jeans which ran June 6 through June 8?

## DETAIL vs. SUMMARY DATA

### Detail Data

STORE ITEM HISTORY

STORE NUMBER	ITEM NUMBER	DATE	QUANTITY SOLD
PK			
FK			
1	2	June01	110
1	2	June02	126
1	2	June03	127
1	2	June04	144
1	2	June05	102
1	2	June06	344
1	2	June07	410
1	2	June08	426
1	2	June09	297
1	2	June10	164
1	2	June11	167
1	2	June12	115
1	2	June13	108
1	2	June14	99
...	...	...	...
2	2	June01	50
2	2	June02	47
2	2	June03	32
2	2	June04	20
2	2	June05	37
2	2	June06	144
2	2	June07	126
2	2	June08	164
2	2	June09	96
2	2	June10	48
2	2	June11	34
2	2	June12	24
2	2	June13	30
2	2	June14	45
...	...	...	...
5	2	June01	13
5	2	June02	14
5	2	June03	10
5	2	June04	12
5	2	June05	15
5	2	June06	32
5	2	June07	65
5	2	June08	87
5	2	June09	18
5	2	June10	26
5	2	June11	8
5	2	June12	14
5	2	June13	22
5	2	June14	11
...	...	...	...

### Summary Data

STORE ITEM HISTORY WEEKLY SUMMARY

STORE NUMBER	ITEM NUMBER	WEEK ENDING	QUANTITY SOLD
PK			
FK			
...	...	...	...
1	2	June07	1363
1	2	June14	1376
...	...	...	...
2	2	June 07	456
2	2	June14	441
...	...	...	...
5	2	June07	161
5	2	June14	186
...	...	...	...

- Crucial data is lost.
- Business decisions suffer.
- Revenue drops.

- Gives accurate picture.
- Correct business decisions result.

# The Data Warehouse

A **data warehouse** is a central, enterprise-wide database that contains information extracted from the operational data stores. Data warehouses have become more common in corporations where enterprise-wide detail data may be used in on-line analytical processing to make strategic and tactical business decisions. Warehouses often carry many years worth of detail data so that historical trends may be analyzed using the full power of the data.

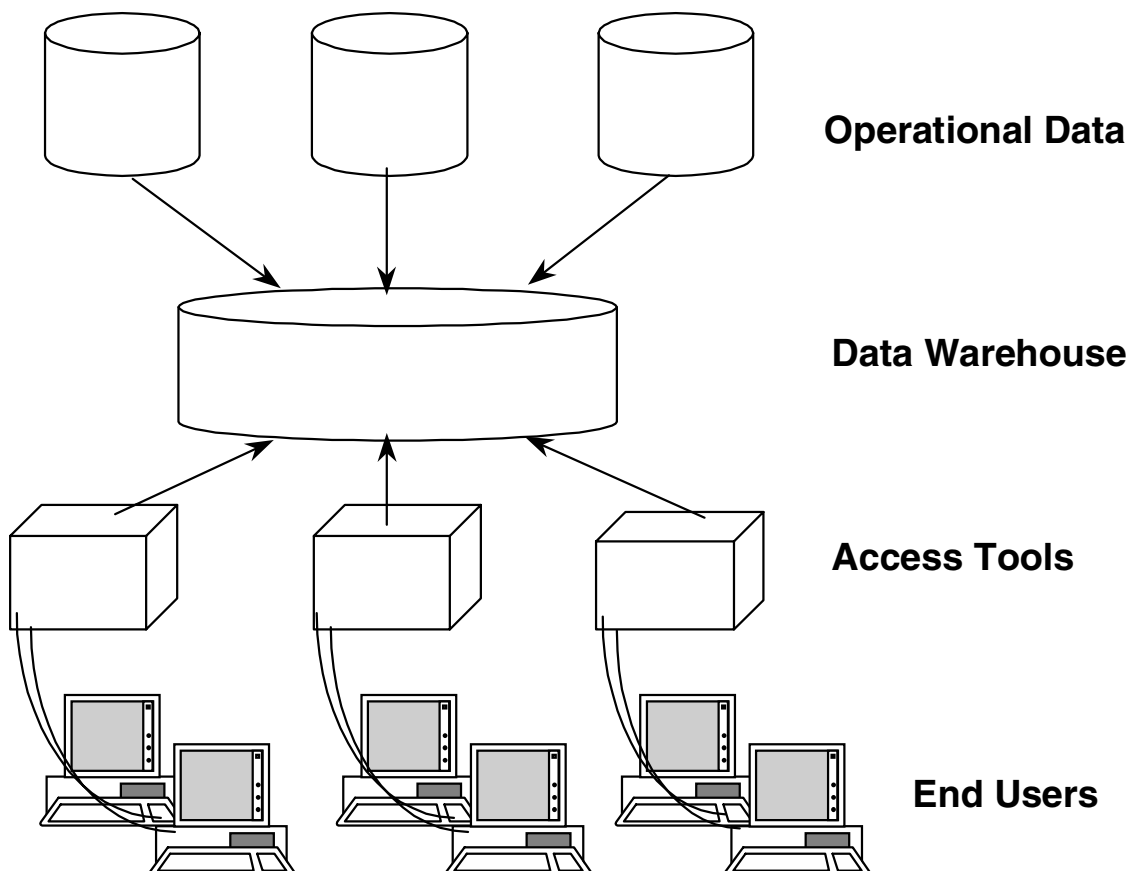
Many data warehouses get their data directly from operational systems so that the data is timely and accurate. While data warehouses may begin somewhat small in scope and purpose, they often grow quite large as their utility becomes more fully exploited by the enterprise.

# The Data Warehouse

A data warehouse is a central, enterprise-wide database which contains information extracted from the operational data stores.

## Characteristics of a data warehouse include:

- Based on enterprise-wide model
- Can begin small but may grow large rapidly
- Populated by extraction/loading of data from operational systems
- Maintain detail data
- Responds to end-user “what if” queries



## Data Marts

A **data mart** is a special-purpose subset of enterprise data used by a particular department, function, or application. Data marts may have both **summary and detail data**, however, usually the data has been pre-aggregated or transformed in some way to better handle the particular type of requests of a specific user community.

### Independent Data Marts

Independent data marts are created directly from operational systems, just as is a data warehouse. In the data mart, the data is usually transformed as part of the load process. Data might be aggregated, dimensionalized or summarized historically, as the requirements of the data mart dictate.

### Logical Data Marts

Logical data marts are **not separate physical structures** but rather are **an existing part of the data warehouse**. Because in theory the data warehouse contains the detail data of the entire enterprise, a logical view of the warehouse might provide the specific information for a given user community, much as a physical data mart would. Without the proper technology, a logical data mart can be a slow and frustrating experience for end users. With the proper technology, it removes the need for massive data loading and transforming, making a single data store available for all user needs.

### Dependent Data Marts

Dependent data marts are created from the detail data in the data warehouse. While having many of the advantages of the logical data mart, this approach still requires the movement and transformation of data but may provide a better vehicle for performance-critical user queries.

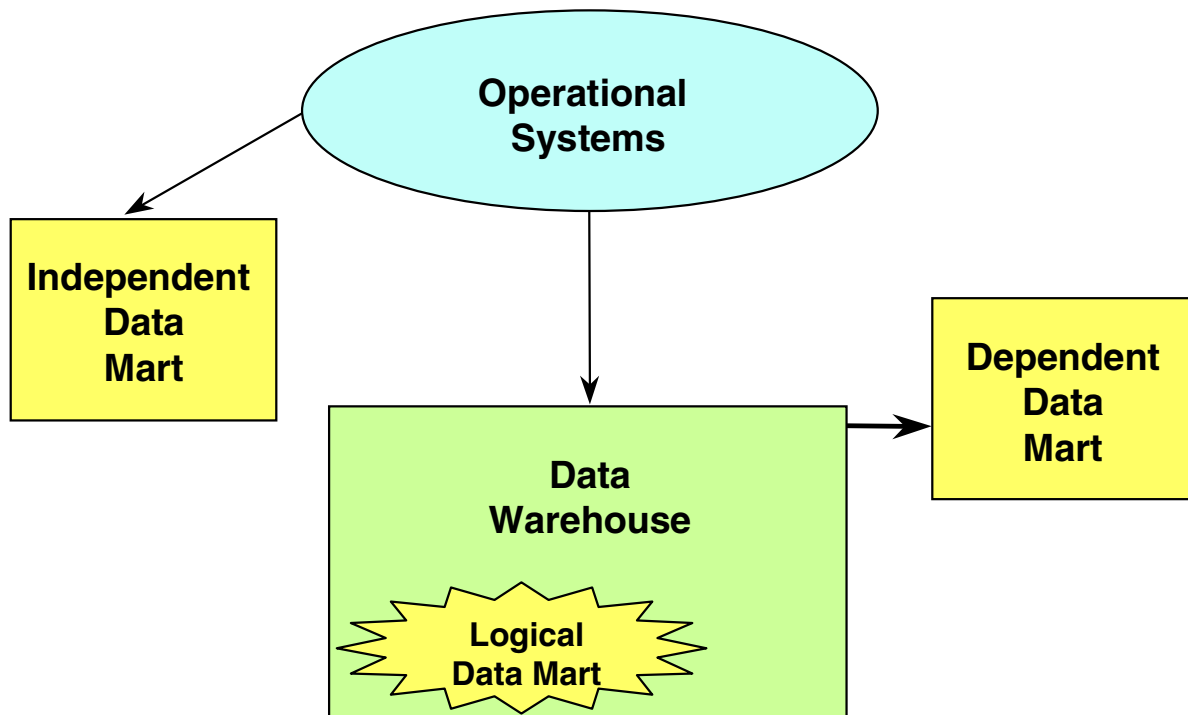
# Data Marts

## A data mart:

- A special-purpose subset of enterprise data
- Used for a particular function or application
- May contain detail or summary data or both

## Data mart types:

- Independent—created directly from operational systems to a separate physical data store.
- Logical—exists as a subset of existing data warehouse.
- Dependent—created from data warehouse to a separate physical data store.



# Data Mart Pros and Cons

*Topics reflected on Certification Exam.*

## Independent Data Marts

Independent data marts are usually the easiest and fastest to implement and their payback value can be almost immediate. Some corporations start with several data marts before deciding to build a true data warehouse. This approach has several inherent problems:

- While data marts have obvious value, they are not a true enterprise-wide solution and can become very costly over time as more and more are added.
- A major problem with proliferating data marts is that, depending on where you look for answers, there is often **more than one version of the truth.**
- They do not provide the historical depth of a true data warehouse.
- Because data marts are designed to handle specific types of queries from a specific type of user, they are often **not good at what if queries like a data warehouse would be.**

## Logical Data Marts

Logical data marts overcome most of the limitations of independent data marts. They provide a single version of the truth. There is no historical limit to the data and what if querying is entirely feasible. The major drawback to logical data marts is **the lack of physical control over the data.** Because data in the warehouse is not pre-aggregated or dimensionalized, performance against the logical mart will not usually be as good as against an independent mart. However, use of parallelism in the logical mart can overcome some of the limitations of the non-transformed data.

## Dependent Data Marts

Dependent data marts provide all advantages of a logical mart and also allow for physical control of the data as it is extracted from the data warehouse. Because dependent marts use the warehouse as their foundation, they are generally considered a better solution than independent marts, but they **take longer and are more expensive to implement.**



# Data Mart Pros and Cons

## Independent Data Mart

### Pro

Easy to implement  
Can eventually become a DW

### Con

Not an enterprise-wide solution  
Costly as more DMs are added  
More than one version of the truth  
Limits amount of historical data  
Data transformations needed  
Doesn't do "what if" queries

## Logical Data Mart

### Pro

Single version of the truth  
No historical data limits  
Allows drill downs, trend analysis  
No transformations needed

### Con

Less physical control over data

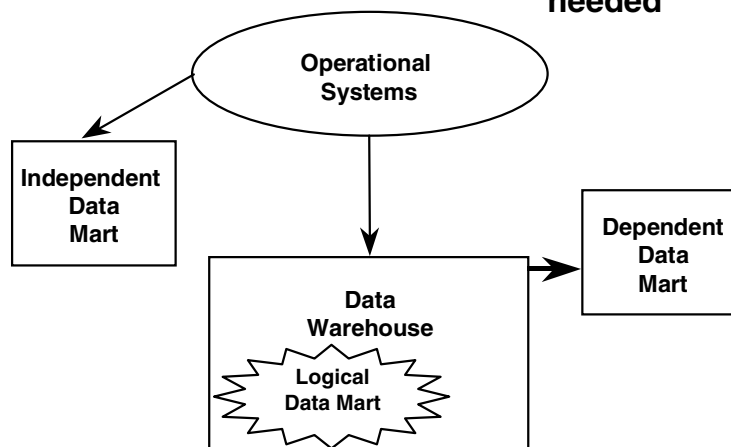
## Dependent Data Mart

### Pro

All advantages of Logical DM  
Allows for physical control over data

### Con

Additional movement of data is necessary  
Some transformation may be needed



## Teradata and the Data Warehouse

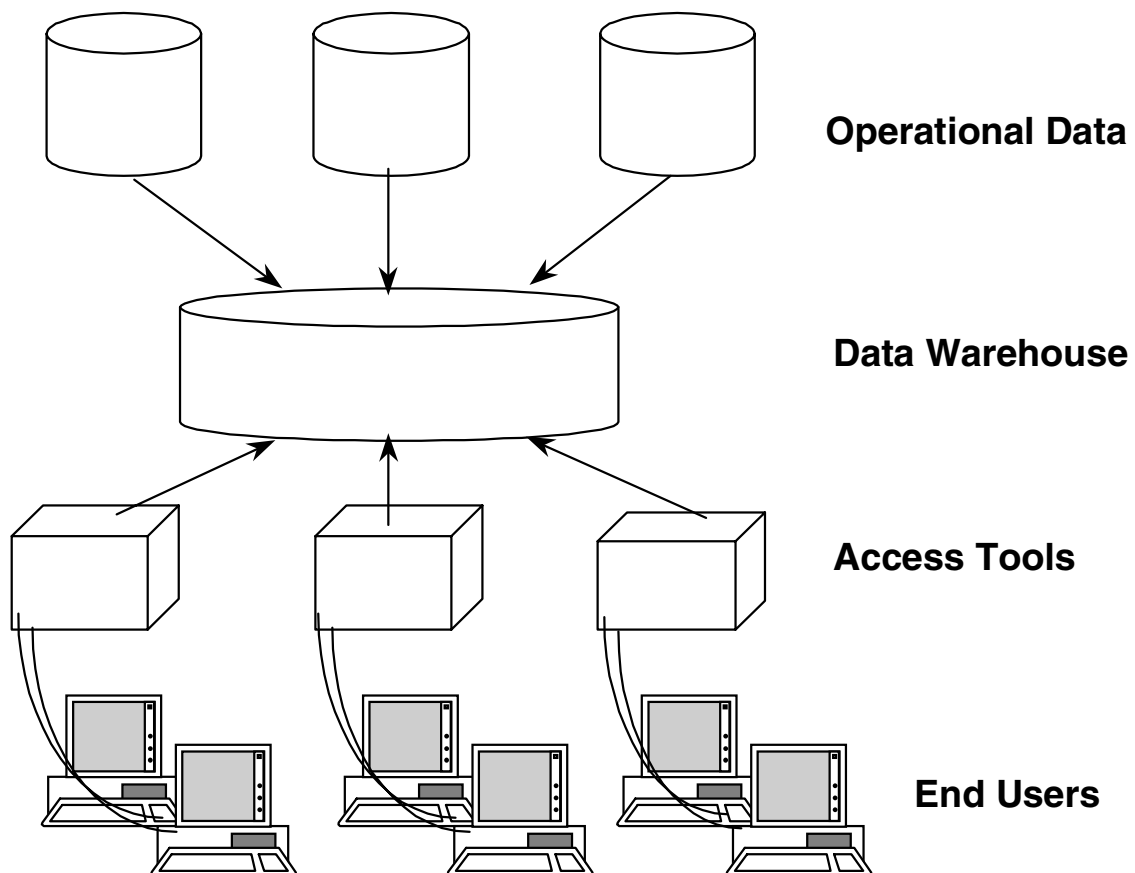
Because the data warehouse will carry a massive volume of detail data, it is particularly well suited to the parallel architecture of the Teradata RDBMS. Because Teradata has traditionally specialized in large-scale, decision-support queries against huge volumes of detail data, the use of Teradata for data warehousing becomes an obvious solution.

Teradata can run in single-node environments with a database of 10 Gigabytes and under, which also makes it well suited for data mart application. As data mart requirements grow, the single node can easily be expanded to multiple nodes with Teradata's linear expandability.

In the multi-node or MPP environment, Teradata can scale upward to systems supporting hundreds of terabytes of data. With the power of multiple CPUs propelling multiple AMPs over multiple disks, getting answers from warehouse detail data becomes fast and efficient.

## Teradata and the Data Warehouse

- Teradata provides a complete data warehousing solution.
- It runs on single node systems for most data mart requirements.
- It runs on multi-node systems for data warehouse requirements.
- Linear scalability allows for ease of expansion of systems.
- It can store hundreds of Terabytes of data on a single system.



## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.

## **Review Questions**

- 1. Name the four types of enterprise data processing and give examples of each.**
- 2. What is the difference between a data warehouse and a data mart?**
- 3. Which type of data mart gets its data directly from the data warehouse?**
- 4. Which type of data mart should be avoided if you want to have only one version of the truth?**
- 5. What distinguishes the data found in the warehouse from the data found in a mart?**

## Notes

### Teradata Configurations

After completing this module, you will be able to:

- Distinguish between SMP and MPP configurations.
- Name configurations that currently support the Teradata RDBMS.

## Notes



## Table of Contents

Single-Node SMP .....	4
MPP System.....	6
Multi-Node Cliques .....	8
BYNET (for MPP) .....	10
Platforms .....	12
Review Questions.....	14

## Single-Node SMP

In the **single-node SMP (Symmetric Multi-Processing)** implementation of Teradata, all applications run under UNIX or NT and all Teradata software runs under PDE. **All share the resources of CPUs and memory of the SMP node.**

**Note:** A node consists of one or more processors that share memory. The BYNET provides the communication interconnect between the nodes.

In addition to user applications, gateway software and channel driver support may also be running.

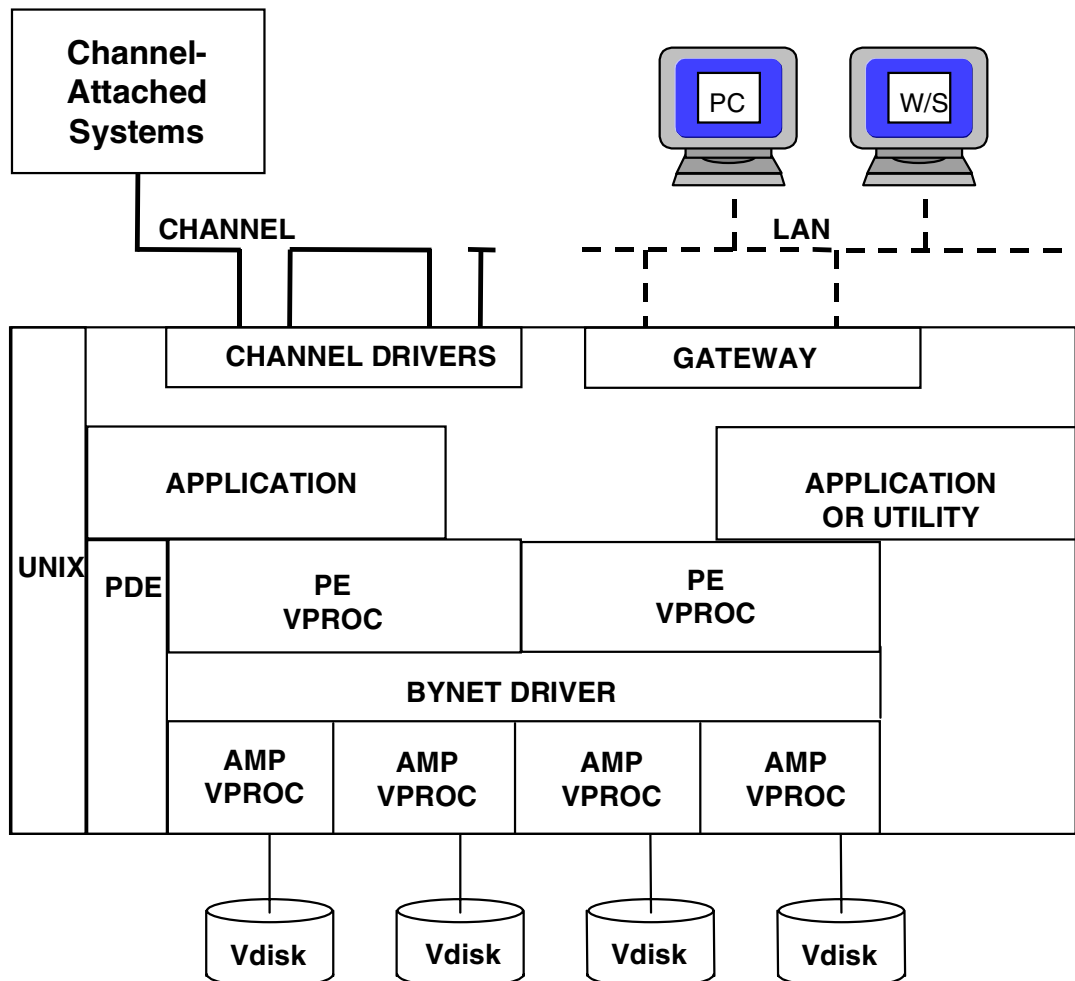
AMPs and PEs are **virtual processors** running under control of the PDE. Their numbers are software configurable.

AMPs are associated with **virtual disks (vdisks)** which are configured as ranks of a disk array.

All AMPs and PEs communicate via the **boardless BYNET**, which is a message flow control system. PDE actually controls message passing activity, while the boardless BYNET handles message queuing and flow control.

An application that runs under the control of PDE, such as the Teradata RDBMS, is considered a **Trusted Parallel Application (TPA)**.

## Single-Node SMP



- A node consists of **one or more processors that share memory.**
- Applications, the LAN gateway, and channel-driver software run as **UNIX processes.**
- AMPs and PEs are virtual processors (vprocs) which run under the Parallel Database Extensions (PDE).
- AMPs are associated with virtual disks (vdisks) via the disk array controller.
- Teradata is called a **Trusted Parallel Application (TPA) of PDE.**

## MPP System

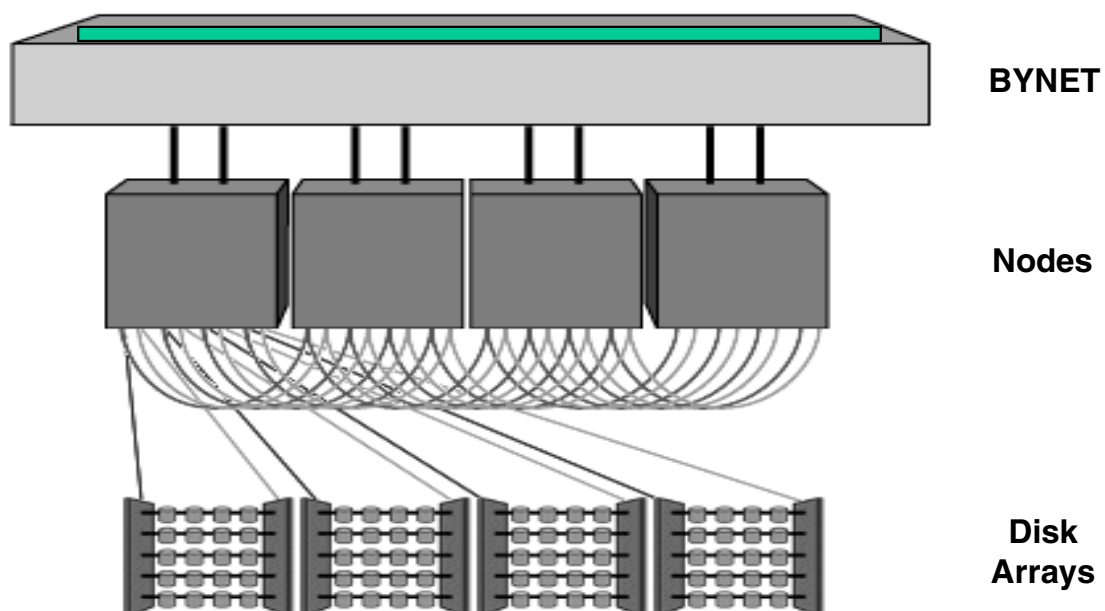
When **multiple SMP nodes** are connected to form a larger configuration, we refer to this as a **Massively Parallel Processing (MPP) system**.

The connecting layer, or message-passing layer, in the case of MPP systems is called the **BYNET**. The BYNET is a combination of hardware and software that allows multiple vprocs on multiple nodes to communicate with each other.

Because Teradata is a **linearly expandable** database system, as additional nodes and vprocs are added to the system, the system capacity scales in a linear fashion.

The BYNET can currently support 512 nodes with the capacity to support 2048 nodes in the future. Version 2 of the BYNET is rated at 200 MB per second per node per BYNETs per system, with current support for two BYNET per system and the capacity to expand to four BYNETs per system in the future.

## MPP System



- Multiple nodes may be configured to provide a **Massively Parallel Processing (MPP) system**.
- A physical message-passing layer called the **BYNET** is needed to **interconnect multiple nodes**.
- Teradata is linearly expandable—as your database grows, additional nodes may be added.
- The BYNET can support **512 nodes**.

## Multi-Node Cliques

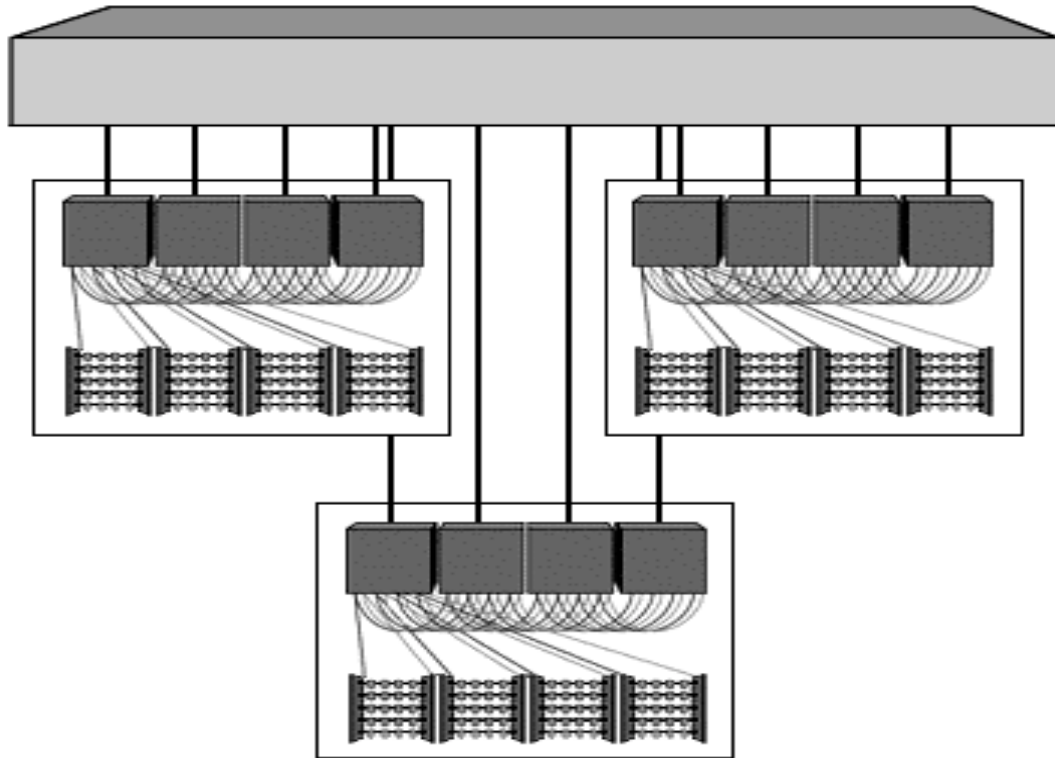
A **clique** is a set of Teradata nodes that share a common set of disk arrays. In the event of node failure, all vprocs can migrate to another available node in the clique. All nodes in the clique must have access to the same disk arrays.

The diagram on the facing page shows a twelve-node system consisting of three cliques, each containing four nodes. Because all disk arrays are available to all nodes in the clique, the AMP vprocs will retain access to their responsible rows.

In the event of three out of four nodes failing, the remaining node would attempt to absorb all vprocs from the failed nodes. Because each node can support a maximum of 128 vprocs, the total number of vprocs for the clique should not exceed 128.

**Note:** The maximum number of AMP and PE vprocs per system is 16,384.

## Multi-Node Cliques



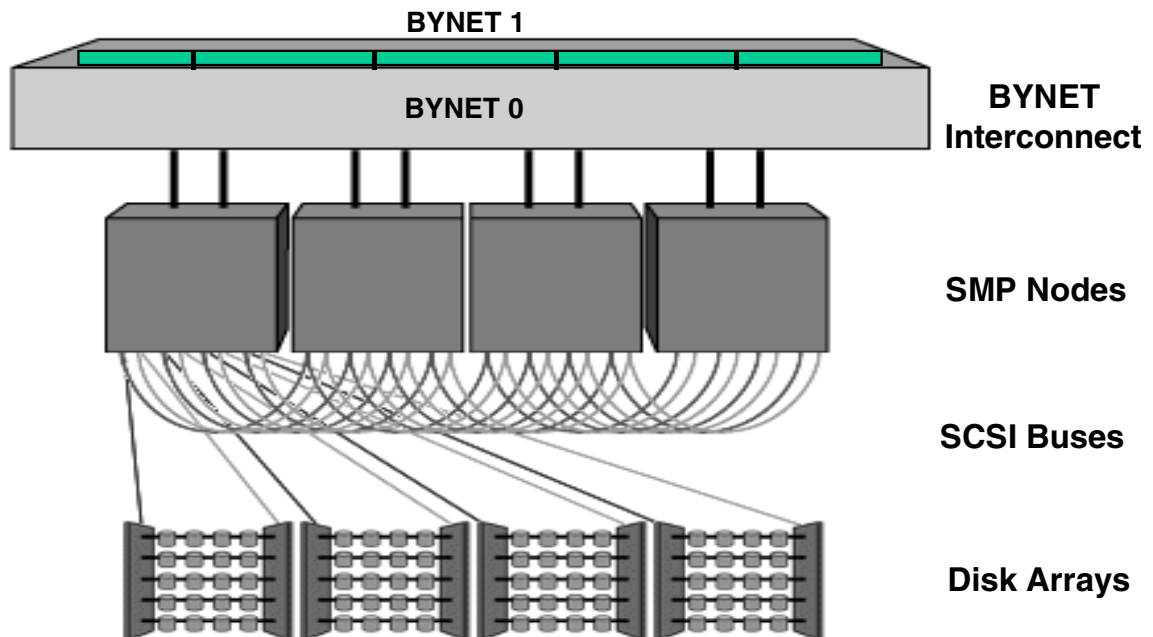
- **A clique is a defined set of nodes with failover capability.**
- All nodes in a clique must be able to access all vdisks for all AMPs in the clique.
- If a node fails, all vprocs will migrate to the remaining nodes in the clique.
- Each node can support 128 vprocs.
- **The total number of vprocs for the clique should not exceed 128.**

## **BYNET (for MPP)**

There are two physical **BYNETs**: BYNET 0 and BYNET 1. Both are fully operational and provide fault tolerance in the event of a BYNET failure. The BYNETs automatically handle load balancing and message routing. BYNET reconfiguration and message rerouting in the event of a component failure are handled transparently to the application.



## BYNET (for MPP)



The Bynet is a dual redundant, bi-directional interconnect network.

The Bynet enables:

- Communications between multiple SMP nodes (up to 512).
- Automatic load balancing of message traffic.
- Automatic reconfiguration after fault detection.
- **Fault tolerance** with fully-operational dual Bynets.
- Scalable bandwidth as nodes are added.

# Platforms

*Topics reflected on Certification Exam.*

The facing page identifies hardware and software platforms for Teradata V2R4.

Both NT and UNIX MP-RAS platforms support up to 36 GB disk drives.

**Note:** Bus and Tag (ESCON) channel capability is available for both UNIX MP-RAS and NT platforms.

## Nodes supported

The number of nodes supported for each platform include:

- 128 nodes for UNIX MP-RAS systems
- 4 nodes for Windows NT systems
- 128 nodes for Windows 2000 systems

## Database size

The size of relational databases stored by the Teradata RDBMS has a range of up to 2048 CPUs and 2048 GB of memory to support a 128 TB or larger database. The BYNET interconnect (bandwidth of more than 200 MB per second per node) supports up to 512 nodes.

# Platforms

## Hardware

V2R4 can run on:

- Teradata-supported UNIX MP-RAS platforms, including the following nodes: 4300, 4400, 4455, 4700, 4800, 4850, 5100, 5150, 5200, 5250
- Teradata-supported Windows NT platforms (SMP and MPP)
- Non-NCR platforms (SMP only), including:
  - Compaq
  - Dell
  - HP

## Software

- MP-RAS version 3.02 or higher
- Windows NT 4.0
- Windows 2000

Refer to the Teradata RBDMS Release Summary V2R4 for differences between MP-RAS and NT implementations.

## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.

## **Review Questions**

- 1. What is the communication layer in a single node Teradata system?**
- 2. What are cliques?**
- 3. What type of failure does a clique protect the system from?**
- 4. What is the maximum number of vprocs in a clique?**

# Notes

### Creating a Teradata Database

After completing this module, you will be able to:

- Distinguish between a Teradata Database and a Teradata User.
- Define Perm Space and explain how it is used.
- Define Spool Space and its use.
- Describe the hierarchy of objects in a Teradata system.

## Notes

*Attention Instructors! Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*



## Table of Contents

A Teradata Database .....	4
A Teradata User .....	6
The Hierarchy of Databases .....	8
Creating Tables .....	10
Data Types .....	12
Access Rights and Privileges .....	14
Review Questions.....	16

# A Teradata Database

*Topics reflected on Certification Exam.*

Databases provide a logical grouping for information (tables, views, and macros).

All databases have a defined upper limit of **permanent space**. Permanent space is used for storing the data rows of tables. Perm space is not pre-allocated. It represents a maximum limit.

All databases also have an upper limit of **spool space**. Spool space is temporary space used to hold intermediate query results or formatted answer sets to queries.

## A 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)

Other attributes may also be specified for a database:

- **Perm Space**—Amount of **space available for tables**
- **Spool Space**—Amount of **temporary work space used to hold results**

A database with no perm space can have views and macros but no tables.

A Teradata database is created with the **CREATE DATABASE** command.

### Example

```
CREATE DATABASE new_db FROM existing_db AS  
    PERMANENT = 20000000  
    ,SPOOL = 50000000  
;
```

‘New\_db’ is owned by ‘existing\_db.’

**A database is *empty* until objects are created within it.**

# A Teradata User

*Topics reflected on Certification Exam.*

A **user** is almost the same as a **database** except that a user has a password and can log on to the RDBMS. A user may or may not have perm space.

Users can access other databases depending on the privileges they have been granted.

## A Teradata User

- A Teradata user is a database with an assigned password.
- A user may own tables, views, and macros. A user with no perm space can own views and macros but not tables.
- A user may logon to Teradata and access objects within:
  - Itself
  - Other databases for which it has access rights
- 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 new_user FROM existing_user AS  
    PERMANENT = 10000000  
    ,PASSWORD = 'lucky_day'  
    ,SPOOL = 20000000  
;
```

'New\_user' is owned by 'existing\_user.'  
A user is *empty* until objects are created within it.

# The Hierarchy of Databases

*Topics reflected on Certification Exam.*

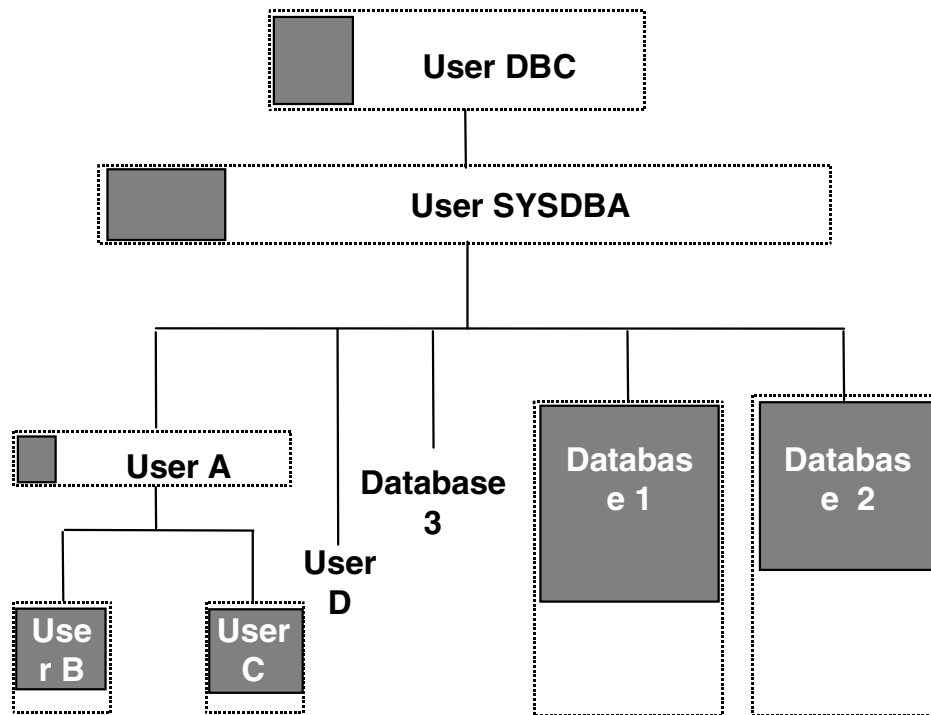
The facing page shows the relationship of users and databases to their parent objects.

Databases/Users and tables are organized into a hierarchical structure much like folders and files on your PC. At the **top** is the **super-user DBC**. Your System Manager or DBA creates your structure beneath that starting point.

In the example on the facing page, a User called **"SYSDBA"** was created. SYSDBA then created the other users and databases.

- Database 1 and Database 2 contain the production tables.
- Database 3 contains views and macros to control access to the data by end users and does not consume any database space.
- User D is an example of an end user who has no perm space but has been granted rights on certain views and macros to accomplish work.
- Users A, Users B, and Users C are examples of application developers who need some Perm space to hold sample data for program checkout.

# The Hierarchy of Databases



 Maximum Perm Space—available but not yet assigned to a table

 Current Perm Space—contains tables

No Box No Perm Space—may not contain tables

- At the top of the hierarchy is the **super-user DBC**.
- A new database or user must be created from an existing database or user.
- All perm space specifications are subtracted from the creator.
- Perm space is a zero-sum game—the total of all perm space allocations must equal the total amount of disk space available.
- Perm space is used for tables only.
- Perm space currently unassigned is available to be used as **Spool**.

# Creating Tables

**Creation of tables** is done via the Data Definition Language (DDL) portion of the SQL command vocabulary. The table definition, once accepted, is stored in the Data Dictionary (DD).

To create a table, a definition of at least one column and the assignment of a Primary Index is required. Columns are assigned data types, attributes, and may optionally be assigned constraints such as a range constraint.

Tables, like views and macros, may be dropped when they are no longer needed. Dropping a table both deletes the data from the table and removes the definition of the table from the DD.

Secondary indexes are optional and may be assigned when the table is created, or may be deferred until after the table has been built. Secondary indexes also may be dropped if they are no longer needed. **It is not uncommon to create secondary indexes to assist in the processing of a specific job sequence, and then to delete the index and its associated overhead once the job is complete.**

Indexes are discussed elsewhere in the course.



# Creating Tables

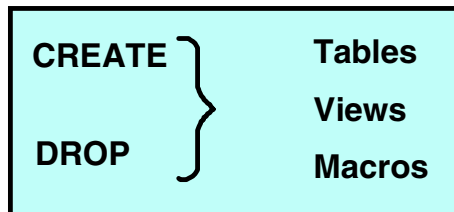
Creating a table requires:

- Defining columns
- Assigning a primary index
- Optional assignment of secondary indexes

**CREATE TABLE Employee**

```
(Employee_Number          INTEGER NOT NULL
,Last_Name                CHAR(20) NOT NULL
,First_Name               VARCHAR(20)
,Salary_Amount            DECIMAL(10,2)
,Department_Number        SMALLINT
,Job_Code                  CHAR(3))
Primary -----> UNIQUE PRIMARY INDEX (Employee_Number)
Secondary --> INDEX (Last_Name)
;
```

Database objects may be created or dropped as needed.



Secondary indexes may be:

- Created at table creation
- Created after table creation
- Dropped after table creation



# Data Types

When a table is created, a **data type** is specified for each column. Data types are divided into three classes: numeric, byte, and character.

The facing page shows data types.

**DATE** is a 32-bit integer that represents the date as yyyyymmdd. It supports century and year 2000 and is implemented with calendar-based intelligence. Be sure to refer to the documentation for a detail explanation of the date data type.

**TIME WITH ZONE** and **TIMESTAMP WITH ZONE** are ANSI-standard data types that allow support of clock and time-zone based intelligence.

**DECIMAL (n,m)** is a number of n digits, with m of these digits to the right of the decimal point.

**BYTEINT** is an 8-bit, signed binary whole number that may vary in range from -128 to +127.

**SMALLINT** is a 16-bit signed binary whole number that may vary in range from -32,768 to +32,767.

**INTEGER** is a 32-bit signed binary whole number that may vary in size from -2,147,483,648 to +2,147,483,647.

**FLOAT** is a 64-bit IEEE floating point number.

**BYTE (n)** is a fixed-length binary string of n bytes. Byte (and Varbyte) are never converted to a different internal format. They can be used for digitized objects.

**VARBYTE (n)** is a variable-length binary string of n bytes.

**CHAR (n)** is a fixed-length character string of n characters.

**VARCHAR (n)** is a variable-length character string of n characters.

**LONG VARCHAR** is the longest variable-length character string. It is equivalent to **VARCHAR (32000)**.

**GRAPHIC, VARGRAPHIC** and **LONG VARGRAPHIC** are the equivalent character types for multi-byte character sets such as Kanji.

Refer to the *Teradata RBDMS SQL Reference Volume 3, Data Types* for more information.

## Data Types

TYPE	Name	Bytes	Description
Date/Time	DATE	4	(YYYYMMDD)
	TIME (WITH ZONE)	8	(HHMMSSZZ)
	TIMESTAMP	12	(YYYYMMDD
	(WITH ZONE)		HHMMSSZZ)
Numeric	DECIMAL (18,18)	2, 4, 8	+ OR -
	NUMERIC (18,18)		
	BYTEINT	1	-128 to +127
	SMALLINT	2	-32,768 to + 32,767
	INTEGER	4	-2,147,484,648 to +2,147,483,647
	FLOAT	8	IEEE floating pt
Byte	BYTE (n)	0 - 64,000	
	VARBYTE (n)	0 - 64,000	
Character	CHAR (n)	0 - 64,000	
	VARCHAR (n)	0 - 64,000	
	LONG VARCHAR	same as VARCHAR(64000)	
	GRAPHIC	0 - 32,000	
	VARGRAPHIC	0 - 32,000	
	LONG VARGRAPHIC	same as VARGRAPHIC(32000)	

- Data is stored in 8-bit ASCII format on Teradata.
- Answer sets are returned in client format (8-bit ASCII or EBCDIC or Unisys 9-bit ASCII).

## Access Rights and Privileges

The diagram on the facing page shows access rights and privileges as they might be defined for the database administrator, programmer, user, system operator, and administrative user.

The **database administrator** has the right to use all of the commands in the data definition privileges, the data manipulation privileges, and the data control privileges.

The **programmer** has all of the above except the ability to GRANT privileges to others.

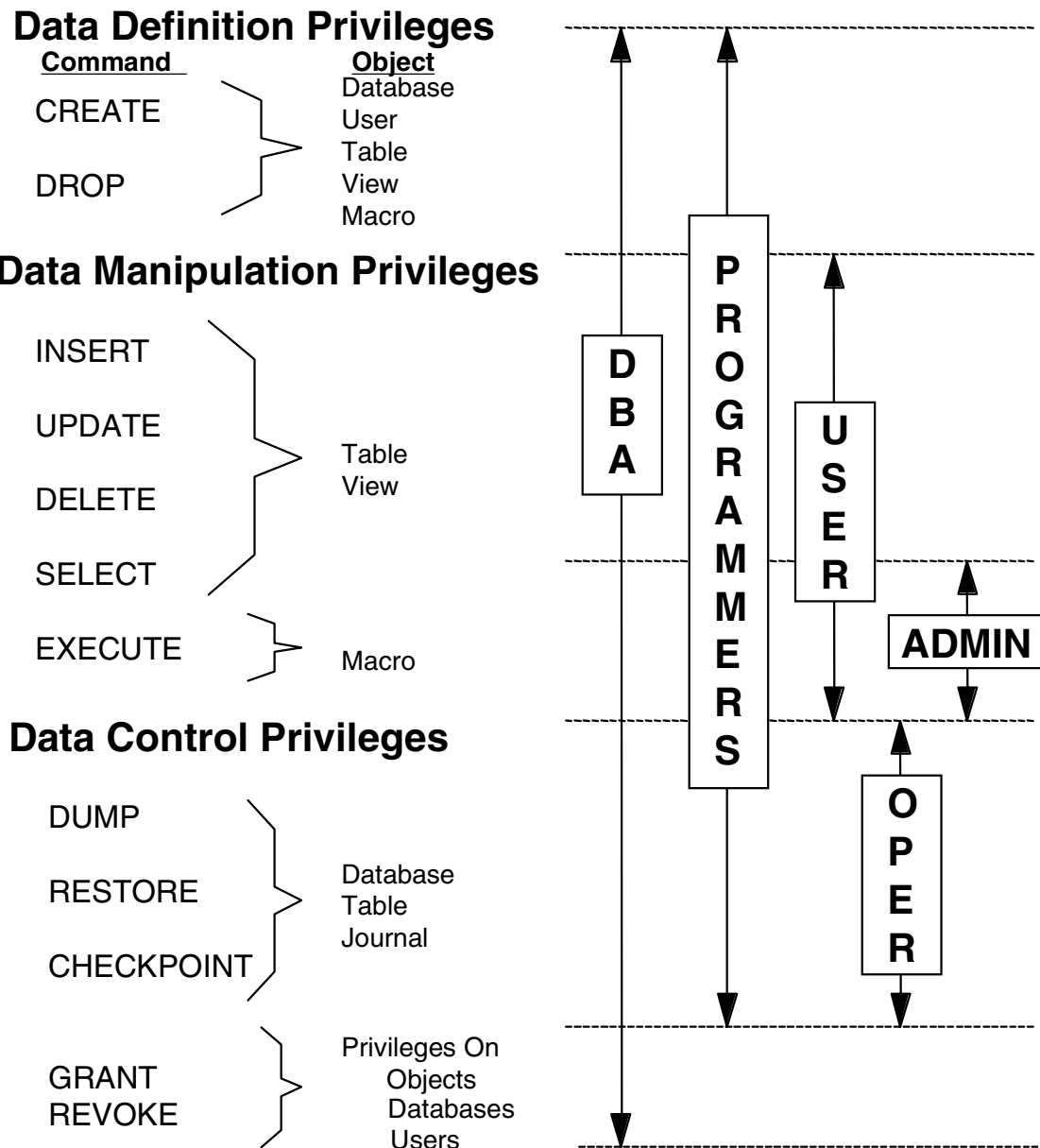
A typical **user** is limited to data manipulation privileges, while the **operator** is limited to data control privileges.

Finally, the **administrative user** is limited to a subset of data manipulation privileges, SELECT and EXECUTE.

Each site should carefully consider the access rules that best meet their needs.

# Access Rights and Privileges

## A Sample Scenario



- Access rights and privileges usually are explicitly defined by a Database Administrator.
- Access right information is kept in the Data Dictionary.

## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.

## **Review Questions**

**Indicate whether a statement is True or False.**

- 1. A database will always have tables.**
- 2. A user will always have a password.**
- 3. The super-user at the top of the hierarchy of databases is always “SYSDBA.”**
- 4. A user creating a subordinate user must give up some of their Perm Space.**
- 5. Tables must always be assigned a Primary Index at creation.**
- 6. Perm Space is not pre-allocated.**
- 7. The sum of all user and database Perm Space will equal the total available space on the system.**
- 8. The sum of all user and database Spool Space will equal the total available space on the system.**
- 9. Before a user can read a table, a table SELECT privilege must exist in the DD for that user.**
- 10. Deleting a macro from a database reclaims Perm Space for the database.**
- 11. Secondary indexes may be created or dropped at any time during the life of the table.**

## Notes



### Storing and Accessing Data Rows

After completing this module, you will be able to:

- Explain the purpose of the Primary Index.
- Distinguish between Primary Index and Primary Key.
- State the reasons for selecting a UPI vs. a NUPI.

## Notes

*Attention Instructors! Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*

## Table of Contents

Efficient Data Storage and Access .....	4
Storing Rows .....	6
Creating a Primary Index.....	8
Primary Index Values .....	10
Accessing Via a Unique Primary Index.....	12
Accessing Via a Non-Unique Primary Index.....	14
Primary Keys and Primary Indexes .....	16
Duplicate Rows.....	18
Row Distribution Using a Unique Primary Index (UPI) (Case 1).....	20
Row Distribution Using a Non-Unique Primary Index (NUPI) (Case 2)...	22
Row Distribution Using a Highly Non-Unique Index (NUPI) (Case 3) .....	24
Review Questions.....	26
Reference .....	28

## Efficient Data Storage and Access

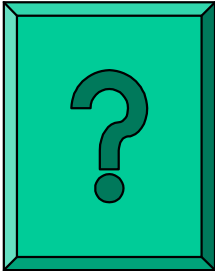
Tables in a Teradata system can be extremely large. This means that getting results from a query could take a long time. It also means that, if the data is not evenly distributed, system performance will not be optimal.

Teradata is designed to help prevent these situations. With Teradata, you can use **indexes** to help distribute the data evenly, and to make accessing data faster.

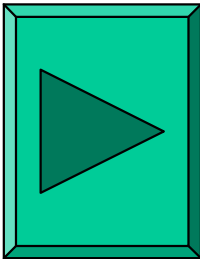
Database designers need to know how to set up indexes for maximum efficiency. Users who access data can use indexes to get query results faster.

## Efficient Data Storage and Access

- Teradata tables can be very large.
- The system itself can be very large, with many nodes, AMPs, and disks.



- How does the system efficiently store data so that all resources are used optimally?
- How can the system efficiently access data?



## Indexes

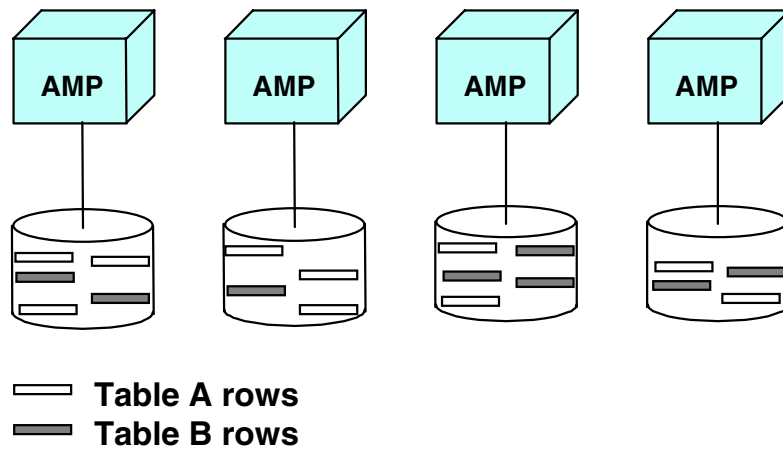
## Storing Rows

Each AMP is designed to hold a portion of the rows of each table. An AMP is responsible for the **storage, maintenance and retrieval** of the data under its control.

Ideally, the rows of every table will be distributed among all of the AMPs. There may be some circumstances where this is not true. For example, what if there are fewer rows than AMPs? In this case, at least some AMPs will hold no rows from that table. (This example should be considered the exception, and not the rule.)

In an ideal situation, the rows of each table will be *evenly* distributed across all of the AMPs. Even distribution is desirable because in an operation involving all rows of the table (such as a full table scan), each AMP will have an equal portion of work to do. When workloads are not evenly distributed, the desired response will be only as fast as the slowest AMP.

## Storing Rows



- The rows of *every* table are distributed among all AMPs.
- Each AMP is responsible for a *subset* of the rows of each table.
- Ideally, each table will be evenly distributed among all AMPs.
- Evenly distributed tables result in evenly distributed workloads.

# Creating a Primary Index

*Topics reflected on Certification Exam.*

**Indexes** are used to access rows from a table without having to search the entire table.

On Teradata, the **Primary Index (PI)** is the mechanism for assigning a data row to an AMP and a location on the AMP's disks. **When a table is created, it must have a Primary Index specified. The PI cannot be changed without dropping and creating the table.**

Choosing a **Primary Index** for a table is perhaps the most critical decision a database designer makes. The choice will affect the distribution of the rows of the table and the performance of the table in a production environment. Although many tables use combined columns as the Primary Index choice, the examples we use here are single-column indexes.

There are two types of primary index —**unique (UPI)** and **non-unique (NUPI)**.

**A Unique Primary Index (UPI) is a column that has no duplicate values.** UPIs are desirable because they guarantee uniform distribution of table rows.

Because it is not always feasible to pick a Unique Primary Index, it is sometimes necessary to pick a column (or columns) which have non-unique values, that is, there are duplicate values. This type of index is called a **Non-Unique Primary Index or NUPI**. While not a guarantor of uniform row distribution, the degree of uniqueness of the index will determine the degree of uniformity of the distribution. Because **all rows with the same PI value end up on the same AMP**, columns with a small number of distinct values that are repeated frequently do not make good PI candidates.

Choosing a Primary Index is not an exact science. It requires analysis and thought for some tables and will be completely self-evident on others.

The Primary Index is always designated as part of the **CREATE TABLE** statement. **Once you choose a Primary Index for a table, it cannot be changed to something else.** If an alternate choice of column(s) is desired for the PI, it is necessary to drop and recreate the table.



## Creating a Primary Index

- **A Primary Index is defined at table creation.**
- It may consist of a single column or a combination of up to 16 columns.
- All rows with the same PI value are on the same AMP.
- The choice of a PI is important as it affects the distribution of rows, and thus performance.

### UPI

```
CREATE TABLE sample_1
      (col_a      INT
       ,col_b      INT
       ,col_c      INT)
UNIQUE PRIMARY INDEX (col_b);
```

- If the index choice of column(s) is unique, we call this a **UPI** (Unique Primary Index).
- A UPI choice will result in even distribution of the rows of the table across all AMPs.

### NUPI

```
CREATE TABLE sample_2
      (col_x      INT
       ,col_y      INT
       ,col_z      INT)
PRIMARY INDEX (col_x);
```

- If the index choice of column(s) isn't unique, we call this a **NUPI** (Non-Unique Primary Index).
- A NUPI choice will result in even distribution of the rows of the table proportional to the degree of uniqueness of the index.
- A NUPI can create a skewed distribution.

**Note:** Changing the Primary Index requires dropping and recreating the table.

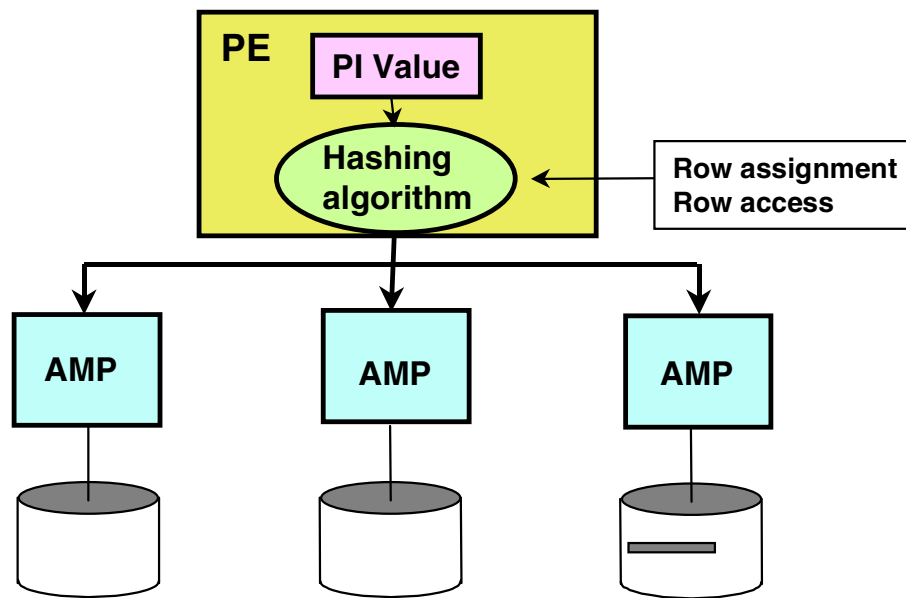
## Primary Index Values

Primary Indexes have a powerful effect on database performance. The most important thing to remember is that a Primary Index is the mechanism used to **assign each row to an AMP** and may be used to retrieve that row from the AMP. Retrievals, updates, and deletes that specify the Primary Index will be much faster than those that don't. The selection of a Primary Index is probably the most important factor in the efficiency of join processing.

The Primary Key is **unique, unchanging**, and based on the **logical model** of the data. The Primary Index may change and may be **non-unique**; it is chosen for **the physical performance** of the database.

## Primary Index Values

- The value of the Primary Index for a specific row **determines its AMP assignment.**
- This is done using a **hashing algorithm.**



Accessing the row by its Primary Index value is:

- **Always a *one-AMP* operation**
- The most efficient way to access a row

Other table access techniques:

- Secondary index access
- Full table scans

## Accessing Via a Unique Primary Index

**A Primary Index operation** is always a one-AMP operation. In the case of a UPI, the one-AMP access can return, at most, one row. In the facing example, we are looking for the row whose primary index value is 45. If you specify the PI value as part of our selection criteria, you are guaranteed that only the AMP containing the specified row will be searched.

The system locates the correct AMP by taking the PI value and passing it through a hashing algorithm. Hashing takes place in the Parsing Engine. The output of the hashing algorithm contains information that will point to a specific AMP. Once it has isolated the appropriate AMP, finding the row is quick and efficient.

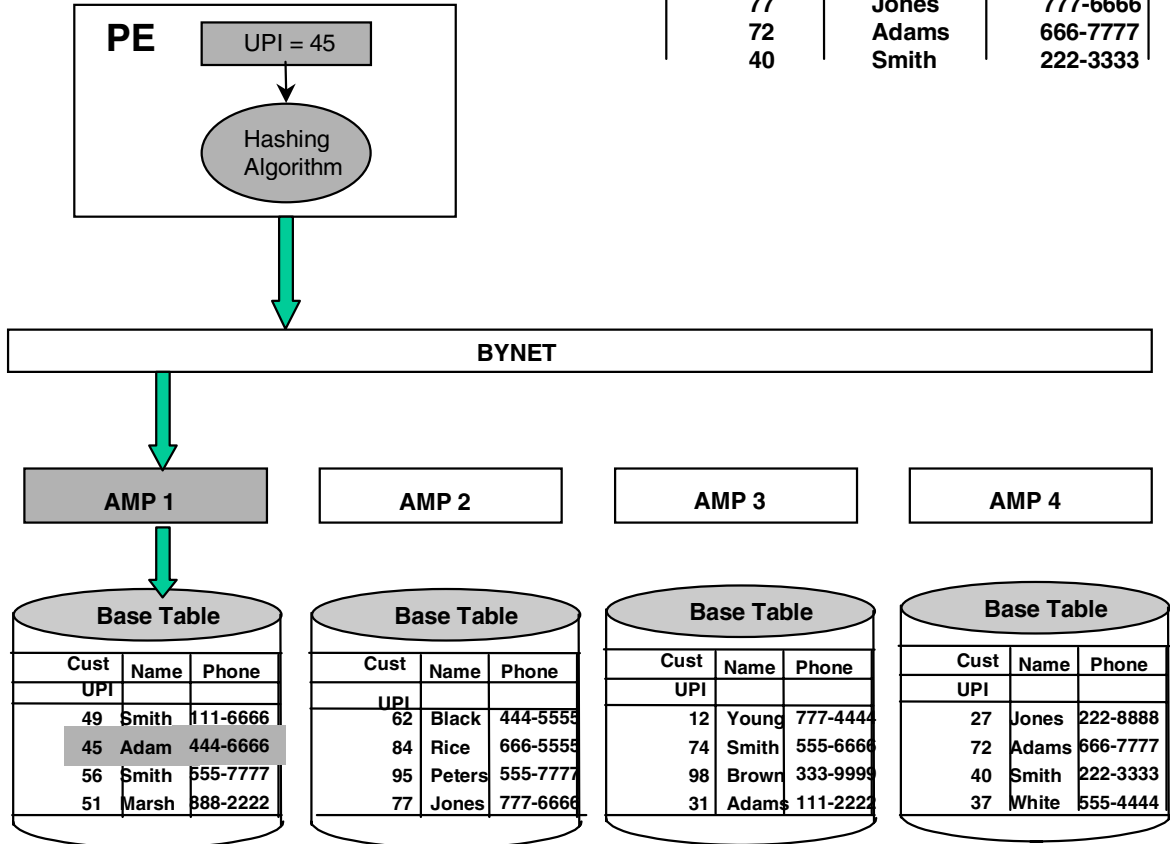
# Accessing Via a Unique Primary Index

```
CREATE TABLE Customer
(Cust INT
,Name CHAR(10)
,Phone CHAR(8) )
UNIQUE PRIMARY INDEX (Cust);
```

**CUSTOMER table**

Cust	Name	Phone
PK		
UPI		
37	White	555-4444
98	Brown	333-9999
74	Smith	555-6666
95	Peters	555-7777
27	Jones	222-8888
56	Smith	555-7777
45	Adams	444-6666
84	Rice	666-5555
49	Smith	111-6666
51	Marsh	888-2222
31	Adams	111-2222
62	Black	444-5555
12	Young	777-4444
77	Jones	777-6666
72	Adams	666-7777
40	Smith	222-3333

SELECT \* FROM customer WHERE cust = 45;



## Accessing Via a Non-Unique Primary Index

**A Non-Unique Primary Index operation (NUPI)** is also a one-AMP operation. In the case of a NUPI, the one-AMP access can return zero to many rows. In the example on the facing page, we are looking for the rows whose primary index value is 555-7777. By specifying the PI value as part of our selection criteria, we are once again guaranteeing that only the AMP containing the required rows will need to be searched.

The correct AMP is located by taking the PI value and passing it through a hashing algorithm executed in the Parsing Engine. The output of the hashing algorithm will once again point to a specific AMP. Once it has isolated the appropriate AMP, it must find all rows that have the specified value. In the example, the AMP returns two rows.

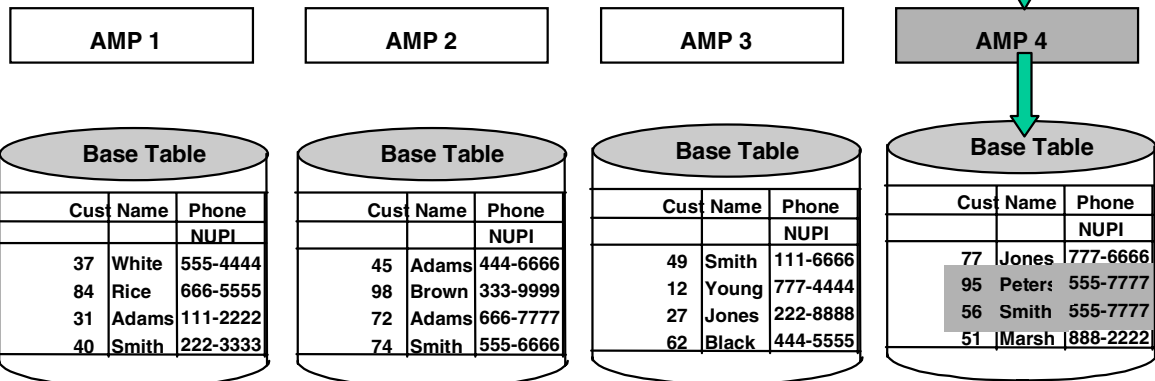
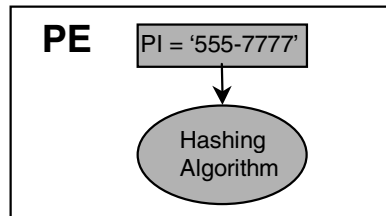
# Accessing Via a Non-Unique Primary Index

```
CREATE TABLE Customer
(Cust INT
,Name CHAR(10)
,Phone CHAR(8) )
PRIMARY INDEX (Phone);
```

```
SELECT * FROM customer WHERE phone = '555-7777';
```

**CUSTOMER table**

Cust	Name	Phone
PK		
		NUPI
37	White	555-4444
98	Brown	333-9999
74	Smith	555-6666
95	Peters	555-7777
27	Jones	222-8888
56	Smith	555-7777
45	Adams	444-6666
84	Rice	666-5555
49	Smith	111-6666
51	Marsh	888-2222
31	Adams	111-2222
62	Black	444-5555
12	Young	777-4444
77	Jones	777-6666
72	Adams	666-7777
40	Smith	222-3333



## Primary Keys and Primary Indexes

While it is true that many tables use the same columns for both Primary Indexes and Primary Keys, **Indexes are conceptually different from Keys**. The table on the facing page summarizes those differences.

- **A Primary Key is relational, data-modeling term.** It defines, in the logical model, the columns that uniquely identify a row.
- **A Primary Index is a physical database implementation term that defines the actual columns used to distribute and access rows in a table.**

A significant percentage of the tables in any database will use the same column(s) for both the PI and the PK. However, one should expect that in any real scenario there might be some tables that will not conform to this rule. Only after a careful analysis of the type of processing that will take place can the tables be properly evaluated for PI candidates. **Remember, changing your mind about the PI means reloading the table.**



# Primary Keys and Primary Indexes

*Indexes* are conceptually different from *keys*:

- A **PK** is a relational modeling convention which allows each row to be uniquely identified.
- A **PI** is a Teradata convention which determines how the row will be stored and accessed.

Primary Key	Primary Index
Logical concept of data modeling	Physical mechanism for access and storage
Teradata doesn't need to recognize	Each table must have exactly one
No limit on column numbers	16-column limit
Documented in data model (Optional in CREATE TABLE)	Defined in CREATE TABLE statement
Must be unique	May be unique or non-unique
Identifies each row	Used to place and locate each row on an AMP
Values should not change	Values may be changed (Del+ Ins)
May not be NULL—requires a value	May be NULL
Does not imply an access path	Defines most efficient access path
Chosen for logical correctness	Chosen for physical performance

- A significant percentage of tables may use the same columns for both the PK and the PI.
- A well-designed database will use a PI that is different from the PK for some tables.

## Duplicate Rows

A **duplicate row** is a row whose column values are all identical to another row of the same table. If designers adhere to the rule that **a Primary Key must be unique**, then they should preclude the possibility of having duplicate rows.

Having said that, the ANSI standard does permit duplicate rows to satisfy the requirements of vendors who rely on them for certain types of auditing systems. For example, if I am loading a table from several different databases and the same record appears from three different places, I might want to know that it originated from those three places.

Even though the above example contradicts relational theory, the standard generously permits duplicate rows for these anomalous situations.

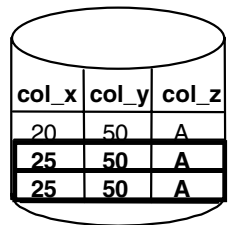
Teradata, adhering to the ANSI standard, permits duplicate rows by specifying that you wish to create a **MULTISET table**. The default, however, is **a SET table** that does not permit duplicate rows. A table must be defined as either SET or MULTISET.

When MULTISET is enabled, Teradata does not do a duplicate row check for new rows added. If SET is enabled, it will only do this check if the Primary Index is a NUPI, otherwise the duplicate index check itself will suffice.

If MULTISET is enabled, it will be overridden by choosing a UPI as Primary Index. Choosing a UPI as the PI effectively disables the MULTISET.

## Duplicate Rows

A duplicate row is a row whose column values are all identical to another row of the same table.



col_x	col_y	col_z
20	50	A
25	50	A
25	50	A

Duplicate Rows

- Because a PK uniquely identifies each row, ideally a relational table should not have duplicate rows!
- By creating a **MULTISET** table rather than a default **SET** table, Teradata permits duplicate rows for specialized situations.
- A table must be defined as either **SET** or **MULTISET**.

The Teradata default

```
CREATE SET TABLE tbl_a  
:  
:
```

Checks for\* and *disallows* duplicate rows.

The ANSI default

```
CREATE MULTISET TABLE tbl_b  
:  
:
```

Doesn't check for and *allows* duplicate rows.

\* If a UPI is selected on a **SET** table, the duplicate row check is replaced by a check for duplicate index values.

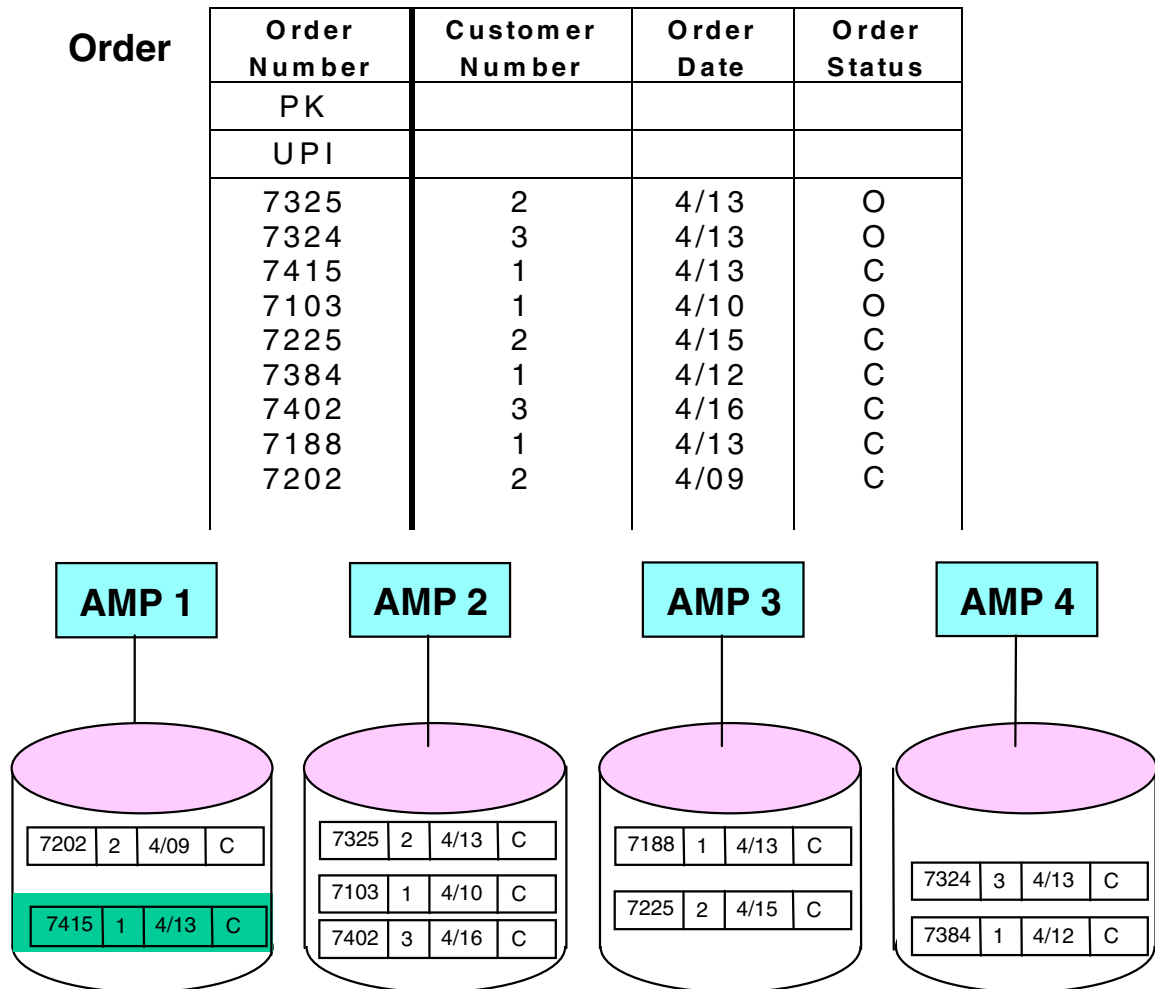
## Row Distribution Using a Unique Primary Index (UPI) (Case 1)

At the heart of the Teradata database is a way of predictably distributing and retrieving rows across AMPs, the Teradata hashing algorithm. The same value stored in the same data type will always produce the same hash value. If the Primary Index is unique, Teradata can distribute the rows **evenly**. If the Primary Index is non-unique and there are only four or five rows per index value, the table will still distribute **evenly**. But if there are hundreds or thousands of rows for some index values, the distribution will probably be **lumpy**.

In the example on the facing page, the Order\_Number is used as a unique primary index. Since the primary index value for Order\_Number is unique, the distribution of rows among AMPs is uniform. This assures maximum efficiency because each AMP does approximately the same amount of work. No AMPs sit idle waiting for another AMP to finish a task.

This way of storing the data provides for maximum efficiency and makes the best use of the parallel features of the Teradata system.

## Row Distribution Using a Unique Primary Index (UPI) (Case 1)



- The PK column(s) will often be used as a UPI.
- PI values for Order\_Number are known to be unique (it's a PK).
- Teradata will distribute different index values evenly across all AMPs.
- Resulting row distribution among AMPs is uniform.

## Row Distribution Using a Non-Unique Primary Index (NUPI) (Case 2)

*Topics reflected on Certification Exam.*

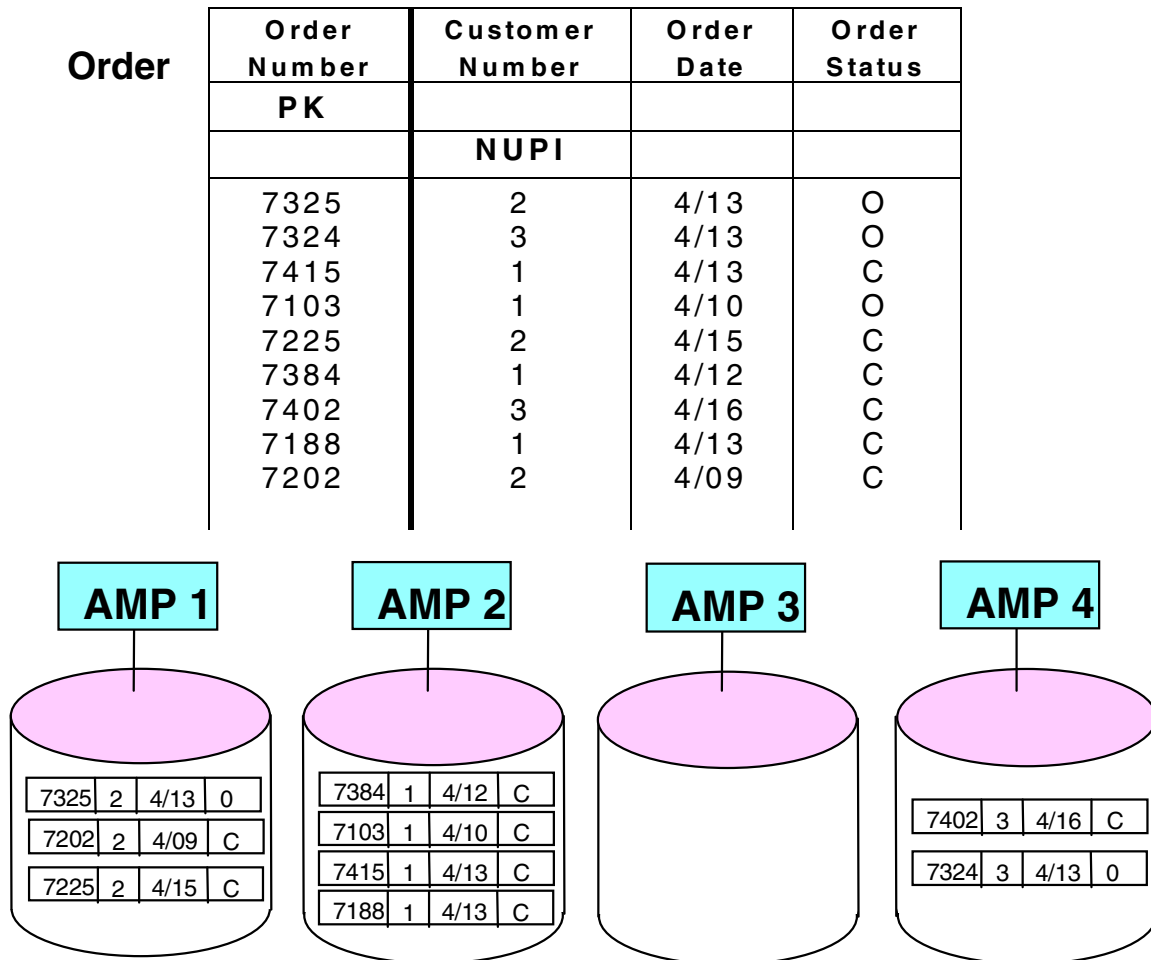
In the example on the facing page, **Customer\_Number** has been used as a non-unique Primary Index (NUPI). Note that row distribution among AMPs is uneven. All rows with the same primary index value (with the same customer number) are stored on the same AMP.

Customer\_Number has three possible values, so all the rows are hashed to three AMPs, leaving the fourth AMP without rows from this table. While this distribution will work, it is not as efficient as spreading all rows among all AMPs.

AMP 2 has a disproportionate number of rows and AMP 3 has none. In an all-AMP operation, AMP 2 will take longer than the other AMPs. The operation cannot complete until AMP 2 completes its tasks. Overall operation time is increased and some AMPs are under-utilized.

The example above shows how NUPIs can create **irregular distributions**, called **skewed distributions**. AMPs that have more than an average number of rows will take longer for full-table operations than other AMPs. Because an operation is not complete until all AMPs have finished, the operation where distribution is skewed will take longer than it would if all AMPs were utilized evenly.

## Row Distribution Using a Non-Unique Primary Index (NUPI) (Case 2)



- Customer\_Number may be the preferred access column for ORDER table, thus **a good index candidate**.
- Values for Customer\_Number are **non-unique** and therefore a NUPI.
- **Rows with the same PI value distribute to the same AMP causing row distribution to be less uniform or skewed.**

## Row Distribution Using a Highly Non-Unique Index (NUPI) (Case 3)

The example on the facing page uses **Order\_Status** as a **NUPI**. **Order\_Status** is a poor choice, because it yields the most **uneven distribution**. Because there are only two possible values for **Order\_Status**, all rows are placed on two AMPs.

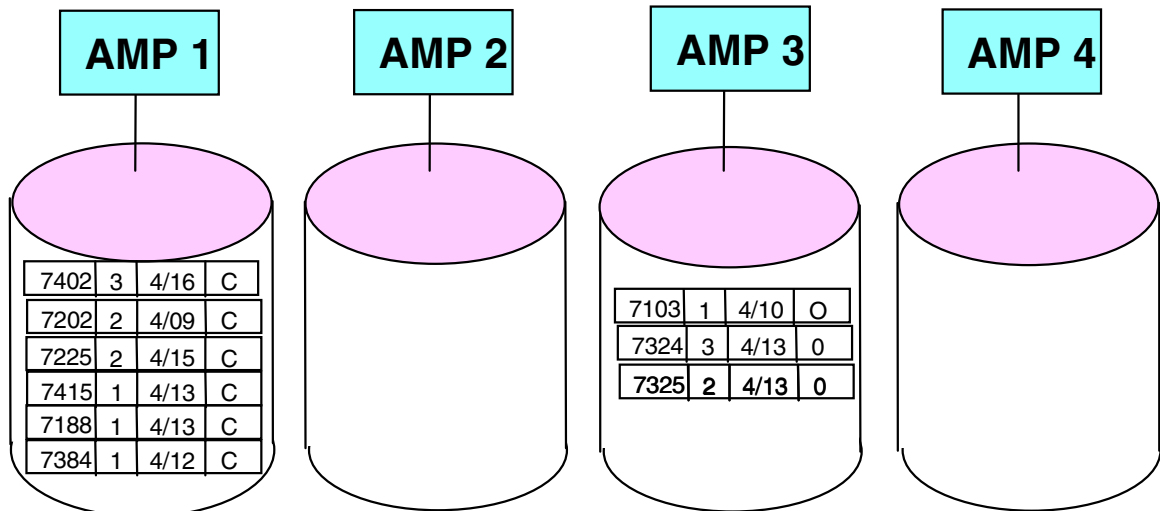
**STATUS** is an example of a **highly non-unique Primary Index**.

**When choosing a Primary Index, never pick a column with a severely-limited value set.** The degree of uniqueness is critical to efficiency. Choose NUPIs that allow all AMPs to participate fairly equally.



## Row Distribution Using a Highly Non-Unique Index (NUPI) (Case 3)

Order	Order Number	Customer Number	Order Date	Order Status
	PK			
				NUPI
	7325	2	4/13	O
	7324	3	4/13	O
	7415	1	4/13	C
	7103	1	4/10	O
	7225	2	4/15	C
	7384	1	4/12	C
	7402	3	4/16	C
	7188	1	4/13	C
	7202	2	4/09	C



- Values for Order\_Status are **highly non-unique**, and therefore, it is a NUPI.
- Only two values exist, so only two AMPs will ever be used for this table.
- This table will not perform well in parallel operations.
- **Highly non-unique columns** are poor PI choices.
- The degree of uniqueness is critical to efficiency.

## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.

## **Review Questions**

- 1. For each statement, indicate whether it applies to: UPIs, NUPIs, Either, or Neither**
  - a. Specified in CREATE TABLE statement.**
  - b. Provides uniform distribution via the hashing algorithm.**
  - c. May be up to 16 columns.**
  - d. Always a one-AMP operation.**
  - e. Access will return a single row.**
  - f. Used to assign a row to a specific AMP.**
  - g. Allows nulls.**
  - h. Value cannot be changed.**
  - i. Required on every table.**
  - j. Permits duplicate rows.**
  - k. Can never be the PK.**
- 2. Why is the choice of a Primary Index important?**

## Reference

For more information on storing and accessing data rows, refer to *Teradata RDBMS Database Design*.

### Primary Index Mechanics

**After completing this module, you will be able to:**

- **Explain the roles of the hashing algorithm and hash map in locating rows.**
- **Explain the makeup of the Row-ID and its role in row storage.**
- **Describe the sequence of events for locating a row given its PI value.**

## Notes

*Attention Instructors! Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*

## Table of Contents

Hashing Down to the AMPs.....	4
A Hashing Example .....	6
The Hash Map .....	8
Identifying Rows .....	10
The Row-ID .....	12
Physical Data Block Layout .....	14
Physical Row Layout .....	16
Review Questions.....	18

# Hashing Down to the AMPs

*Topics reflected on Certification Exam.*

By understanding how the system uses hashing and Row-IDs to distribute data, you will see how indexes contribute to efficient use of the system.

## How rows are distributed

The rows of all tables are distributed across the AMPs according to their **primary index** value. The primary index value goes into the hashing algorithm. The output is a 32-bit **row hash**. The high-order 16 bits are referred to as the **Destination Selection Word (DSW)** and are used to **identify a hash-map entry**. This entry is used to identify the AMP to be targeted. The remaining 16 bits are not used to locate the AMP.

The entire 32-bit row hash is used by the selected AMP to locate the row within its disk space.

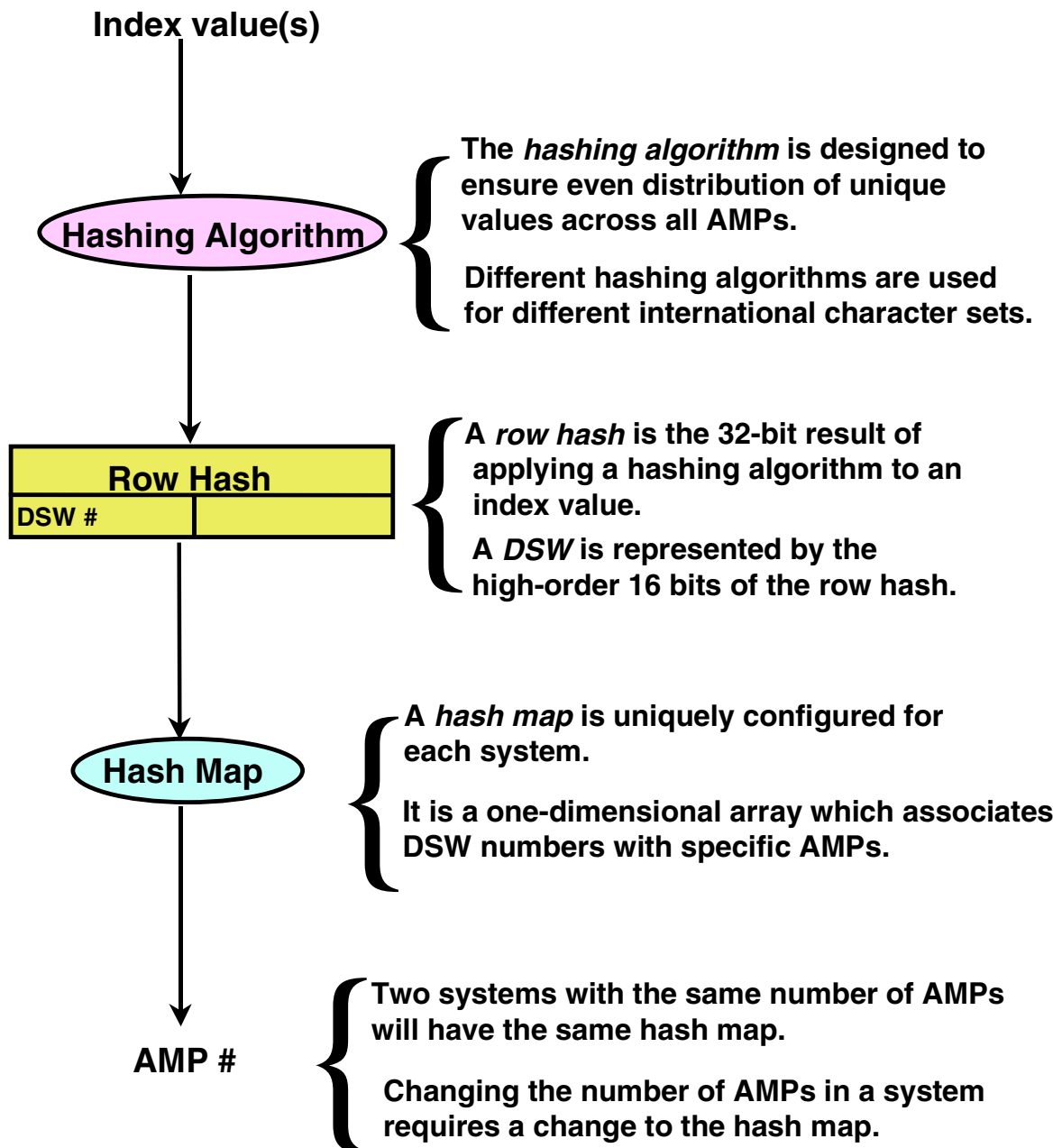
**Hash maps** are uniquely configured for each **size** of system. A 50-AMP system will have a different hash map from a 30-AMP system.

A hash map **associates bucket numbers** with specific AMPs.

When a system grows, new AMPs are typically added. **Adding AMPs** requires a change to the hash map to reflect the new total number of possible target AMPs.



## Hashing Down To The AMPs



# A Hashing Example

*Topics reflected on Certification Exam.*

The facing page shows an example of how the hashing algorithm would produce a 32-bit row hash value on the primary index value, 7225.

The hash value is divided into two parts. The first 16 bits are the **Destination Selection Word (DSW)**. The DSW consists of:

- 4 bits that select a hash map
- A 12-bit hash map buckets—number that points to a position in the hash map.

Thus, the DSW points to a particular hash map entry, which in turns points to one AMP. The entire row hash along with the Table ID references a particular logical location on that AMP.

## A Hashing Example

Order

Order Number	Customer Number	Order Date	Order Status
PK			
UPI			
7325	2	4/13	O
7324	3	4/13	O
7415	1	4/13	C
7103	1	4/10	O
7225	2	4/15	C
7384	1	4/12	C
7402	3	4/16	C
7188	1	4/13	C
7202	2	4/09	C

```
SELECT * FROM order
WHERE order_number = 7225;
```

7225 (Primary index value)

Hashing Algorithm

32 bit Row Hash

DSW #	Remaining 16 bits
0000 0000 0001 1010	1100 0111 0101 1011

□ □ □ □

0 0 1 A

(Hexadecimal)

# The Hash Map

*Topics reflected on Certification Exam.*

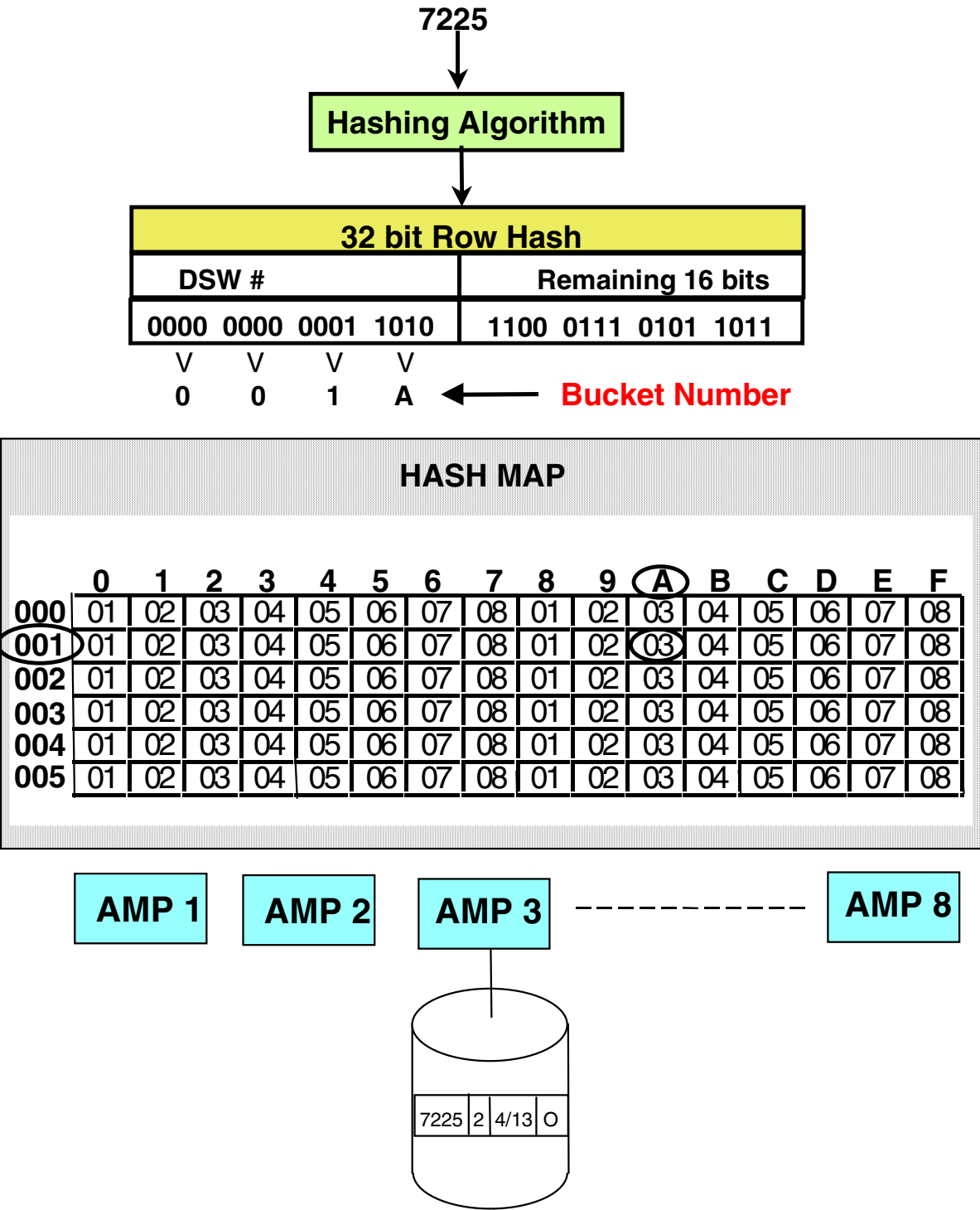
A **hash map** is a two-dimensional array that associates DSW values with specific AMPs. The diagram on the facing page shows it folded after 16 entries and represents what a hash map might look like for an eight-AMP system.

To determine the destination AMP for a primary index operation, the hash map is checked by BYNET software using the row hash information. The **BYNET** looks up the entry in the Hash Map based on the **bucket number**. A message is placed on the BYNET to be sent to the target AMP using point-to-point communication.

In the example on the facing page, the bucket entry 001A (hexadecimal) contains an entry for AMP 3. AMP 3 will be the recipient of the message from the BYNET.

One major difference between Version 1 and Version 2 is the increase in the number of hash map entries from 3643 to 65,536. This increase results in fewer hash collisions, better performance, and the ability to configure systems with thousands of AMPs rather than hundreds of AMPs.

# The Hash Map



## Identifying Rows

Can two different Primary Index (PI) values come out of a hashing algorithm with the same row hash value? Yes. There are two ways this can happen:

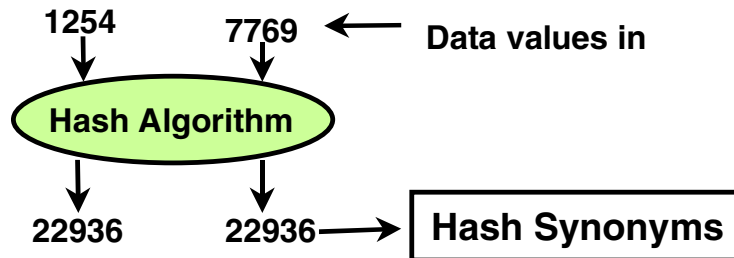
- Two different primary index values may hash identically. This is called a **hash synonym**.
- If a non-unique primary index is used, **duplicate NUPI values will produce the same row hash**.

# Identifying Rows

A row hash is not adequate to uniquely identify a row.

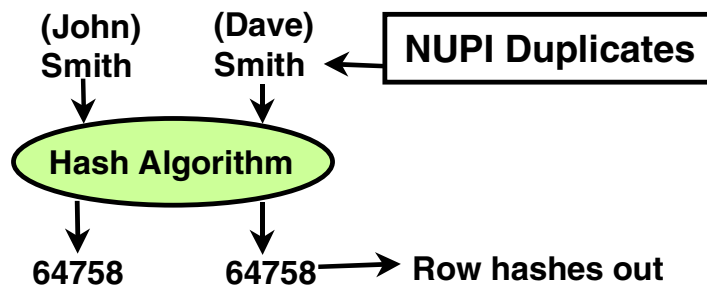
- **Consideration #1**

- A Row Hash = 32 bits = 4.2 billion possible values.
- Because there is an infinite number of possible values, some values will have to share the same row hash.



- **Consideration #2**

- A primary index may be non-unique (NUPI).
- Different rows will have the same PI value and thus the same row hash.



- **Conclusion**

- A row hash is not adequate to uniquely identify a row.

# The Row-ID

*Topics reflected on Certification Exam.*

To differentiate each row in a table, every row is assigned a unique **Row-ID**. The Row-ID is a combination of the **row hash** value plus a **uniqueness value**. The AMP appends the uniqueness value to the row hash when it is inserted. The uniqueness value is used to differentiate between PI values that generate identical row hashes.

The first row inserted with a particular row hash value is assigned a uniqueness value of 1. Each new row with the same row hash is assigned an integer value one greater than the current largest uniqueness value for this Row-ID.

If a row is deleted or the primary index is modified, the uniqueness value can be reused.

Only the row hash portion is used in primary index operations. **The entire Row-ID is used for secondary index support.**



## The Row-ID

To uniquely identify a row, the system adds a 32-bit uniqueness value.

- The combined row hash and uniqueness value is called a Row-ID.

Row-ID

Row Hash (32 bits)	Uniqueness ID (32 bits)
-----------------------	----------------------------

- Row hash only is used for PI operations.
- Entire Row-ID is used for SI operations.

- Each stored row has a Row-ID as a prefix.

Row-ID	Row Data
--------	----------

- Rows are logically maintained in Row-ID sequence.

Row-ID		Row Data		
Row Hash	Unique ID	EmpNo	LastNm	FirstNm
5032	0001	1018	Reynolds	Jane
5032	0002	1020	Davidson	Evan
5032	0003	1031	Green	Jason
5033	0001	1014	Jacobs	Paul
5034	0001	1012	Chevas	Jose
5034	0002	1021	Carnet	Jean
:	:	:	:	:

## Physical Data Block Layout

Individual rows are stored in **data blocks**. Note that data blocks in a given table do not have to be the same size. They can range from 1 to 256 sectors in length. However, there is a maximum block size for multi-row blocks. It is system selectable and may have any value up to 127 sectors (the largest blocksize the file system supports).

Within the data block, rows are arranged in ascending Row-ID order. Rows never span across more than one block. A row that is greater than the specified multi-row blocksize occupies a complete block of its own. Rows are by definition variable length.

The block contains an index array pointing to each row in the block sequence. This improves performance of very large blocks by allowing a binary search of the array to locate the row.

Any changes to data block size require the allocation of a new block, a write to the new block, and the freeing of the old block to the Free Block List.

Advantages to the Teradata blocking strategy are:

- Unused space in the block never exceeds 511 bytes.
- Data blocks are never chained so there can never be broken pointer chains.
- Traditional reorganization (due to block split fragmentation) is never needed.

## Physical Data Block Layout

789	Block Header	Row <sub>1</sub>	Row <sub>2</sub>
790	Row <sub>2</sub>	Row <sub>3</sub>	
791	Row <sub>4</sub>		Row <sub>5</sub>
792	Row <sub>5</sub>	Row <sub>6</sub>	
793	Row <sub>7</sub>		Row <sub>8</sub>
794	Row <sub>9</sub>	Pointer Array	Block Trailer

### Block size

- Size is limited to 127 sectors (65024 bytes).
- Maximum block size is a user-tunable attribute.

### Rows

- Are maintained in Row-ID sequence by the pointer array.
- May never span blocks.
- Are variable length by definition.
- May be up to 64 KB in length

Inserts, updates and deletes may require a new block.

- The new block, of appropriate size, will be allocated.
- The old block plus the changes are written to the new block.
- The old block is put on the free block list.

Teradata blocking strategy advantages:

- Unused space in a block never exceeds 511 bytes.
- There are no corruptible pointers to other blocks.
- Reorganization due to block split fragmentation is NEVER NEEDED!

## Physical Row Layout

The diagram on the facing page shows the **physical row layout** within a data block.

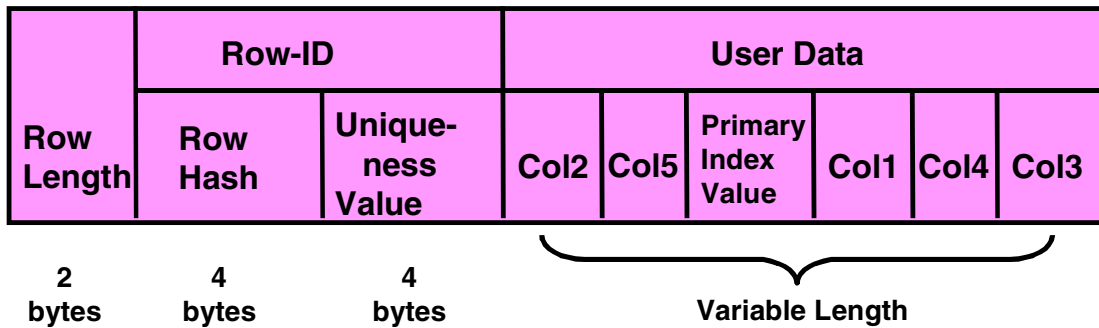
The two bytes the beginning of the row reflect the length of the row.

All rows have a unique Row-ID that is composed of the row hash and the uniqueness value.

Normally, only the row hash portion of the Row-ID is used to locate the row. An exception occurs during Secondary Index access.

Within the User Data columns is the primary index value on which the row hash is based. The PI may represent one or several columns of data. Teradata determines the sequence of columns within the row to optimize performance.

## Physical Row Layout



- Each row has a unique Row-ID within a table.
  - The uniqueness value is appended at insert time.
  - Only secondary indexes use the complete Row-ID to find a row.
- Row length is specified at the beginning of each row.
- Data columns are physically sequenced for optimal storage.
- The PI value is just one or more columns of data in the row.

## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.

## Module Review Questions

Fill in the blanks. (May be more than one word.)

1. To identify the location of a row on an AMP, the system uses the primary index \_\_\_\_\_ and \_\_\_\_\_.
2. The output of the hashing algorithm is called the \_\_\_\_\_.
3. To determine the target AMP, the BYNET must look up an entry in the Hash Map based on the \_\_\_\_\_ number.
4. Two different PI values which hash to the same value are called hash \_\_\_\_\_.
5. A Row-ID consists of a row hash plus a \_\_\_\_\_ value.
6. A uniqueness value is required to produce a unique Row-ID because of \_\_\_\_\_ and \_\_\_\_\_.
7. Rows in a data block are maintained in \_\_\_\_\_ sequence by the \_\_\_\_\_.
8. Because a new block is always allocated for any insert, update, or delete operation, \_\_\_\_\_ is never needed.
9. Rows are, by definition, of \_\_\_\_\_ length in a block.
10. Hash maps are configured base on the \_\_\_\_\_ of a system.

# Notes



### **Secondary Indexes and Table Scans**

**After completing this module, you will be able to:**

- **Define secondary indexes.**
- **Distinguish between the implementation of unique and non-unique secondary indexes.**
- **Define full table scans and describe their causes.**
- **Describe the operation of a full-table scan in a parallel environment.**

## Notes

*Attention Instructors! Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*

## Table of Contents

Secondary Indexes.....	4
Choosing a Secondary Index.....	6
Unique Secondary Index (USI) Access .....	8
Non-Unique Secondary Index (NUSI) Access .....	10
Comparison of Primary and Secondary Indexes .....	12
Full-Table Scans.....	14
Review Questions.....	16
Reference .....	18

# Secondary Indexes

*Topics reflected on Certification Exam.*

Primary index access to data is very efficient—involving one AMP and one row.

A **secondary index** is an alternate path to the data. The use of secondary indexes allows the user to avoid scanning the entire table during a query thereby reducing the impact on system performance.

A secondary index is like a primary index in that it allows the user to locate rows. Unlike a primary index, it has no influence on the way rows are distributed among AMPs. A database designer typically chooses a secondary index because it provides fast set selection.

Primary index requests require only one AMP to access rows, while secondary indexes require at least two and possibly all AMPs, depending on the index and the type of operation. A secondary index search will typically be less expensive than a full table scan.

Secondary indexes add overhead to the table, both in terms of disk space and maintenance, but they may be added or dropped when needed.

## Secondary Indexes

There are three general ways to access a table:

- Primary index access      (*one*-AMP access)
- Secondary index access   (*two*-or *all*-AMP access)
- Full Table Scan            (*all*-AMP access)

- A secondary index is an alternate path to the rows of a table.
- A table can have from 0 to 32 secondary indexes.
- Secondary indexes:
  - Do not affect table distribution.
  - Add overhead, both in terms of disk space and maintenance.
  - May be added or dropped dynamically as needed.
  - Are chosen to improve table performance.

# Choosing a Secondary Index

*Topics reflected on Certification Exam.*

As with primary indexes, there are two types of **secondary indexes**—**unique (USI)** and **non-unique (NUSI)**.

A secondary index may be specified at table creation or at any time during the life of the table. It may consist of up to 16 columns. To get the benefit of the index, the query has to specify a value for all columns in the secondary index.

A **unique secondary index (USI)** has two possible purposes:

- It can **speed up access to a row** which otherwise might require a full table scan, without having to rely on the primary index.
- It can **enforce uniqueness on a column or set of columns**. This is sometimes used when a primary key is not designated as the primary index. **Making it a USI enforces the uniqueness of the PK.**

A **non-unique secondary index (NUSI)** is usually specified in order to **prevent full-table scans**. NUSIs, however, do activate all AMPs because the value being sought might reside on many different AMPs (only primary indexes have same values on same AMPs). If the Optimizer decides that the cost of using the secondary index is greater than a table scan would be, it opts for the table scan.

As column values change, secondary indexes cause an AMP-local subtable to be built and maintained. Secondary index **subtables** consist of rows which associate the secondary index value with one or more rows in the base table. When the index is dropped, the subtable is physically removed.

## Choosing a Secondary Index

A secondary index may be defined:

- At table creation (CREATE TABLE)
- Following table creation (CREATE INDEX)
- Using up to 16 columns

### USI

- If the index choice of column(s) is unique, it is called a USI (unique secondary index).
- Accessing a row via a USI **requires 2 AMPs**.

### NUSI

- If the index choice of column(s) is non-unique, it is called a NUSI (non-unique secondary index).
- Accessing a row via a NUSI requires **all AMPs**.

### USI

**CREATE UNIQUE INDEX  
(employee-number) on  
employee**

### NUSI

**CREATE INDEX (last-name)  
on employee**

### Note:

- Secondary indexes cause an internal sub-table to be built.
- Dropping the index causes the sub-table to be deleted.

## Unique Secondary Index (USI) Access

The facing page shows the two-AMP accesses necessary to retrieve a row via a **unique secondary index (USI)** access.

- After the row hash of the secondary index value is calculated, the hash map points us to AMP 2, containing the subtable row for this USI value.
- After locating the subtable row in AMP 2, we find the Row-ID of the base row. This base Row-ID (which includes the row hash) allows the hash map to point us to AMP 4 which contains the base row.

Secondary index access uses the complete Row-ID to locate the row, whereas primary index access uses only the row hash portion.

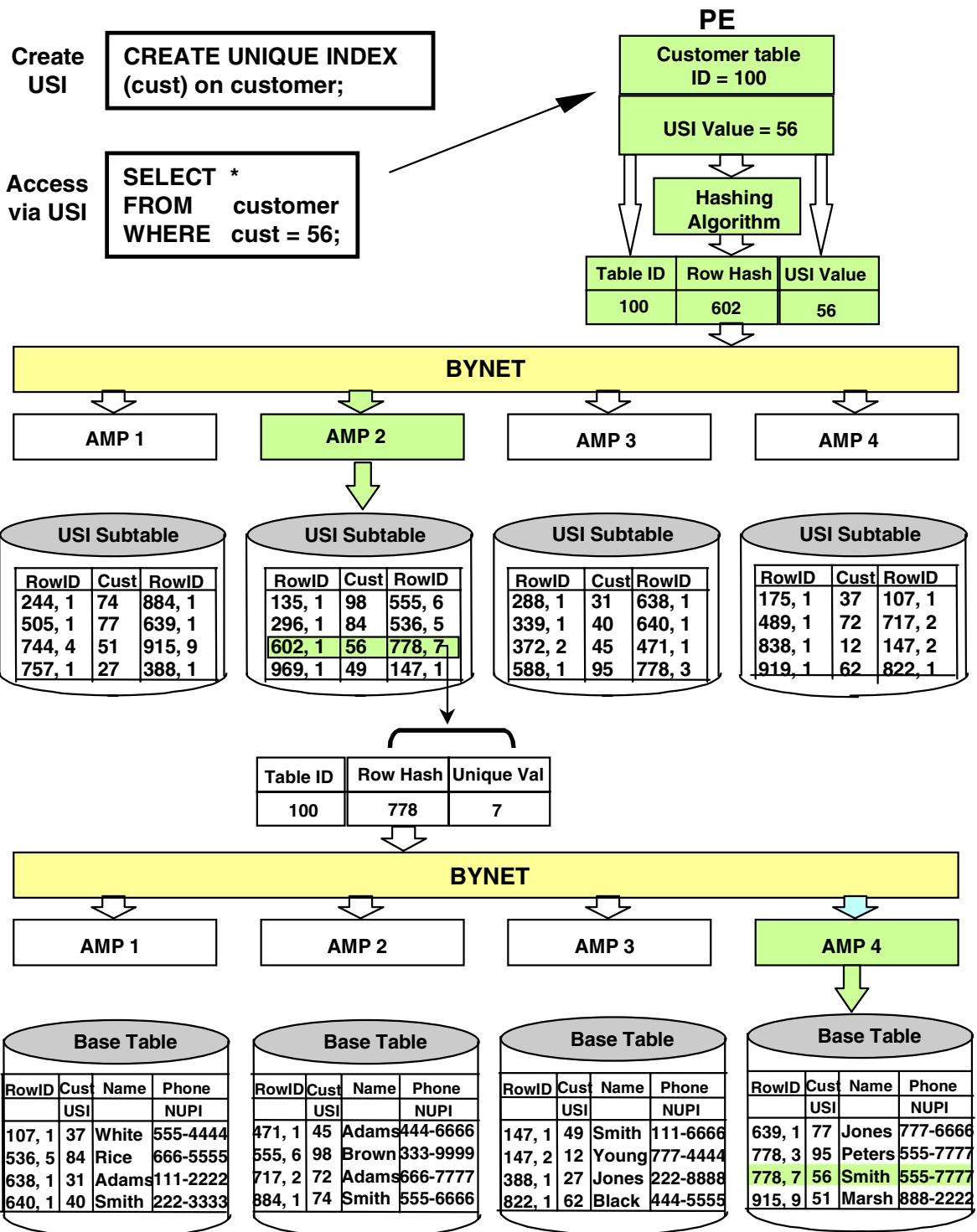
The Customer table below is used in the example on the facing page. It displays only a partial listing of the rows.

**CUSTOMER table**

Cust	Name	Phone
PK		
USI		NUPI
37	White	555-4444
98	Brown	333-9999
74	Smith	555-6666
95	Peters	555-7777
27	Jones	222-8888
56	Smith	555-7777
45	Adams	444-6666
84	Rice	666-5555
49	Smith	111-6666
51	Marsh	888-2222
31	Adams	111-2222
62	Black	444-5555
12	Young	777-4444
77	Jones	777-6666
72	Adams	666-7777
40	Smith	222-3333



# Unique Secondary Index (USI) Access



## Non-Unique Secondary Index (NUSI) Access

The facing page shows an all-AMP access necessary to retrieve a row via a **non-unique secondary index** access.

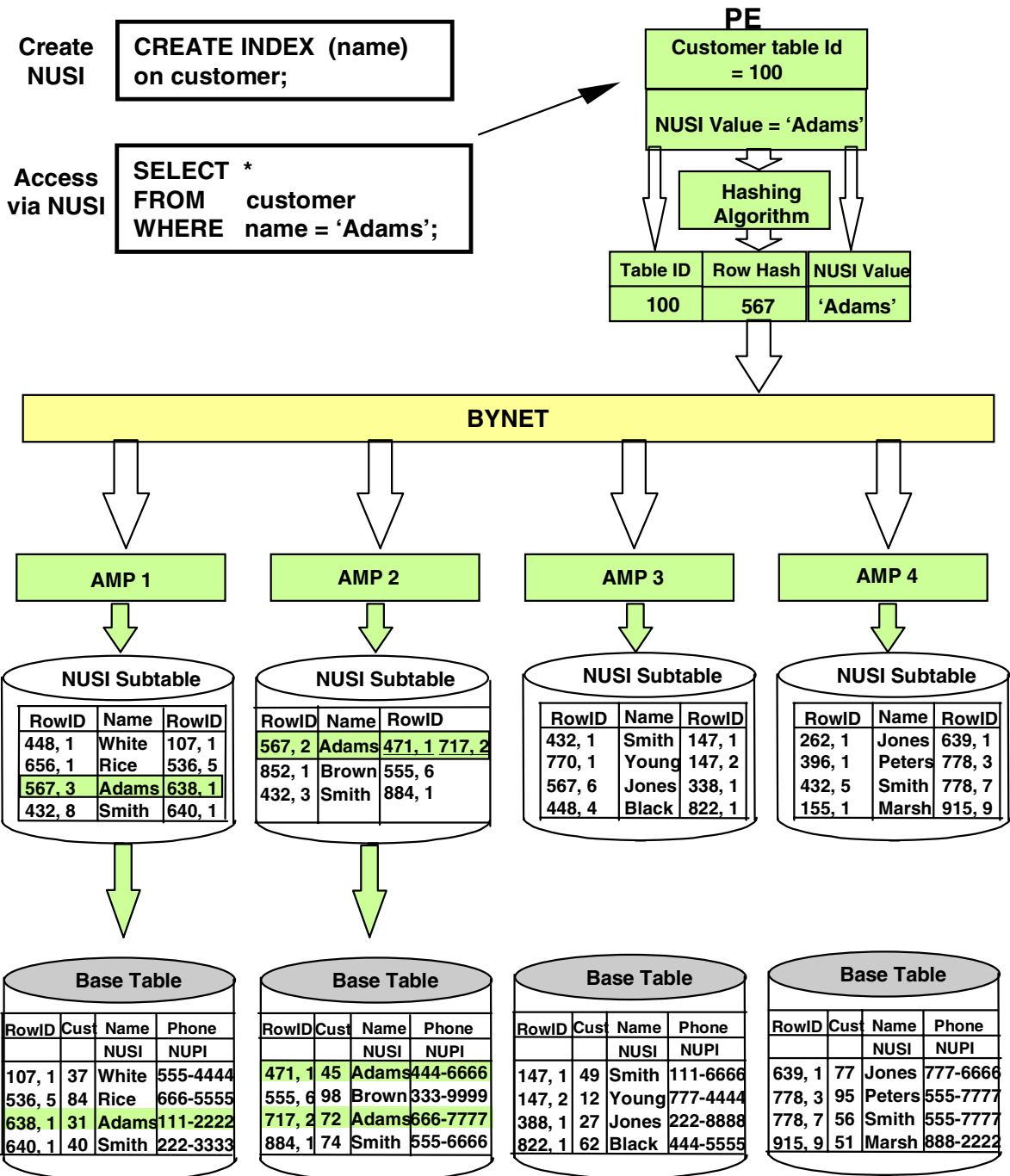
After the row hash of the secondary index value is calculated, the BYNET will automatically activate all AMPs per the Parsing Engine's instructions. Each AMP locates the subtable rows containing the qualifying value and row hash. These subtable rows contain the Row-ID(s) for the base rows, which are guaranteed to be on the same AMP as the subtable row. This reduces activity in the BYNET and essentially makes the query an AMP-local operation.

Because each AMP may have more than one qualifying row, it is possible for the subtable row to have more than Row-ID for the base table rows.

In the Customer table example below, there is only a partial listing of the rows.

CUSTOMER table		
Cust	Name	Phone
PK		
	NUSI	NUPI
37	White	555-4444
98	Brown	333-9999
74	Smith	555-6666
95	Peters	555-7777
27	Jones	222-8888
56	Smith	555-7777
→ 45	Adams	444-6666
84	Rice	666-5555
49	Smith	111-6666
51	Marsh	888-2222
→ 31	Adams	111-2222
62	Black	444-5555
12	Young	777-4444
77	Jones	777-6666
→ 72	Adams	666-7777
40	Smith	222-3333

# Non-Unique Secondary Index (NUSI) Access



# Comparison of Primary and Secondary Indexes

The table on the facing page compares **primary and secondary indexes**:

- **Primary indexes are required; secondary indexes are optional.** All tables must have a method of distributing rows among AMPs—the Primary Index.
- **A table can have only one primary index, but up to 32 secondary indexes.**
- **Both primary and secondary indexes can have up to 16 columns.** Secondary indexes, like primary indexes, can be either unique (USI) or non-unique (NUSI).
- The secondary index does not affect the distribution of rows. Rows are distributed according to the primary index values.
- **Secondary indexes can be added and dropped dynamically as needed.** In some cases, it is a good idea to wait and see how the database is used and then add secondary indexes to facilitate that usage.
- **Both primary and secondary indexes affect system performance for different reasons.** A poorly chosen PI results in lumpy data distribution, which causes some AMPs to do more work than others, and slows the system.
- **Secondary indexes affect performance because they require subtables.**
- Both primary and secondary indexes allow rapid retrieval of specific rows.
- Both primary and secondary indexes can be created using multiple data types.
- **Secondary indexes are stored in separate subtables; primary indexes are not.**
- Because secondary indexes require separate subtables, extra processing overhead is needed to maintain those subtables.

## Comparison of Primary and Secondary Indexes

Index Feature	Primary	Secondary
Required?	Yes	No
Number per Table	1	0-32
Max Number of Columns	16	16
Unique or Non-Unique?	Both	Both
Affects Row Distribution	Yes	No
Created/Dropped Dynamically	No	Yes
Improves Access	Yes	Yes
Multiple Data Types	Yes	Yes
Separate Physical Structure	None	Sub-table
Extra Processing Overhead	No	Yes

# Full-Table Scans

*Topics reflected on Certification Exam.*

A **full-table scan** is another way to access data without using any primary or secondary indexes.

In evaluating an SQL request, the Parser examines all possible access methods and chooses the one it believes to be the most efficient. The coding of the SQL request, along with the demographics of the table and the availability of indexes, all play a role in the decision of the Parser.

Some coding constructs, listed on the facing page, always cause a full-table scan. In other cases, a full-table scan might be chosen because it is the most efficient method. If the number of physical reads exceeds the number of data blocks, then the Parser may decide that a full-table scan is faster.

With a full-table scan, each data block is found using the Master and Cylinder Indexes and each data row is accessed only once.

As long as the choice of primary index has caused the table rows to distribute evenly across all of the AMPs, the parallel processing of the AMPs can do the full-table scan quickly. The file system stores each table on as few cylinders as possible to help reduce the cost of full-table scans.

While full-table scans are impractical and even disallowed on some systems, Teradata routinely permits ad hoc queries with full-table scans.

## Full-Table Scans

- **Every row of the table must be read.**
  - Each row is accessed only once.
  - All AMPs scan their portion of the table in parallel.
  - If the Primary Index caused even distribution, the full-table scan can be very effective.
- **Full-table scans typically occur when either:**
  - An index is not used in the query.
  - An index is used in a non-equality test.

**CUSTOMER**

Cust_ID	Cust_Name	Cust_Phone
USI	NUSI	NUPI

### Examples of Full-Table Scans:

```
SELECT * FROM customer WHERE Cust_Phone LIKE  
'524-____';
```

```
SELECT * FROM customer WHERE Cust_Name <>  
'Davis';
```

```
SELECT * FROM customer WHERE Cust_ID > 1000;
```

## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.



## Review Questions

Fill each box with either Yes, No, or the appropriate number.

	USI Access	NUSI Access	FTS
# AMPs			
# rows			
Parallel Operation			
Uses Separate Sub-table			
Reads all data blocks of table			

Indicate whether a statement is True or False.

1. A query using a secondary index requires at least 2 AMPs.
2. A NUSI avoids a full-table scan by limiting the number of AMPs accessed.
3. A USI can be used to enforce uniqueness on a Primary Key column.
4. You can create or drop USIs and NUSIs at any time.
5. An example of the syntax to create a NUSI is:  
`CREATE NUSI (last_name) on CUSTOMER ;`
6. A full-table scan is never efficient because it accesses rows multiple times.
7. A full-table scan can occur when there is a range of values specified for columns in a primary index.

## Reference

For more information on storing and accessing data rows, refer to *Teradata RDBMS Database Design*.

### **Data Protection**

**After completing this module, you will be able to:**

- **Explain how locks prevent loss of data integrity.**
- **Explain the concept of FALLBACK tables.**
- **List the types and levels of locking provided by Teradata.**
- **Describe the Recovery, Transient and Permanent Journals and their functions.**
- **List the utilities available for archive and recovery.**

## Notes

*Attention Instructors! Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*

# Table of Contents

Locks .....	4
Rules of Locking .....	6
Access Locks.....	8
Fallback .....	10
Fallback Clustering (1 of 2).....	12
Fallback Clustering (2 of 2).....	14
Fallback vs. Non-Fallback Tables.....	16
Fallback and Disk Arrays .....	18
Recovery Journals for Down AMPs .....	20
Cliques (1 of 2) .....	22
Cliques (2 of 2) .....	24
Cliques and Clusters (1 of 4) .....	26
Cliques and Clusters (2 of 4) .....	28
Cliques and Clusters (3 of 4) .....	30
Cliques and Clusters (4 of 4) .....	32
Transient Journal.....	34
Archiving and Recovering Data .....	36
Permanent Journal .....	38
Using the Permanent Journal for Recovery .....	40
Review Questions.....	42

# Locks

*Topics reflected on Certification Exam.*

Locking prevents multiple users who are trying to change the same data at the same time from violating data integrity. This concurrency control is implemented by **locking** the desired data. Locks are automatically acquired during the processing of a request and released at the termination of the request. In addition, users can specify locks.

There are four types of locks.

## Exclusive

Exclusive locks are applied only to databases or tables, never to rows. They are the most restrictive types of lock; all other users are locked out. Exclusive locks are rarely used (usually only when structural changes are being made to the database or table).

## Write

Write locks enable users to modify data while locking out all other users except readers not concerned about data consistency (access lock readers). Until a write lock is released, no new read or write locks are allowed.

## Read

Read locks are used to ensure consistency during read operations. Several users may hold concurrent read locks on the same data, during which no modification of the data is permitted.

## Access

Users who are not concerned about data consistency can specify access locks. Using an access lock allows for reading data while modifications are in process. Access locks are designed for decision support on large tables that are updated only by small, single-row changes. Access locks are sometimes called stale read locks. You may get stale data that hasn't been updated.

Three levels of database locking are provided:

<b>Database</b>	Locks all objects in the database.
<b>Table</b>	Locks all rows in the table or view.
<b>Row Hash</b>	Locks all rows with the same row hash.

The type and level of locks are automatically chosen based on the type of SQL command issued as shown on the facing page. The user has, in some cases, the ability to upgrade or downgrade the lock.

# Locks

There are four types of locks:

- **Exclusive**—prevents any other type of concurrent access
  - Applies only to **databases or tables, not rows.**
  - Very restrictive
- **Write**—prevents other Read, Write, Exclusive locks.
- **Read**—prevents Write and Exclusive locks.
- **Access**—prevents Exclusive locks only.

Locks may be applied at three database levels:

- **Database**—applies to all tables/views in the database.
- **Table/View**—applies to all rows in the table/views.
- **Row Hash**—applies to all rows with same row hash.

Lock types are automatically applied based on the SQL command:

- **SELECT**—applies a Read lock.
- **UPDATE**—applies a Write lock.
- **CREATE TABLE**—applies an Exclusive lock.

Certain locks can be upgraded or downgraded:

**LOCKING FOR ACCESS SELECT \* FROM TABLE\_A;**

**LOCKING FOR EXCLUSIVE UPDATE TABLE\_B SET A = 2000;**

## Rules of Locking

As the facing page illustrates, a new lock request must wait (queue) behind other incompatible locks that are either in queue or in effect. In example 1, the new read lock must wait until the write lock ahead of it is released before it can go into effect.

In the second example, the second read lock request might occupy the same position in the queue as the read lock already there. When the current write lock is released, both requests may be given access concurrently only if both locks are compatible.

When an SQL statement provides row hash information, a row hash lock will be used. If multiple row hashes within the table are affected, a table lock is used.



# Rules of Locking

LOCK REQUEST	LOCK LEVEL HELD				
	NONE	ACCESS	READ	WRITE	EXCLUSIVE
ACCESS	Granted	Granted	Granted	Granted	Queued
READ	Granted	Granted	Granted	Queued	Queued
WRITE	Granted	Granted	Queued	Queued	Queued
EXCLUSIVE	Granted	Queued	Queued	Queued	Queued

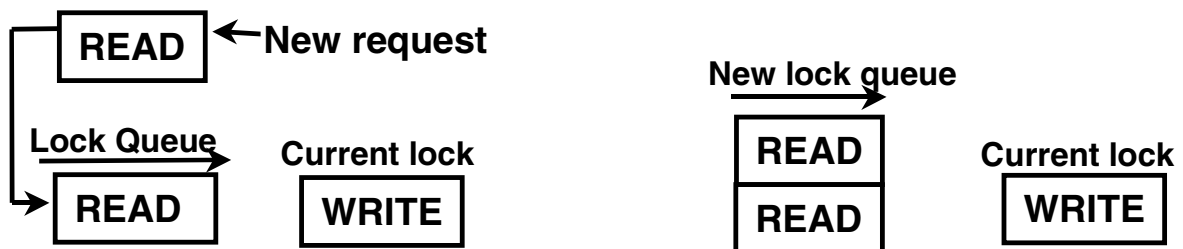
## Rule

Lock requests are queued behind all outstanding incompatible lock requests for the same object.

**Example 1: A new READ lock request goes to the end of queue.**



**Example 2: A new READ lock request shares slot in the queue.**



## Access Locks

**Access locks** have many advantages. They allow quick access to data, even if other requests are updating the data. They also have minimal effect on locking out others—when you use an access lock, virtually all requests are compatible with yours.

When doing large aggregations of numbers, it may be inconsequential if certain rows are updated during the summation, particularly if one is only looking for approximate totals. Access locks are ideal for this situation.

In example 3, what happens to the Write lock request when the Read lock goes away? Look at the chart on the facing page. The request will be granted since Write and Access are considered compatible.

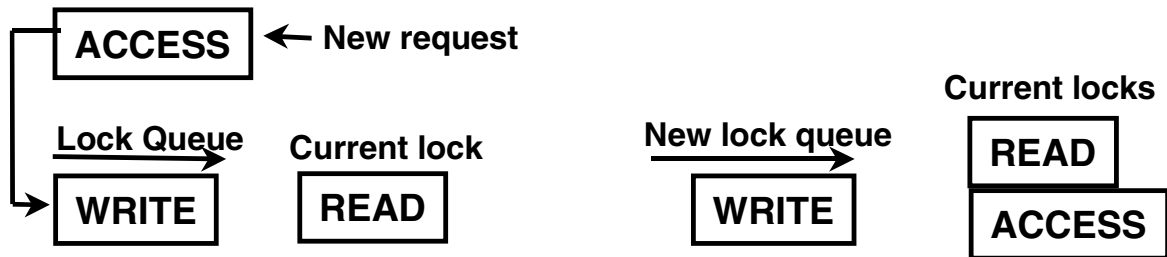
## Access Locks

LOCK REQUEST	LOCK LEVEL HELD				
	NONE	ACCESS	READ	WRITE	EXCLUSIVE
ACCESS	Granted	Granted	Granted	Granted	Queued
READ	Granted	Granted	Granted	Queued	Queued
WRITE	Granted	Granted	Queued	Queued	Queued
EXCLUSIVE	Granted	Queued	Queued	Queued	Queued

### Rule

Lock requests are queued behind all outstanding incompatible lock requests for the same object.

**Example 3:** A new ACCESS lock request is granted immediately.



### Advantages of Access locks:

- Permit quicker access to table in multi-user environment.
- Have minimal blocking effect on other queries.
- Very useful for aggregating large numbers of rows.

### Disadvantages of Access locks:

- May produce erroneous results if performed during table maintenance.

# Fallback

*Topics reflected on Certification Exam.*

**Fallback** protects your data by storing a second copy of each row of a table on an alternate, fallback AMP in the same cluster. If an AMP fails, the system accesses the fallback rows to meet requests. **Fallback provides AMP fault tolerance at the table level.** With fallback tables, if one AMP fails, all table data is still available. Users may continue to use fallback tables without any loss of available data.

During table creation or after a table is created, you may specify whether or not the system should keep a fallback copy. If fallback is specified, it is automatic and transparent.

Fallback guarantees that the two copies of a row will always be on different AMPs. If either AMP fails, the alternate row copy is still available on the other AMP.

There is a benefit to protecting your data, but there are costs associated with that benefit. With fallback use, you need twice the disk space for storage and twice the I/O for INSERTs, UPDATEs, and DELETEs. (The fallback option does not require any extra I/O for SELECT operations and the fallback I/O will be performed in parallel with the primary I/O.)

The benefits of fallback include:

- Protects your data from hardware (disk) failure.
- Protects your data from software (node) failure.
- Automatically recovers with minimum recovery time, after repairs or fixes are complete.

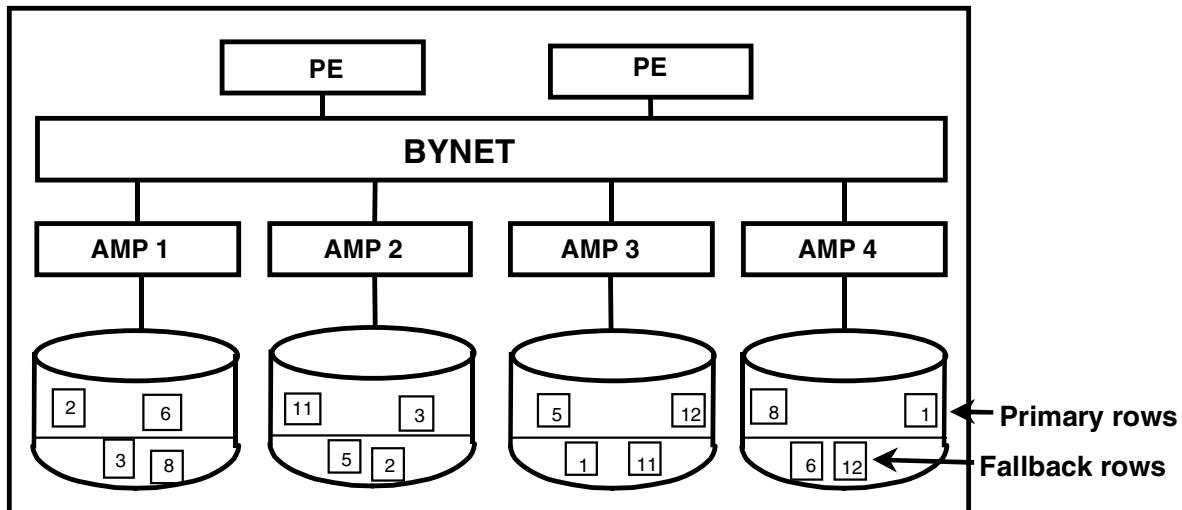
**A hardware (disk, cpu) or software (vproc) failure causes an AMP to be taken off-line until the problem is corrected.**

**During this period, fallback tables are fully available to users.**

**When the AMP is brought back on-line, the associated vdisk is refreshed to reflect any changes during the off-line period.**

# Fallback

- A **fallback table** is fully available in the event of an unavailable AMP.
- A **fallback row** is a copy of a primary row stored on a different AMP.



## Benefits of Fallback

- **Permits access to table data** during AMP off-line period.
- Adds a level of **data protection** beyond disk array RAID.
- **Automatically restores data changed** during AMP off-line.
- **Critical for high availability applications.**

## Cost of Fallback

- Twice the disk space for table storage is needed.
- Twice the I/O for INSERTs, UPDATEs and DELETEs is needed.

**Note: The loss of two AMPs within a cluster will cause the RDBMS to halt!**

## Fallback Clustering (1 of 2)

A fallback **cluster** is a group of AMPs that act as a single fallback unit. Clustering has no effect on primary row distribution of the of a table, but the fallback row copy will always go to another AMP in the same cluster.

Cluster sizes are set through a Teradata console utility and may range from 2 to 16 AMPs per cluster (not all clusters in a system have to be the same size). The example shows an 8-AMP system set up in two clusters of 4-AMPs each.

Should an AMP fail, the primary and fallback row copies stored on that AMP cannot be accessed. However, their alternate copies are available through the other AMPs in the same cluster.

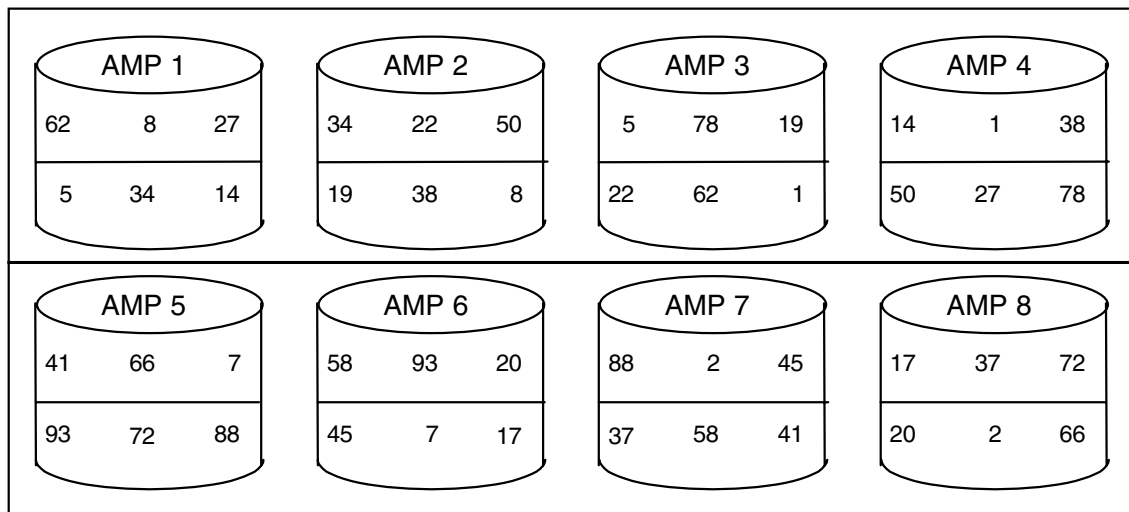
- The **loss** of an AMP in one cluster **has no effect upon other clusters**.
- It is possible to **lose one AMP in each cluster and still have full access to all fallback-protected table data**.
- If there are **two AMP failures** in the same cluster, the entire Teradata system **halts**.

While an AMP is down, the remaining AMPs in the cluster must do their own work plus the work of the down AMP. The larger the size of the cluster, the less noticeable the workload increase is within that cluster when one AMP fails. Large cluster sizes are more vulnerable to a second failure before recovery from the first failure is complete. A second failure halts the entire Teradata system.

## Fallback Clustering (1 of 2)

- A **fallback cluster** is a defined number of AMPs which are treated as a **single, fault-tolerant unit**.
- All fallback rows for AMPs in a cluster must **reside** within the cluster.
- Loss of one AMP in the cluster permits continued table access.
- Loss of two AMPs in the cluster causes the RDBMS to halt.

### Two Clusters of Four AMPs Each



**Lose AMP 3 from cluster → AMPs 1, 2 and 4 experience 33% increase in workload.**

**Lose AMP 6 from cluster → AMPs 5, 7 and 8 experience 33% increase in workload.**

**Lose AMP 7 from cluster → System halts.**

## Fallback Clustering (2 of 2)

The facing page example shows an 8-AMP system with 4 clusters of 2 AMPs each. This system can lose four AMPs (one per cluster) and continue operations. However, if the system workload is near capacity, there will be some loss of performance

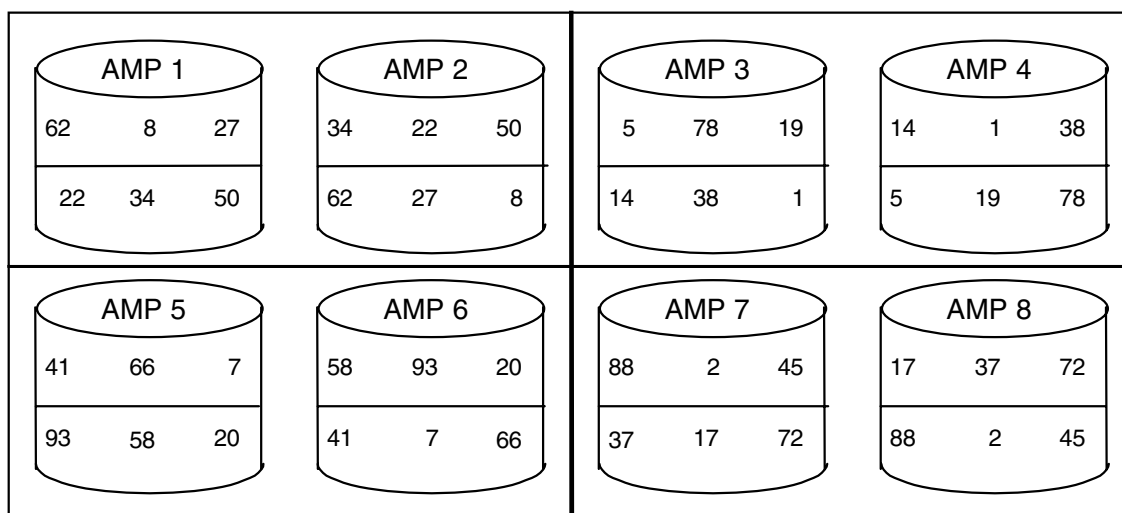
The chart below shows four cluster sizes and the workload increase with a failure. Teradata recommends cluster sizes of 4 AMPs, which minimizes the chances of a second failure, with only a 33% workload increase.

CLUSTER SIZE	WORKLOAD INCREASE	MUST STAY UP
16 AMPs per cluster	1.06	the 15 remaining AMPs in Cluster
8 AMPs per cluster	1.14	the 7 remaining AMPs in Cluster
4 AMPs per cluster	1.33	the 3 remaining AMPs in Cluster
2 AMPs per cluster	2.00	the 1 remaining AMP in Cluster



## Fallback Clustering (2 of 2)

### Four Clusters of Two AMPs Each



**Lose AMP 2 from cluster → AMP 1 experiences 100% increase in workload.**

**Lose AMP 4 from cluster → AMP 3 experiences 100% increase in workload.**

**Lose AMP 5 from cluster → AMP 6 experiences 100% increase in workload.**

**Lose AMP 8 from cluster → AMP 7 experiences 100% increase in workload.**

**Lose AMP 7 from cluster → System halts.**

**Note: System performance can be adversely affected when any AMP has a disproportionate burden.**

# Fallback vs. Non-Fallback Tables

## Fallback tables

Fallback tables have a major advantage in terms of **availability and recoverability**. They can withstand an AMP failure in each cluster and maintain full data availability. **A second AMP failure in any cluster will result in a system halt.** A manual system restart is required in this circumstance.

Fallback tables are easily recovered after a failure due to the availability of fallback rows.

## Non-Fallback tables

**Non-fallback tables are affected by the loss of any one AMP.** The table continues to be accessible, but only for those AMPs still on-line. A one-AMP, primary index access is possible, but a full-table scan is not.

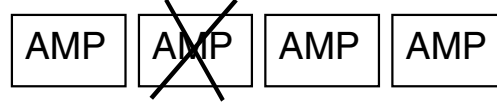
Non-fallback tables may only be restored from external media in the event of a disaster.

## Fallback vs. Non-Fallback Tables

### FALLBACK TABLES

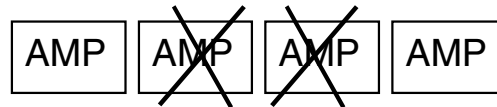
#### ONE AMP DOWN

- Data fully available



#### TWO OR MORE AMPs DOWN

- In different clusters
  - Data fully available
- In the same cluster
  - System halts



### NON-FALLBACK TABLES

#### ONE AMP DOWN

- Data partially available
- Queries avoiding down AMP succeed



#### TWO OR MORE AMPs DOWN

- In different clusters
  - Data partially available
  - Queries avoiding down AMPs succeed
- In the same cluster
  - System halts



## Fallback and Disk Arrays

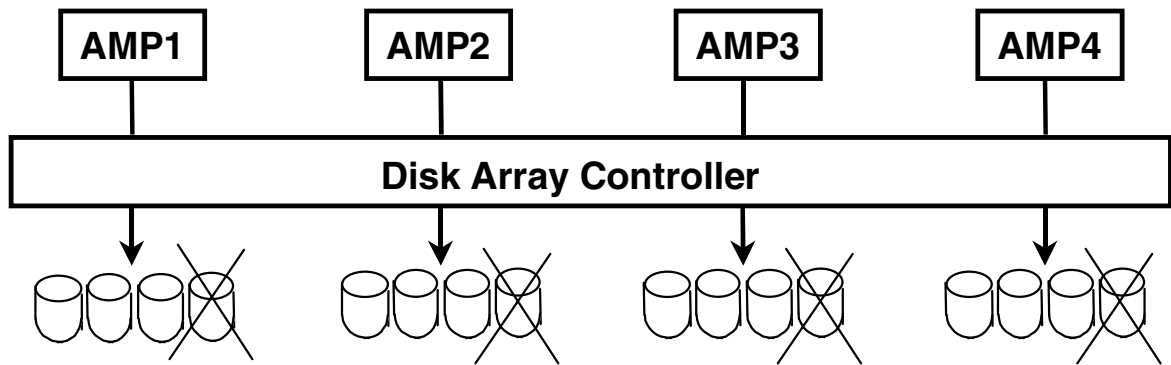
The implementation of **fallback** tables is usually related to the issue of **disk array protection**. With its mirroring feature, RAID 1 protection provides a backup copy of each data row. Fallback adds an additional layer of protection on top of RAID 1, but the cost might be prohibitive for some installations.

**Fallback with RAID 1** offers the highest level of protection and performance available, even protecting against the loss of an entire rank.

**Fallback with RAID 5** is more common, **providing protection against the loss of up to two disks in the same rank**.

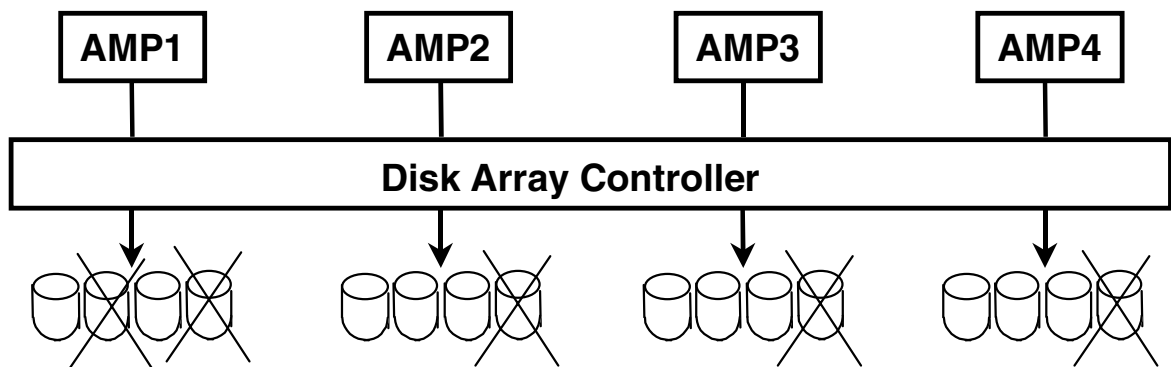
For systems that must operate 24 hours a day, seven days a week, fallback is recommended for minimizing the risk of system downtime.

## Fallback and Disk Arrays



**Lose one disk in any rank:**

<b>RAID 1</b>	<b>Mirror activated</b>	<b>AMP on-line</b>	<b>Fallback unnecessary</b>
<b>RAID 5</b>	<b>Parity activated</b>	<b>AMP on-line</b>	<b>Fallback unnecessary</b>



**Lose two disks in any rank:**

<b>RAID 1</b>	<b>Mirror activated</b>	<b>AMP on-line</b>	<b>Fallback unnecessary</b>
<b>RAID 5</b>	<b>Parity no value</b>	<b>AMP off-line</b>	<b>Fallback necessary!</b>

## Recovery Journals for Down AMPs

After the loss of any AMP, a **down-AMP recovery journal** is started automatically to log any changes to rows which reside on the down AMP. Any inserts, updates, or deletes affecting rows on the down AMP are applied to the fallback copy within the cluster. The AMP that holds the fallback copy logs the Row-ID in its recovery Journal.

This process continues until the down AMP is brought back on-line. As part of restart activity, the recovery journal is read and changed rows are applied to the recovered AMP. When the journal has been exhausted, it is discarded and the AMP is brought on-line fully recovered.

# Recovery Journal For Down AMPs

Recovery Journal is:

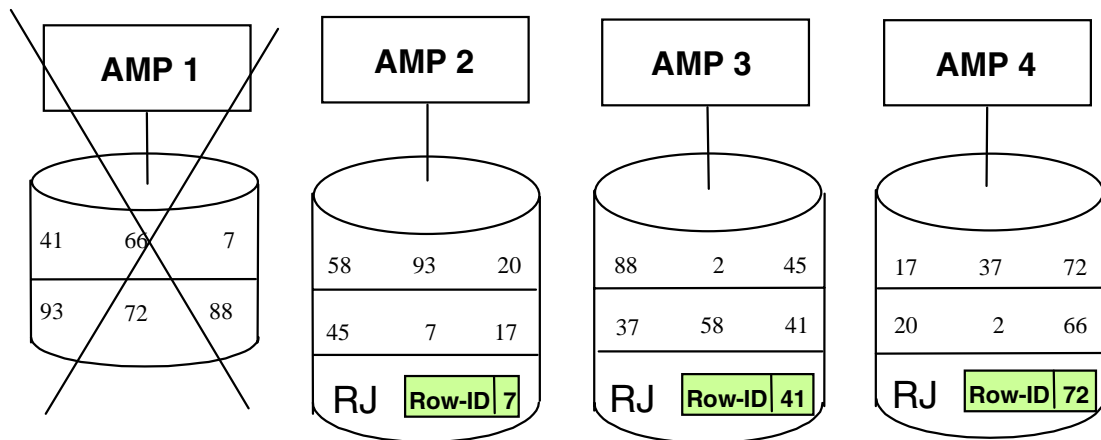
- **Automatically activated** when an AMP is taken off-line.
- **Maintained by other AMPs in the fallback cluster.**
- Totally transparent to users of the system.

While AMP is off-line:

- Journal is active.
- Table updates continue as normal.
- Journal logs Row-IDs of changed rows for down-AMP.

When AMP is back on-line:

- Restores rows on recovered AMP to current status.
- Journal discarded when recovery complete.



## **Cliques (1 of 2)**

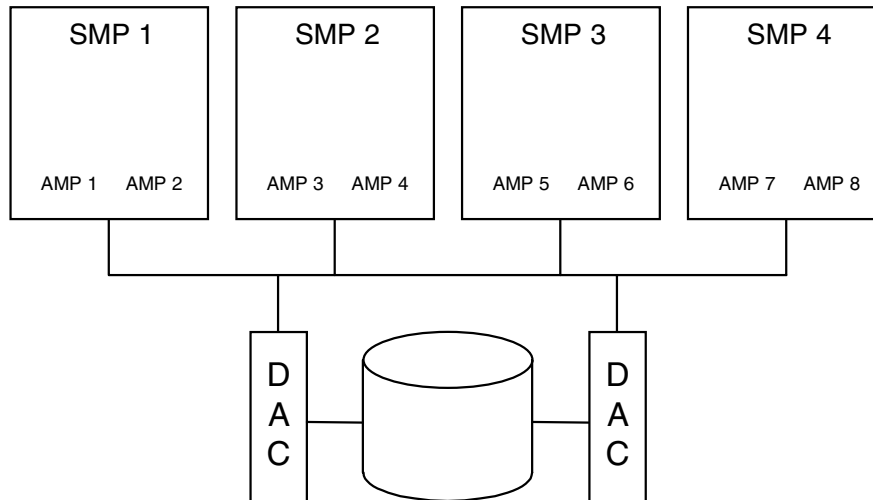
The facing page displays an example of a clique (available for UNIX systems).

A clique can include two to eight nodes. Generally, it is recommended that there be four nodes per clique.

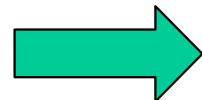


## Cliques (1 of 2)

**Two or more TPA nodes having access to the same disk arrays are called a clique.**



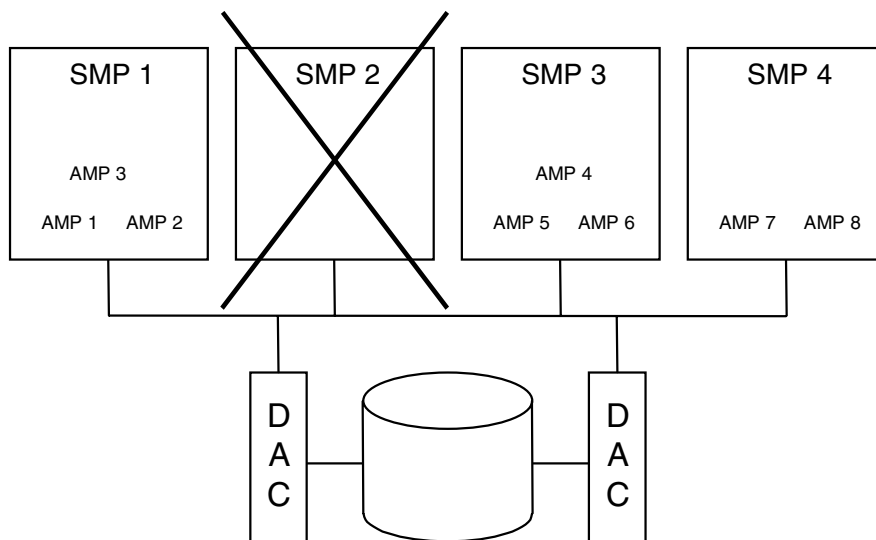
**AMP vprocs can run on any node within the clique and still have full access to their disk array space.**



## Cliques (2 of 2)

In a clique, multiple nodes are accessible by any node in the same clique. The facing page shows a situation where a node fails. A Massively Parallel Processing (MPP) system will continue to run if a node or disk fails in a clique, even if you do not select the fallback option. In the situation shown, the system will restart the lost AMP vprocs on a different node within the clique. When the node is brought back into service, the system will move the AMPs back to their original node. For 7x24 systems, the fallback option is recommended for minimizing the risks of system downtime.

## Cliques (2 of 2)

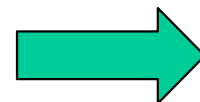
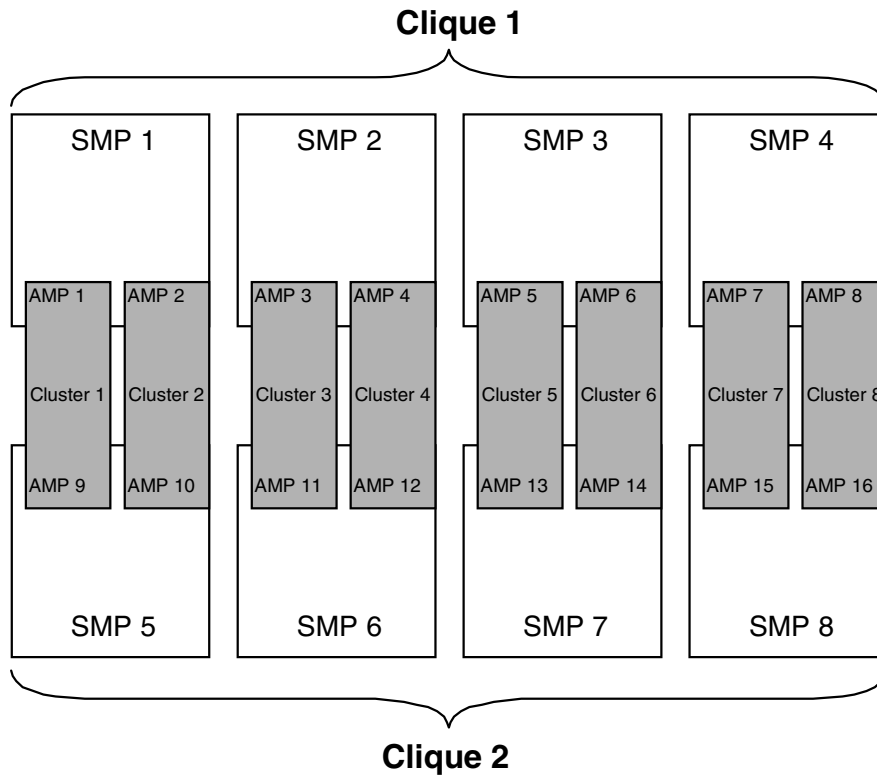


## Cliques and Clusters (1 of 4)

The facing page shows an eight-node system with two AMPs per node and eight clusters defined across two clique boundaries.

- The system can lose one AMP per cluster and still provide full access to all data.
- AMPs can migrate within a clique, but not across clique boundaries.

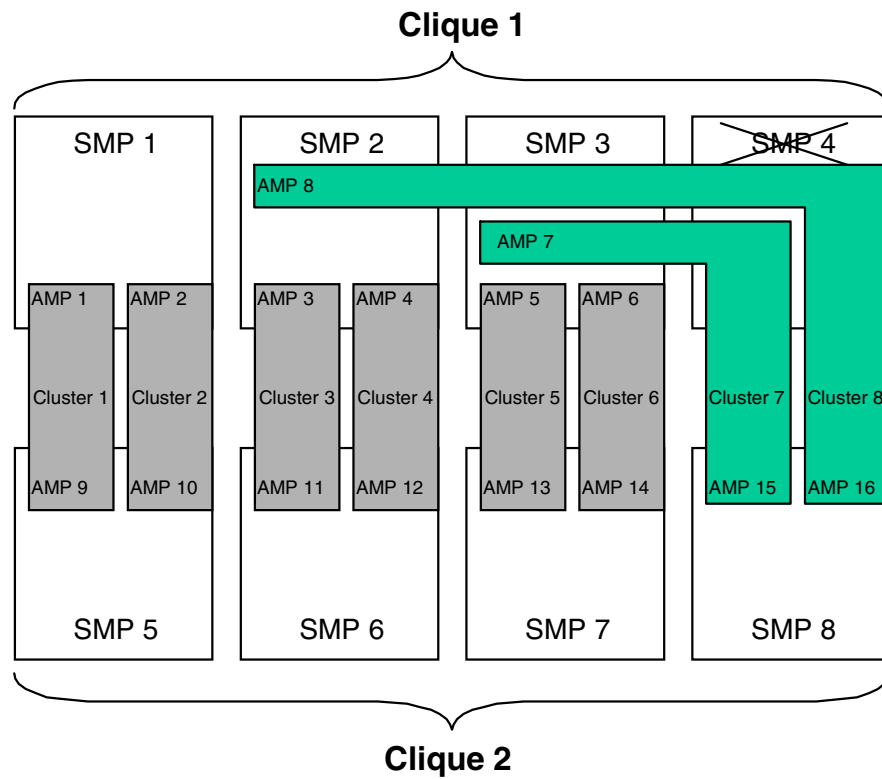
## Cliques and Clusters (1 of 4)



## **Cliques and Clusters (2 of 4)**

If a node should fail when the AMPs are restarted, all of the clusters remain intact. The clique makes all data available. Fallback plays no role in this example.

## Cliques and Clusters (2 of 4)



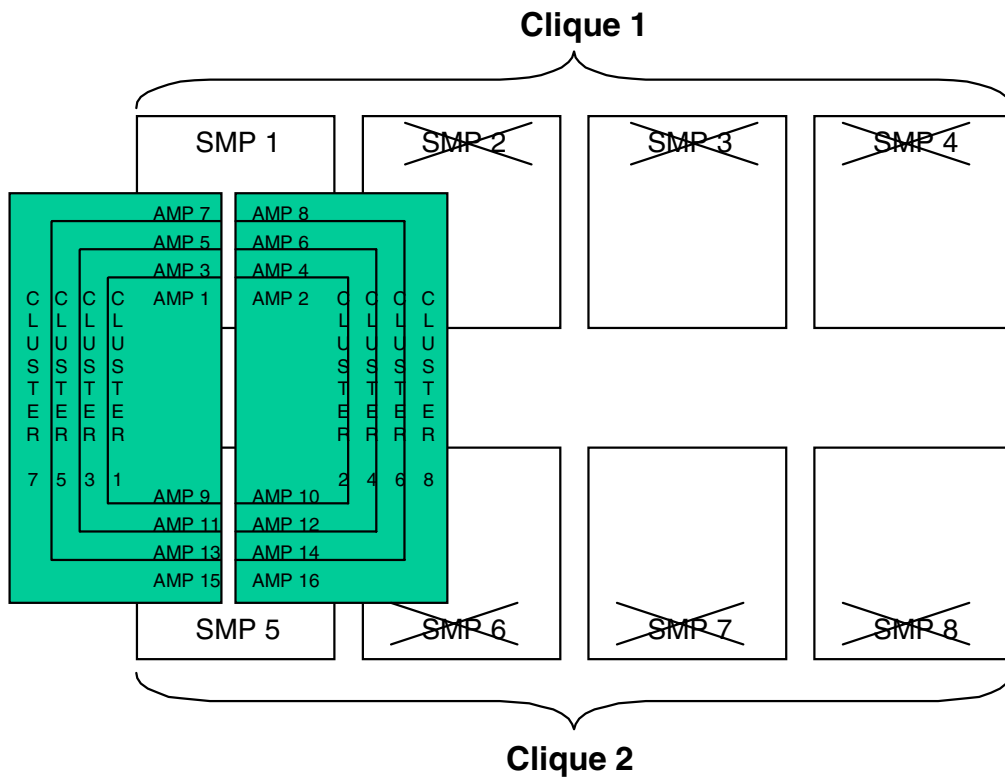
## **Cliques and Clusters (3 of 4)**

The facing page shows that even if three nodes in each clique fail, all clusters remain intact and users still have full access to all of their data, regardless of whether it is fallback-protected or not.

Keep in mind, the system lost six out of eight nodes (75%) and still has full access to all data.



## Cliques and Clusters (3 of 4)

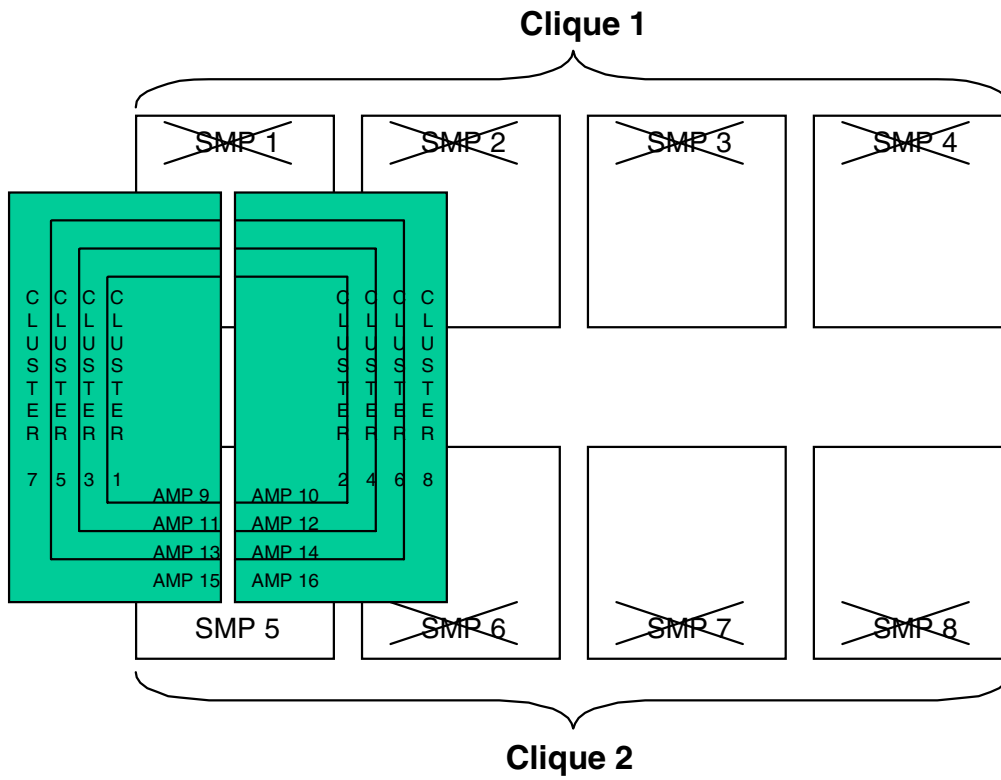


## Cliques and Clusters (4 of 4)

When a system loses all four nodes of a clique, it is considered a single AMP loss in each of the clusters because AMPs cannot migrate across clique boundaries. With the loss of one AMP per cluster, users will have FULL access to all fallback-protected tables. Non-fallback tables will be missing some percentage of their data.

In the example on the facing page, the system has lost seven out of eight nodes and can still provide full access to all fallback data.

## Cliques and Clusters (4 of 4)



# Transient Journal

*Topics reflected on Certification Exam.*

The **transient journal** permits the successful rollback of a failed transaction (TXN). Transactions are not committed to the database until the AMPs have received an End Transaction request, either implicitly or explicitly. There is always the possibility that the transaction may fail. If so, the participating table(s) must be restored to their pre-transaction state.

The transient journal maintains a copy of before images of all rows affected by the transaction. In the event of transaction failure, the before images are reapplied to the affected tables, then are deleted from the journal, and a rollback operation is completed. In the event of transaction success, the before images for the transaction are discarded from the journal at the point of transaction commit.

# Transient Journal

## Transient Journal

- Consists of **a journal of transaction** before images.
- Provides **automatic rollback in the event of TXN failure.**
- **Is automatic and transparent.**
- Before images are reapplied to table if TXN fails.
- Before images are discarded upon TXN completion.

## Successful TXN

<b>BEGIN TRANSACTION</b>		
UPDATE Row A	—	Before image Row A recorded <i>(Add \$100 to checking)</i>
UPDATE Row B	—	Before image Row B recorded <i>(Subtract \$100 from savings)</i>
<b>END TRANSACTION</b>	—	Discard before images

## Failed TXN

<b>BEGIN TRANSACTION</b>		
UPDATE Row A	—	Before image Row A recorded
UPDATE Row B	—	Before image Row B recorded <i>(Failure occurs)</i>
<i>(Rollback occurs)</i>	—	Reapply before images
<i>(Terminate TXN)</i>	—	Discard before images

# Archiving and Recovering Data

## ARC Utility

The ARC utility archives and restores database objects, allowing recovery of data that may have been damaged or lost. There are several scenarios where restoring objects from external media may be necessary:

- Restoring non-fallback tables after a disk failure.
- Restoring tables that have been corrupted by batch processes that may have left the data in an uncertain state.
- Restoring tables, views, or macros that have been accidentally dropped by the user.
- Miscellaneous user errors resulting in damaged or lost database objects.

## ASF2 Utility

The ASF2 utility provides an X-windows-based front-end for creation and execution of ARC command scripts. It is designed to run on UNIX nodes or workstations.

# Archiving and Recovering Data

## **ARC—The Archive/Restore utility**

- **Runs on IBM, Unisys, and UNIX.**
- **Archives data from RDBMS.**
- **Restores data from archive media.**
- **Permits data recovery to a specified checkpoint.**

## **ASF2—The Archive Storage Facility 2**

- **Provides X-Windows front end for ARC.**
- **Allows easy creation of scripts for archive/recovery.**
- **Runs on UNIX nodes.**

## **Common uses of ARC**

- **Restore non-fallback tables after disk failure.**
- **Restore tables after corruption from failed batch processes.**
- **Recover accidentally dropped tables, views, or macros.**
- **Recover from miscellaneous user errors.**

# Permanent Journal

*Topics reflected on Certification Exam.*

The purpose of the **permanent journal** is to provide selective or full database recovery to a specified point in time. It permits recovery from unexpected hardware or software disasters. The permanent journal also reduces the need for full table backups that can be costly in both time and resources.

The permanent journal is an optional journal and its features must be customized to the specific needs of the installation. The journal may capture before images (for rollback), after images (for rollforward), or both. Additionally, the user must specify if single images (default) or dual images (for fault-tolerance) are to be captured.

**Multiple tables or multiple databases may share a permanent journal.**

The journal captures images concurrently with standard table maintenance and query activity. The cost in additional required disk space may be calculated in advance to ensure adequate disk reserve.

The journal is periodically dumped to external media. This reduces the need for full-table backups because only the changes are backed up.



# The Permanent Journal

The *permanent journal* is an optional, user-specified, system-maintained journal used for **database recovery to a specified point in time.**

The permanent journal:

- Is used for recovery from unexpected hardware or software disasters.
- May be specified for:
  - One or more tables
  - One or more databases
- Permits capture of before images for database rollback.
- Permits capture of after images for database rollforward.
- Permits archiving change images during table maintenance.
- Reduces need for full-table backups.
- **Provides a means of recovering NO FALLBACK tables.**
- Requires additional disk space for change images.
- Requires user intervention for archive and recovery activity. It should be periodically purged (dumped to external media).

## Using the Permanent Journal for Recovery

When recovery is initiated using the permanent journal, the question must be posed, recovery to what point in time? Assuming both before and after images have been captured, movement both forward and backward in time is possible.

In the scenario described on the opposite page, a full backup was taken on Monday, and the change images in the permanent journal have been archived on Tuesday and Wednesday. When disaster hits on Thursday morning, a decision is made to restore the database up to the point of the disaster. The restore of Monday's dump is followed by a rollforward of the after images from Tuesday and Wednesday. Thursday's images remain in the journal and are restored directly without the need for external media.

If the failure on Thursday morning had been due to a corrupting application program, a recovery to Wednesday evening might have been chosen. In this case, a rollback operation is initiated, applying all before images captured since Wednesday night.

# Using the Permanent Journal for Recovery

## BEGIN TRANSACTION

UPDATE RowA -----> save before image to PJ

-----> save after image to PJ

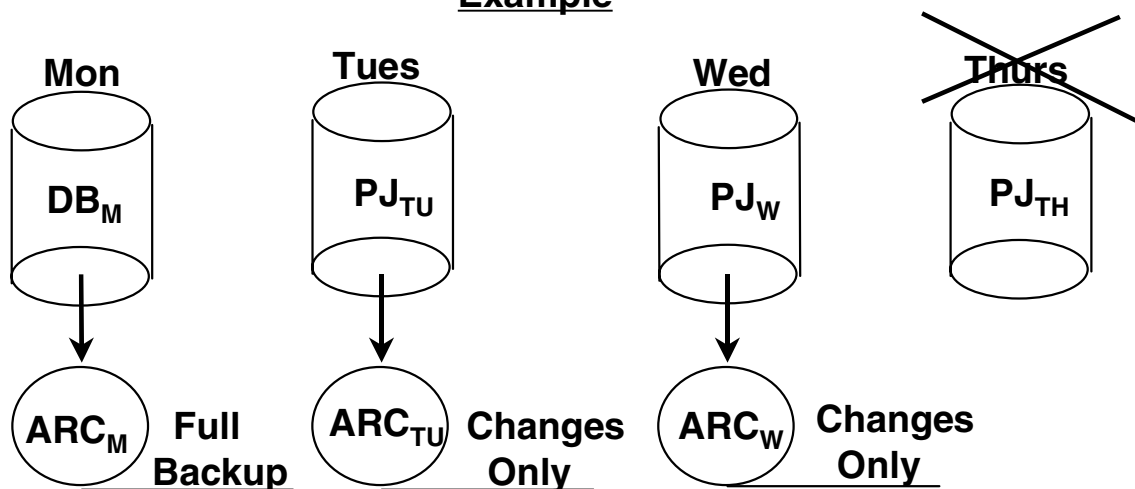
UPDATE RowB -----> save before image to PJ

-----> save after image to PJ

## END TRANSACTION

- *After images* allow a rollforward from a restored point.
- *Before images* allow a rollback from a restored point or an existing point.

### Example



### Failure Thursday at 10 A.M.

- Recover to Monday - Restore  $ARC_M$
- Recover to Tuesday - Restore  $ARC_M + ARC_{TU}$
- Recover to Wednesday - Restore  $ARC_M + ARC_{TU} + ARC_W$
- Recover to Thursday - Restore  $ARC_M + ARC_{TU} + ARC_W + PJ_{TH}$

## **Review Questions**

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.

## Review Questions

Match each item with its definition:

- \_\_\_\_\_ 1. Database lock
- \_\_\_\_\_ 2. Table lock
- \_\_\_\_\_ 3. Row Hash lock
- \_\_\_\_\_ 4. Fallback
- \_\_\_\_\_ 5. Cluster
- \_\_\_\_\_ 6. Recovery journal
- \_\_\_\_\_ 7. Transient journal
- \_\_\_\_\_ 8. ARC
- \_\_\_\_\_ 9. ASF2
- \_\_\_\_\_ 10. Permanent journal
- \_\_\_\_\_ 11. Clique

- a. Provides for TXN rollback in case of failure
- b. X-windows-based archive utility
- c. Protects all rows within
- d. Logs changed rows for down AMP
- e. Provides for recovery to a point in time
- f. Applies to all tables and views within
- g. Multi-platform archive utility
- h. Lowest level of protection granularity
- i. Protects tables from AMP failure by using alternative tables
- j. Fault-tolerant set of AMPs
- k. Nodes sharing disk array

## Review Questions—Continued

## **Review Questions—Continued**

- 1. FALLBACK is recommended if the application requires constant availability.**
- 2. FALLBACK is available only when you use CREATE TABLE.**
- 3. An Exclusive lock applies to rows.**
- 4. With the Transient Journal, before images are discarded only after the transaction is committed, which occurs only after an END transaction request.**
- 5. The Permanent Journal does not require user intervention, as it is automatically purged after a set period.**

## Notes



### **Request Processing in a Parallel Environment**

**After completing this module, you will be able to:**

- **Show how requests are processed in a parallel environment.**
- **Describe the functionality of the BYNET in processing different request types.**

## Notes

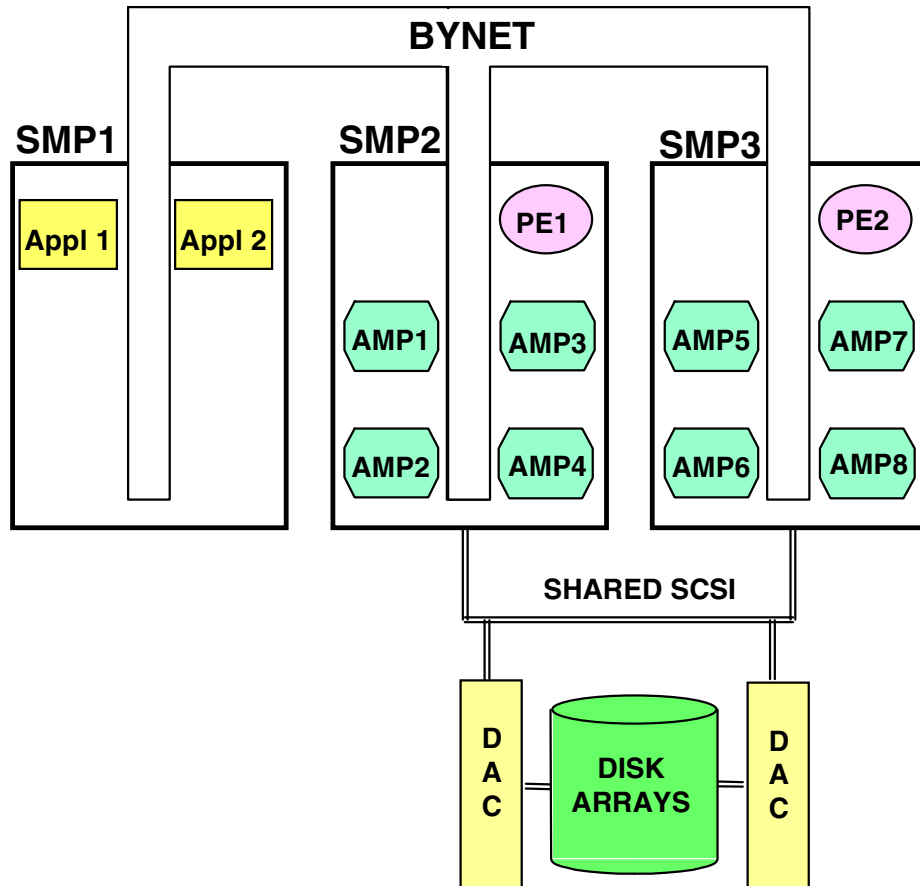
## Table of Contents

Physical View .....	4
Logical View .....	6
AMP Queries .....	8
Single-AMP .....	8
Multi-AMP.....	8
All-AMP request .....	8
Single-AMP Query.....	10
Application to PE.....	12
PE to AMP.....	14
AMP to Vdisk.....	16
AMP to PE.....	18
PE to Application.....	20
All-AMP Query with a Sort.....	22
Application to PE.....	24
PE to AMPs.....	26
PE to AMPs.....	28
AMPs to Merge Process .....	30
AMP to Merge .....	32
PE to Application .....	34
Review Questions.....	36

## Physical View

A physical view of the Teradata environment shows three BYNET-connected nodes. One node contains application programs while the other two contain both PE and AMP vprocs. SMP2 and SMP3 are connected via shared SCSI to the disk array configuration. The disk array has two disk array controllers (DAC). Each is actively used and available to back up the other in case of failure.

## Physical View



**The BYNET handles communications within and between SMP nodes.**

## Logical View

A **logical view** of the Teradata environment shows that all application processes and all vprocs are able to communicate via the BYNET:

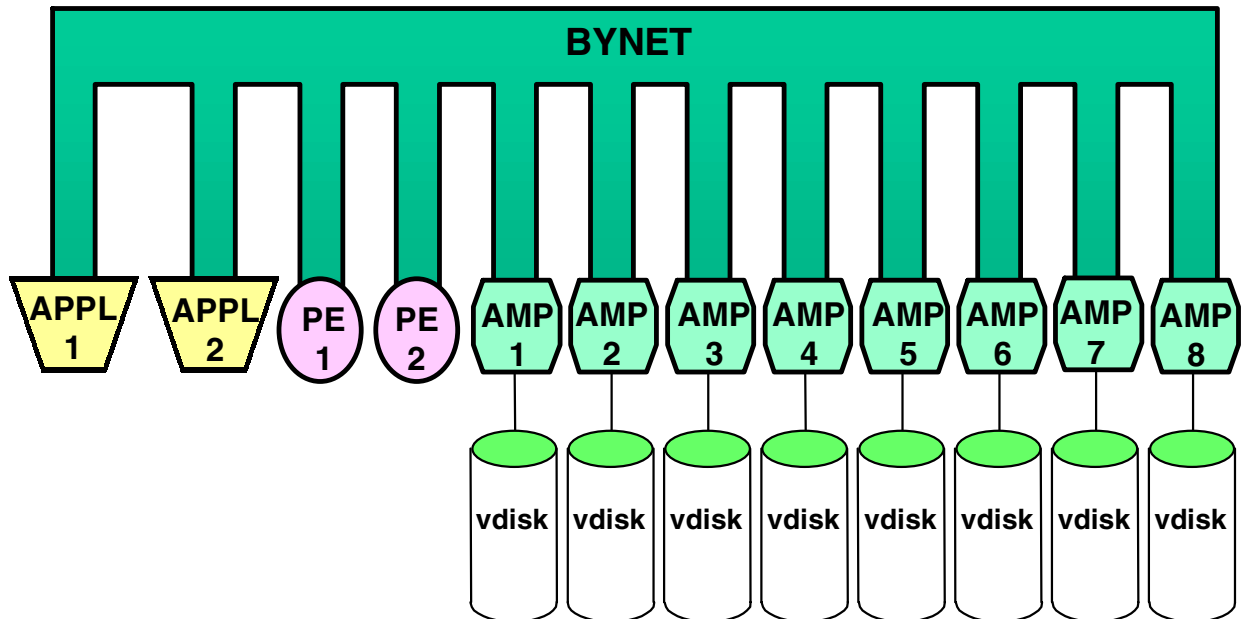
- Applications can talk only to PEs.
- PEs can talk to applications and AMPs.
- AMPs can talk to PEs and other AMPs.

Each AMP has a virtual disk or **vdisk** associated with it via the disk array controller.

The BYNET makes the physical locations of senders and receivers transparent.

Note that vprocs do not directly access the BYNET hardware. Vproc communication is handled by PDE and BYNET software. **The hardware BYNET is a physical transport connecting nodes together.** All of these components are involved in vproc-to-vproc communication across multiple nodes.

## Logical View



**The BYNET makes the physical locations of senders and receivers transparent.**

# AMP Queries

There are three types of requests.

## Single-AMP

A one-AMP or single-AMP request uses the **row hash** to determine which AMP contains the requested rows. An example of a single-AMP request is **primary index retrieval**.

## Multi-AMP

A multi-AMP request may use two or more AMPs. Examples of two-AMP requests are **unique secondary index retrieval**, and **a row inserted to a fallback protected table**.

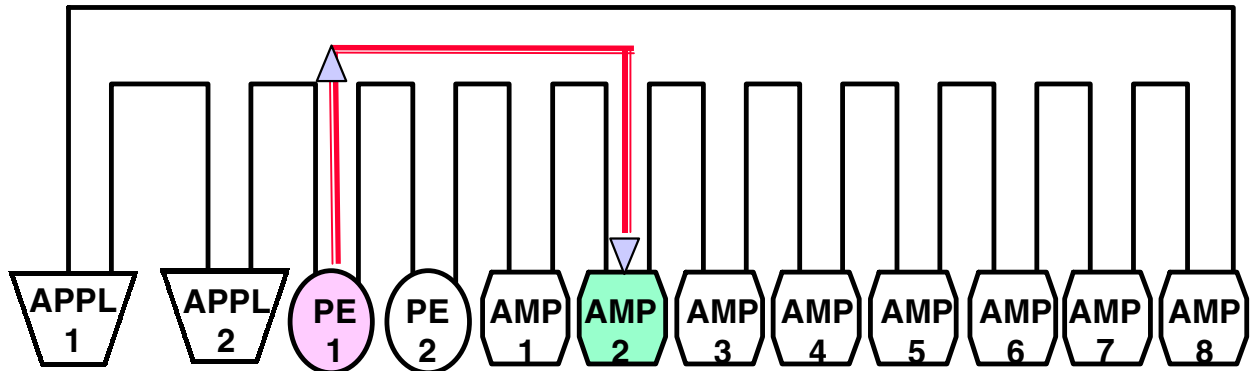
## All-AMP request

When the requested data resides on any AMP, then an all-AMP request is required to obtain it. Examples of all-AMP requests are **full-table scans**, some joins, and all retrievals based on non-unique secondary indexes.

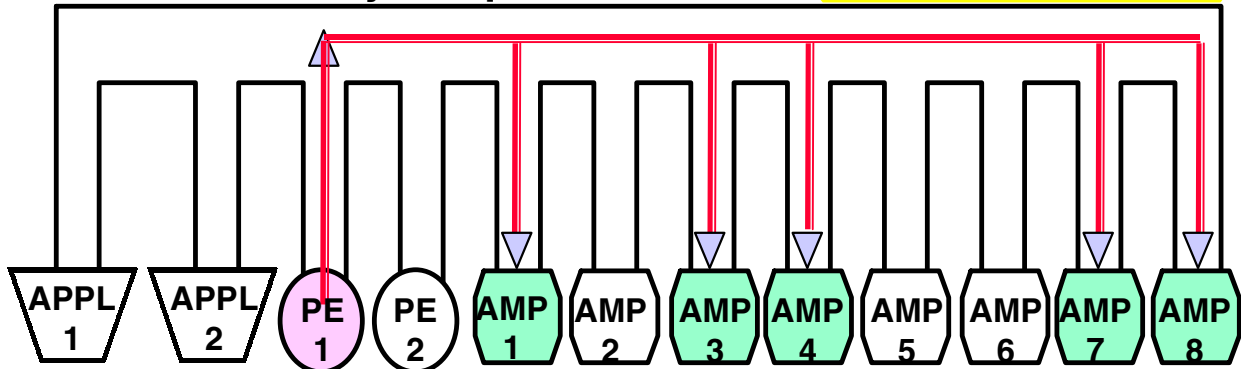


## AMP Queries

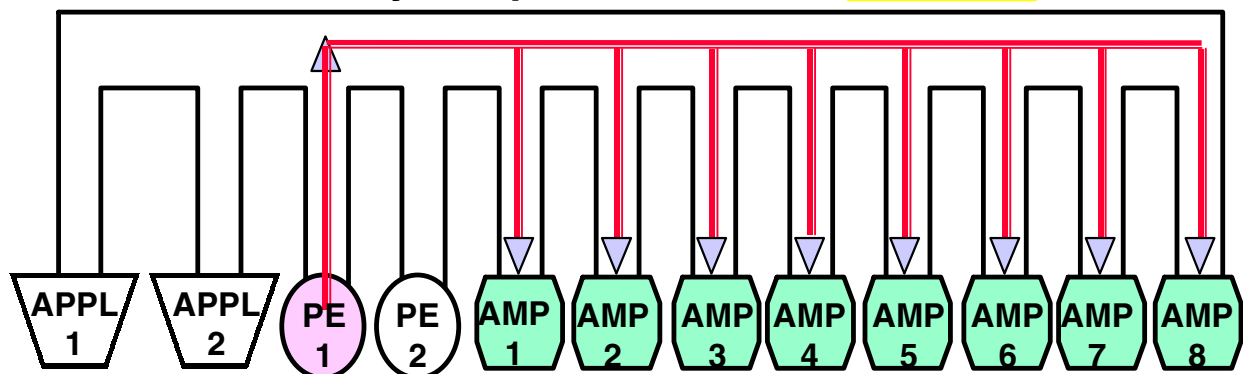
**Single-AMP Query**—requests data from **one AMP only**.



**Multi-AMP Query**—requests data from **more than one AMP**.



**All-AMP Query**—requests data from **all AMPs**.



## Single-AMP Query

The example on the facing page is a one-AMP or single-AMP request because it uses the **Primary Index in the WHERE clause**.

## Single AMP Query

```
SELECT LETTER  
FROM SAMPLE  
WHERE NUMBER = 19  
;
```

ANSWER : N

SAMPLE

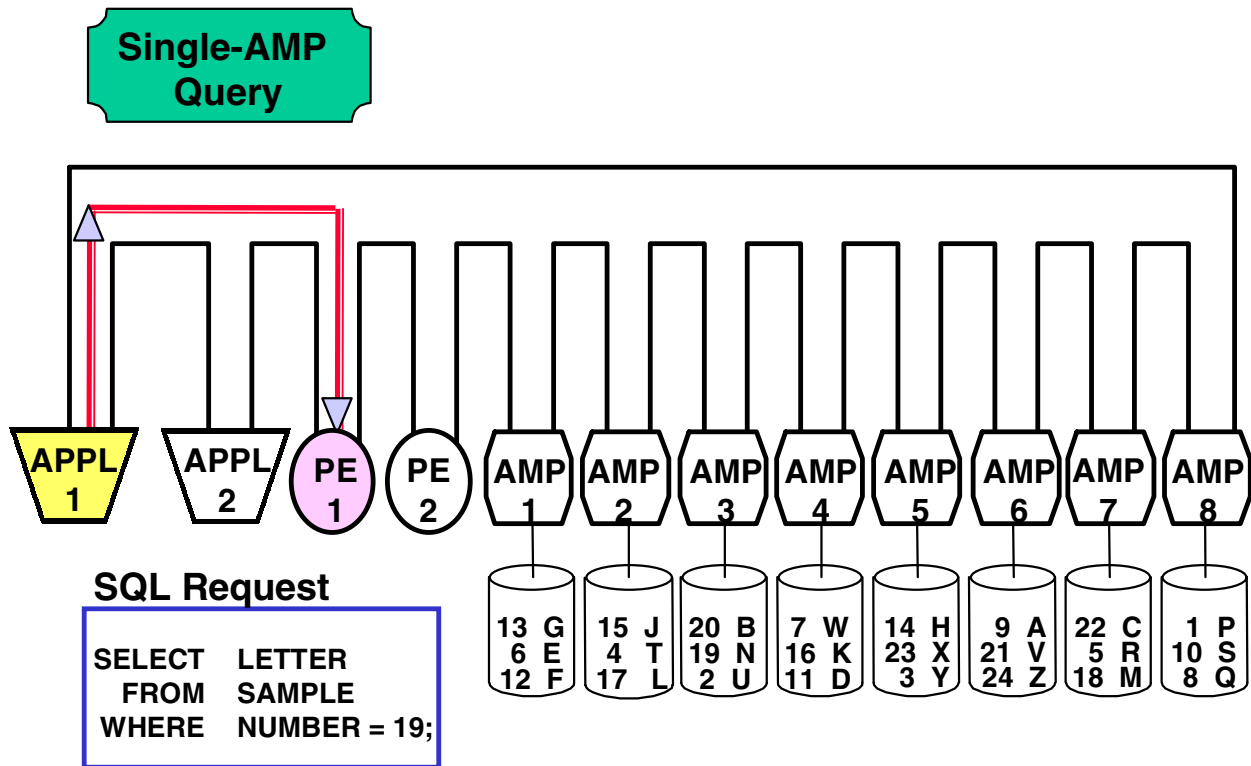
NUMBER	LETTER
UPI	
1	P
2	U
3	Y
4	T
5	R
6	E
7	W
8	Q
9	A
10	S
11	D
12	F
13	G
14	H
15	J
16	K
17	L
18	M
19	N
20	B
21	V
22	C
23	X
24	Z



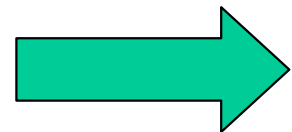
## **Application to PE**

The application sends the request to the PE of the user's session on the forward channel. The PE acknowledges the message across the back channel. The PE then parses and optimizes the request.

## Application to PE



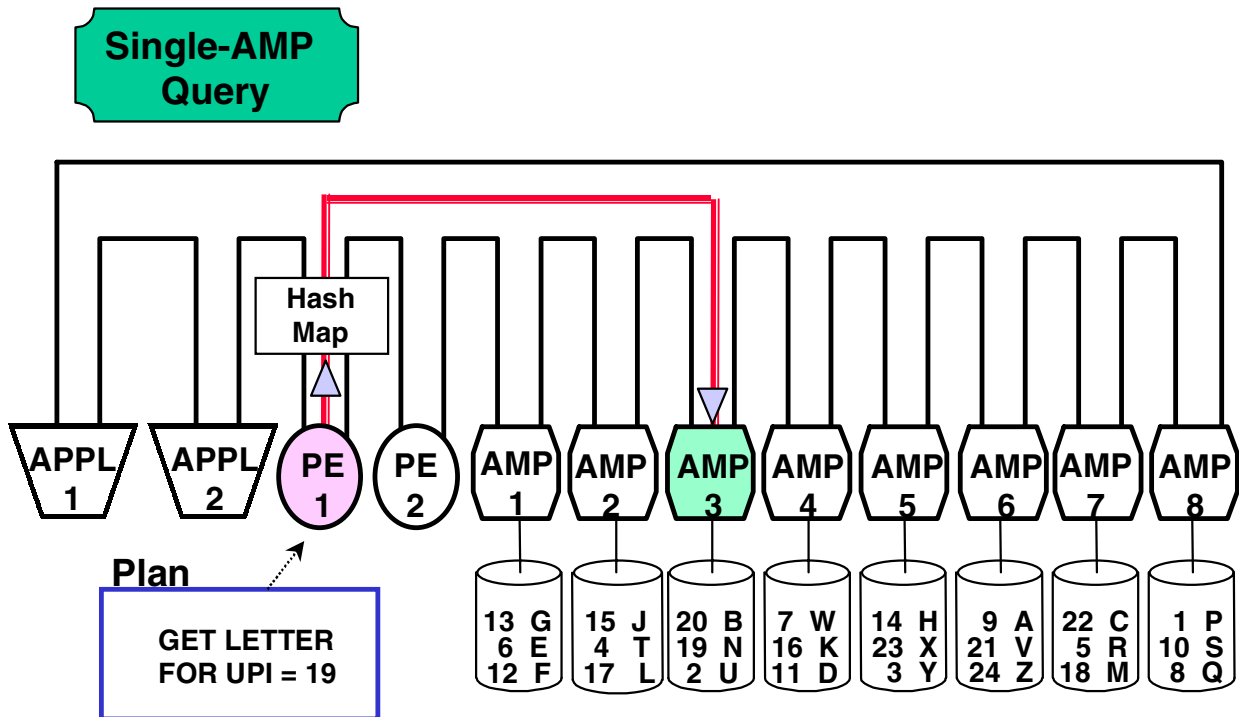
- APPL 1 establishes a user session on PE 1.
- APPL 1 sends the SQL request to the PE on the forward channel.
- PE 1 acknowledges the message on the back channel.
- PE 1 parses and optimizes the request.



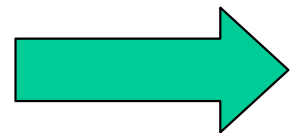
## **PE to AMP**

The PE gives the step message to the BYNET, which uses the primary hash map to determine the destination AMP. The BYNET then sends the message to the selected AMP across the forward channel. The AMP acknowledges the message across the back channel.

## PE to AMP



- PE 1 produces a one-step plan as a message to the BYNET.
- BYNET uses the hash map to determine the destination to AMP 3.
- BYNET sends the message to AMP 3 on the forward channel.
- AMP 3 acknowledges message across the back channel.

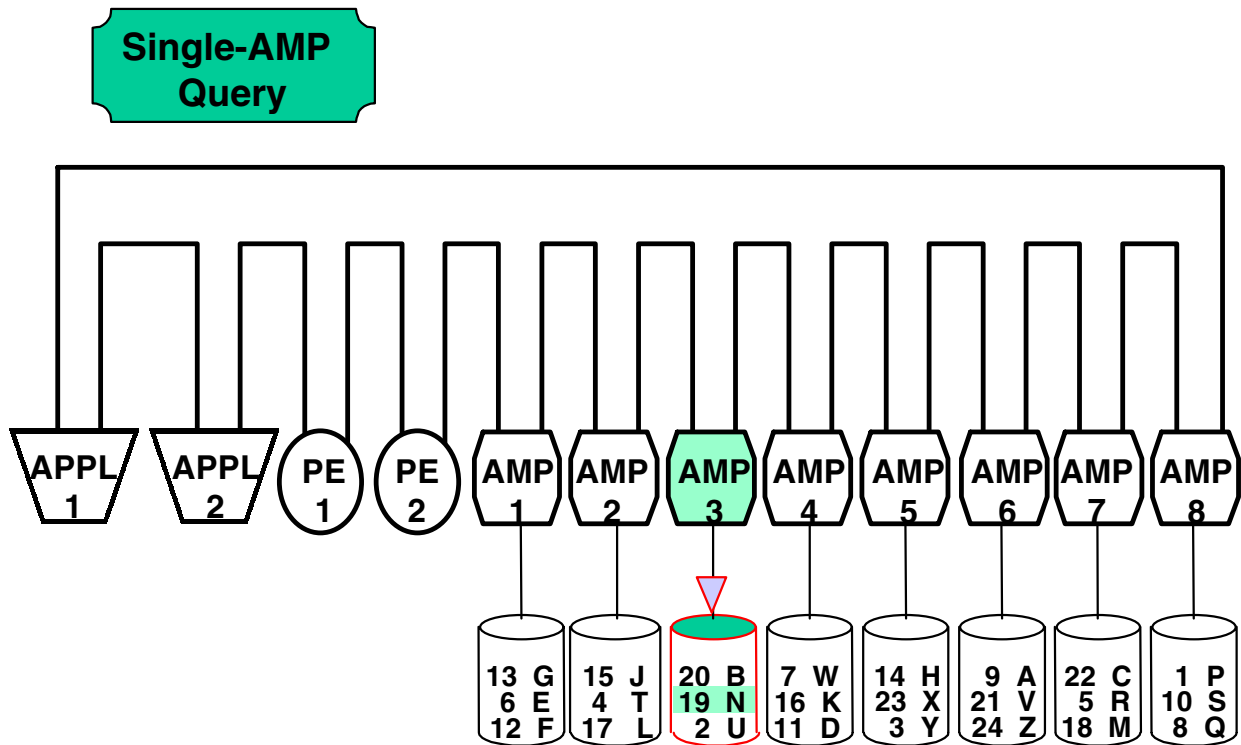


## **AMP to Vdisk**

The AMP locates the requested row on its vdisk by using the master and cylinder indexes. AMP 3 then creates a response message for placement on the BYNET.



## AMP to Vdisk



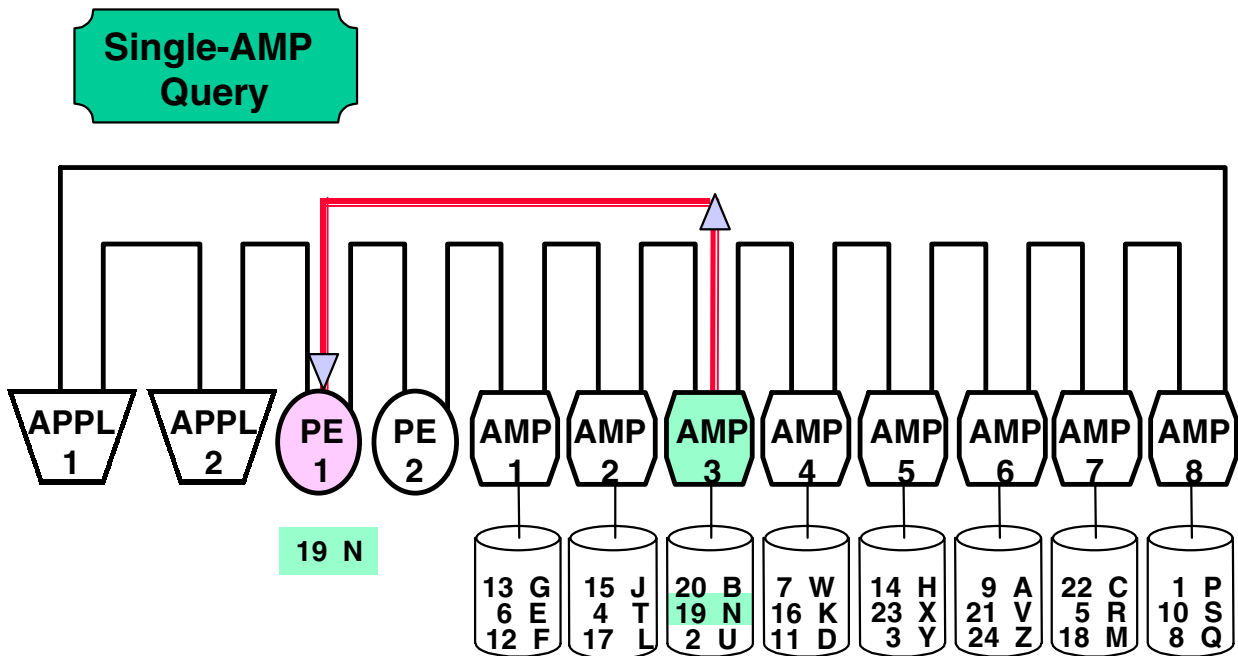
- AMP 3 uses master and cylinder indexes to locate requested row on its vdisk.
- AMP 3 creates a response set as a message for the BYNET.



## **AMP to PE**

The AMP sends the reply to the PE across the forward channel. The PE acknowledges the message across the back channel.

## AMP to PE



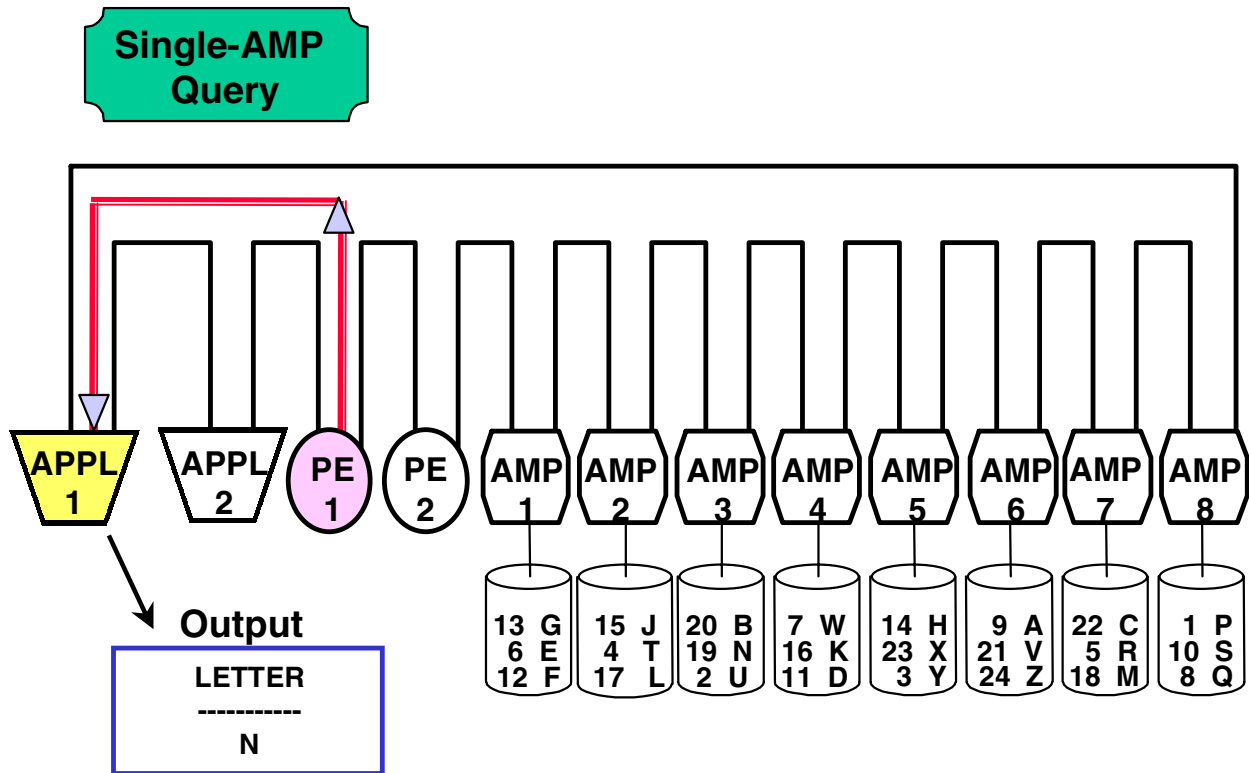
- AMP 3 sends answer set to PE 1 on forward channel.
- PE acknowledges receipt across back channel.



## **PE to Application**

The PE forwards the response message to the requesting application across the forward channel. The application acknowledges the message on the back channel. The application processes the response and returns it to the requesting user.

## PE to Application



- PE 1 forwards response parcels to APPL 1 on forward channel.
- APPL 1 acknowledges messages on back channel.
- APPL 1 processes response and generates output.

## All-AMP Query with a Sort

Consider the query on the facing page. It needs to find all rows whose number is greater than 9. To do so, it must query each AMP, performing an all-AMP operation. The request is for the data to be sorted by number. How can the AMP vprocs, discrete processing units, sort a final answer set? The BYNET plays an active role in sorting the result set.

# All-AMP Query with a Sort

SELECT NUMBER, LETTER FROM SAMPLE WHERE NUMBER > 9 ORDER BY LETTER ;	SAMPLE	
	NUMBER	LETTER
	UPI	
ANSWER: 20 B 22 C 11 D 12 F 13 G 14 H 15 J 16 K 17 L 18 M 19 N 10 S 21 V 23 X 24 Z	1	P
	2	U
	3	Y
	4	T
	5	R
	6	E
	7	W
	8	Q
	9	A
	10	S
	11	D
	12	F
	13	G
	14	H
	15	J
	16	K
	17	L
	18	M
	19	N
	20	B
	21	V
	22	C
	23	X
	24	Z

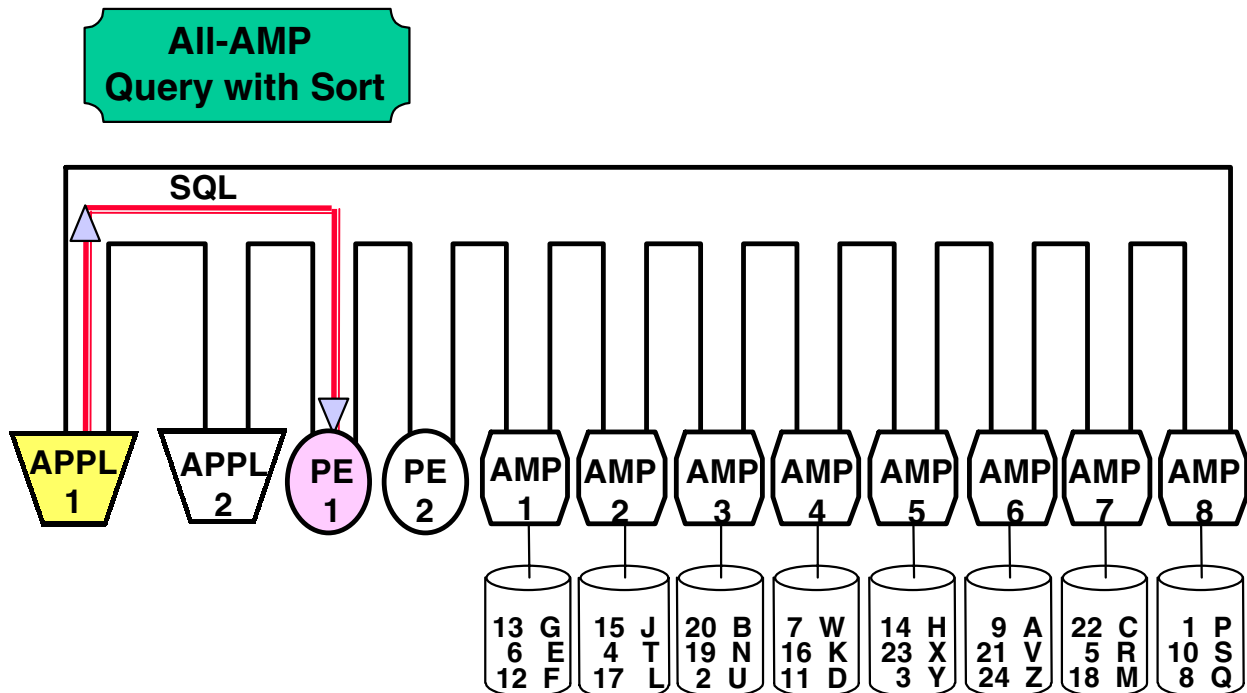


## **Application to PE**

The application first establishes a session with the PE. The application then sends the SQL request to the PE via the forward channel. The PE acknowledges the message on the back channel. The PE then parses and optimizes the request.



## Application to PE



### SQL Request

```
SELECT  NUMBER, LETTER
FROM    SAMPLE
WHERE   NUMBER > 9
ORDER BY LETTER ;
```

- APPL 1 establishes a user session on PE 1.
- APPL 1 sends the SQL request to the PE on the forward channel.
- PE 1 acknowledges the message on the back channel.
- PE 1 parses and optimizes the request.



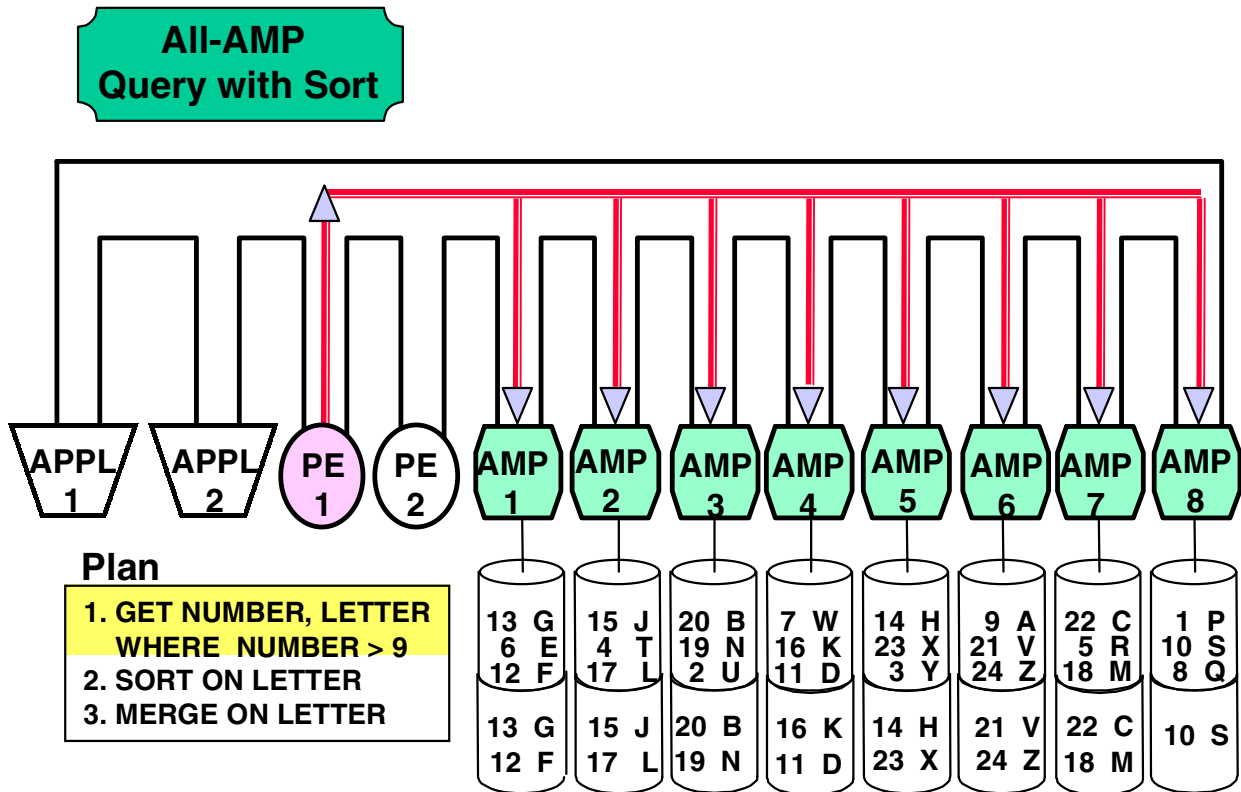
## **PE to AMPs**

The PE optimizer software creates a three-step plan to solve the query. The PE sends only one step of the plan to the AMPs at a time. It will not send a second step until the AMPs have completed the first step.

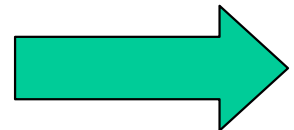
The PE sends the first step via the BYNET to all the AMPs.

The AMPs acknowledge receipt of the step via the back channel.

## PE to AMPs



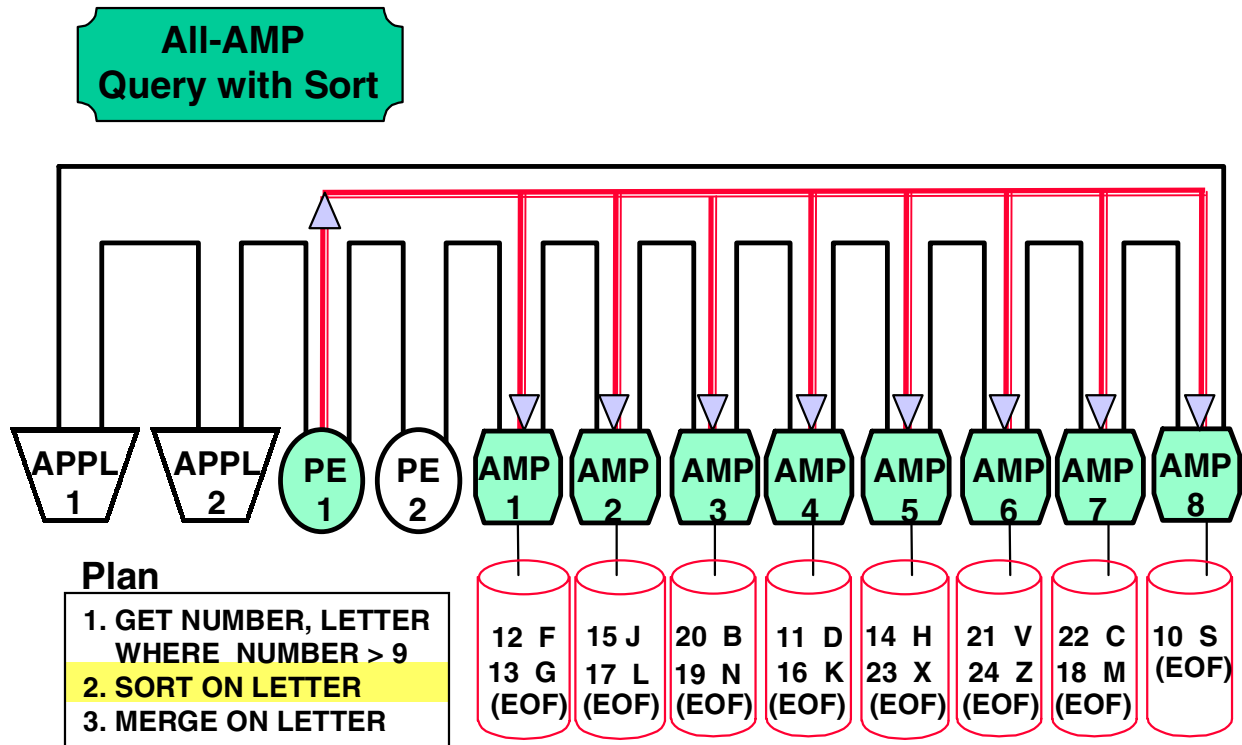
- PE1 produces a three-step plan.
- PE1 gives first step to BYNET to send to all AMPs.
- BYNET sends step over forward channel to all AMPs.
- All AMPs acknowledge receipt over back channel.



## **PE to AMPs**

After each AMP has been heard from, the PE then sends out step 2 across the BYNET to all AMPs.

## PE to AMPs



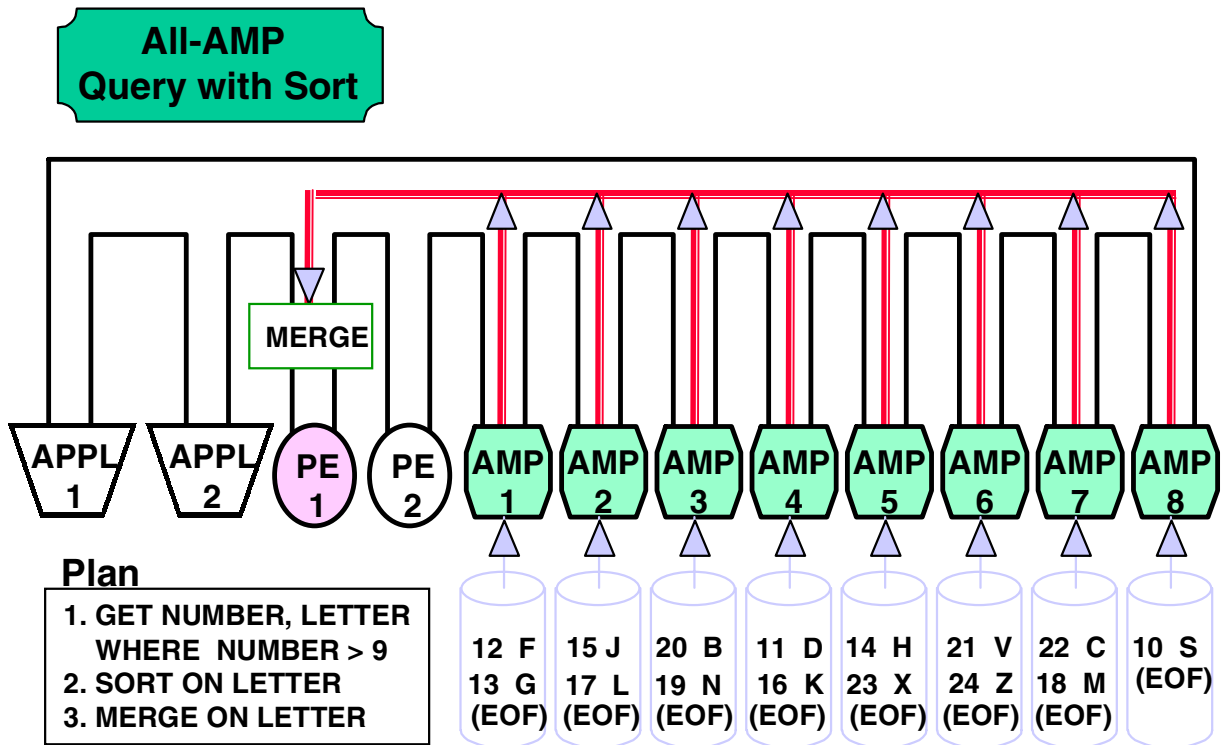
- PE1 sends out step 2 over the BYNET.
- BYNET sends step to all AMPs.



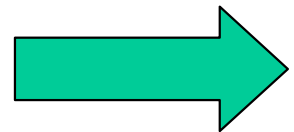
## **AMPs to Merge Process**

Each AMP will execute step 3. The diagram on the right shows each AMP sending its first sorted row to the BYNET merge process. This merge process resides in the node of the requesting PE. Depending on the size of the rows being merged, the process varies. For larger rows (over 2K), each row is sent individually. For smaller rows, sorted blocks of rows are sent to the merge process to reduce BYNET message traffic.

## AMPs to Merge Process



Each AMP sends its first block of sorted data to **BYNET merge process.**

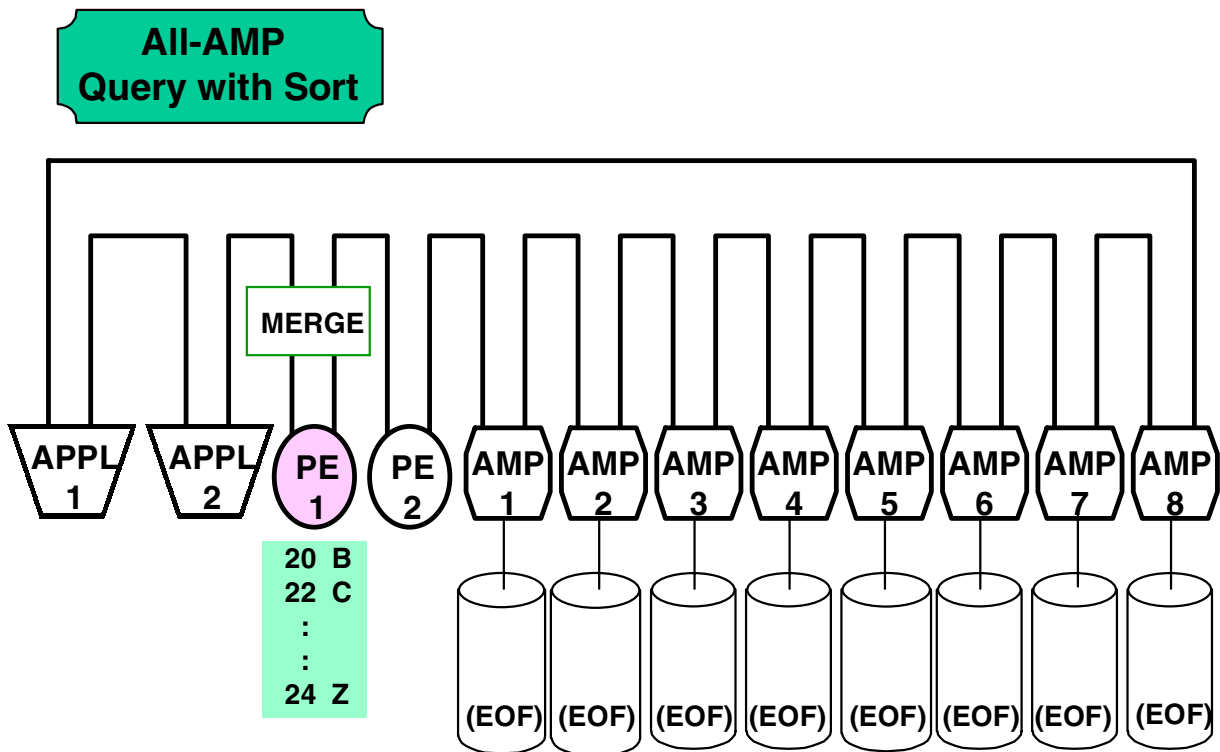


## **AMP to Merge**

At some point, all AMPs will have exhausted their supply of rows from spool. At this time, each AMP will have sent an end-of-file (EOF) indication to the BYNET. The BYNET merge process will realize it now has the entire answer set.



## AMP to Merge



### Plan

1. GET NUMBER, LETTER  
WHERE NUMBER > 9
2. SORT ON LETTER
3. MERGE ON LETTER

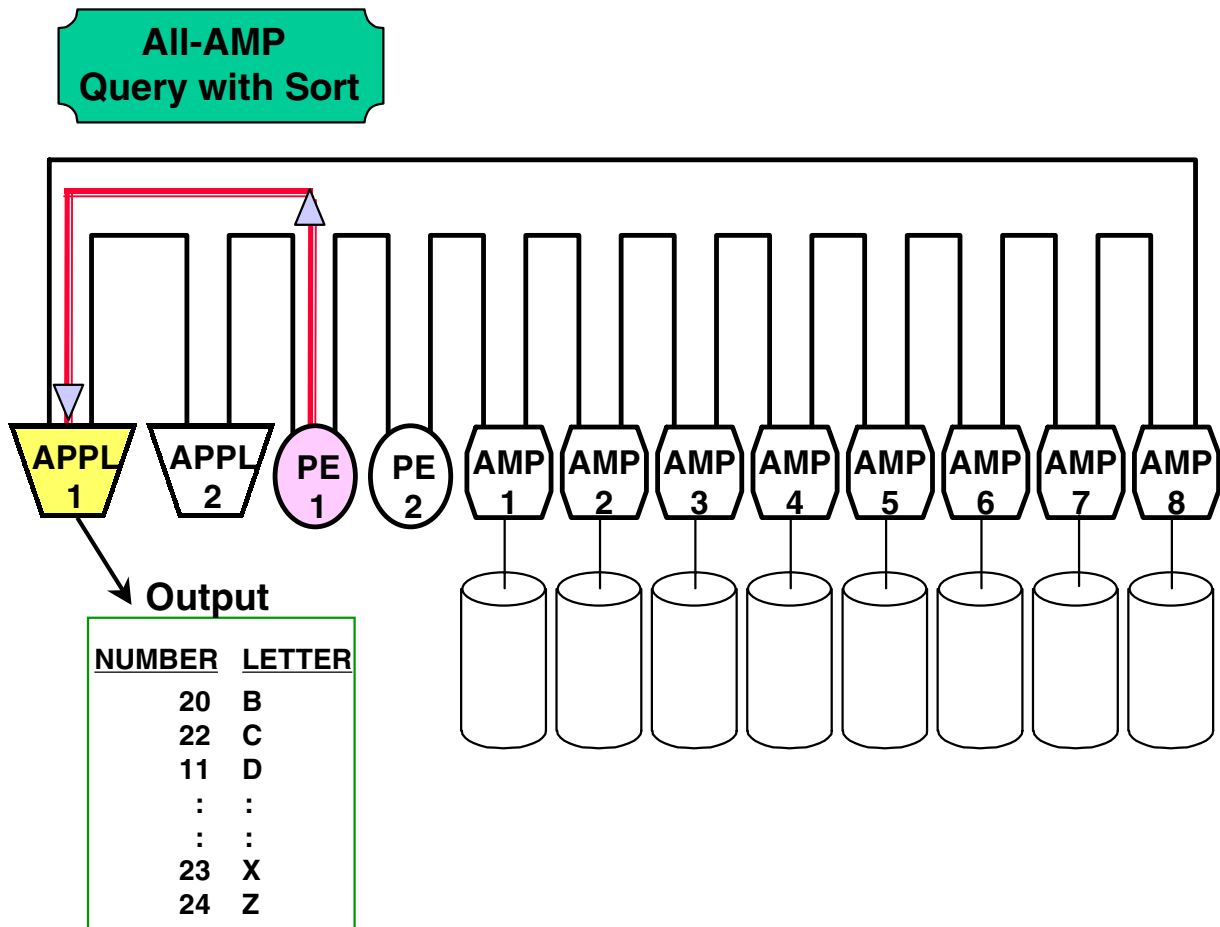
- The merge process continues to request sorted blocks from the AMPs until all AMPs have exhausted their spool supply.
- When the merge process has an EOF from each AMP, the answer set is complete.



## **PE to Application**

The PE sends the final answer set to the requesting application, APPL1. The PE will send either the complete answer set, or a buffer-full of rows, whichever occurs first.

## PE to Application



**PE1 then sends the answer set to the requesting application.**

## Review Questions

Check your understanding of the concepts discussed in this module by completing the review questions as directed by your instructor.

## **Review Questions**

**Indicate whether a statement is True or False.**

- 1. An example of a single-AMP request is a primary index retrieval.**
- 2. It is the role of the AMP to optimize the request.**
- 3. To locate rows on the vdisk, the AMP uses the hash map.**
- 4. AMPs do the sorting, the BYNET does the merging.**
- 5. A full-table scan requires the services of all AMP vprocs.**

## Notes

### Teradata Tools

After completing this module, you will be able to:

- **List the various utilities and support tools available for Teradata users.**
- List other vendor platforms which support the Teradata RDBMS.
- List the features of SQL under Teradata V2.

## Notes

*Attention Instructors! Topics in this module are reflected on the Certification Exam. Specifics will be noted throughout the module.*



## Table of Contents

Query Submitting Tools .....	4
Application Utilities .....	6
Administrative Tools .....	8
Programming Tools .....	10
Platform Support.....	12
Teradata SQL .....	14
Why Teradata? .....	16

# Query Submitting Tools

*Topics reflected on Certification Exam.*

Teradata provides at least two SQL query front-end products.

## BTEQ

BTEQ stands for Basic Teradata Query utility. It is an SQL front-end that runs on all client platforms. It supports both interactive ad hoc and scripted batch queries and provides standard report-writing functionality. It also provides a basic import/export capability. Other utilities are available for these functions, particularly when larger tables are involved. When a user logs on with BTEQ, they get one session as default, but they can set the session number they want before logging on.

## Teradata Query Manager

Teradata Query Manager (sometimes called Queryman) is an ODBC based front-end for Teradata SQL. It offers a full menu of services including query history, timings, status, row counts, random sampling, and limited answer sets. It provides an import/export feature between database and PC and also allows export to EXCEL and ACCESS.

### Customer quote:

“NCR’s Queryman product allows our development organization to use one product for SQL access to any of our different DBMS systems using each DBMSs ODBC connectivity. Because Queryman is installed on the PC, it is a very efficient way to provide access to our DBMS without establishing an ID on a host system.”

# Query Submitting Tools

## **BTEQ**

- **Basic Teradata Query utility**
- **SQL front-end**
- **Report writer features**
- **Interactive and batch queries**
- **Import/Export across all platforms**
- **The default number of sessions, upon login, is 1.**

## **Teradata Query Manager**

- **SQL front-end for ODBC databases**
- **Historical record of queries including:**
  - **Timings**
  - **Status**
  - **Row counts**
- **Random sampling feature**
- **Limit amount of data returned**
- **Import/Export between database and PC**
- **Export to EXCEL or ACCESS**

# Application Utilities

Teradata application utilities are used for table loading, maintaining, importing and exporting data. Each utility addresses specific needs though there are occasionally overlaps in functionality. Understanding the utilities helps you make the best choice for your specific requirements.

## FastLoad

FastLoad is a **data-loading utility**, specifically used for **empty tables**. It achieves high performance by fully exploiting the resources of the system and by using multi-session parallelism.

## MultiLoad

MultiLoad is a **table-maintenance utility**. It allows all basic **SQL/DML** functions. By using block transfers over multiple sessions, it can **work on up to five tables at a time** and can handle multiple input files concurrently. It has full restartability.

## FastExport

FastExport is a **data-extract utility**. It exports table data to files on the client system. It also operates with multi-session parallelism and block transfers to achieve high performance. It has full restartability.

## TPump

It uses standard **SQL/DML** to maintain data in tables and has a throttle that allows the user to specify what percentage of system resource is to be used for this operation. This allows background maintenance to take place any time of day. The throttle is set to best suit the processing resource at a specific time.

# **Application Utilities**

## **FastLoad**

- Data loading utility
- Loads empty tables
- Block transfers with multi-session parallelism

## **MultiLoad**

- Table loading/maintaining utility
- Runs against multiple tables
- Block transfers with multi-session parallelism
- Multiple input source files

## **FastExport**

- Data extraction utility
- Exports tables to client files
- Block transfers with multi-session parallelism

## **TPump**

- Teradata Parallel Data Pump
- Maintains up to 60 tables at a time
- Uses standard DML—not block transfers
- User may adjust throttle to change apply rate
- Same restartability, portability and scalability as MultiLoad

# Administrative Tools

## Teradata Manager

Teradata Manager is a graphical system management tool that permits a single view of the database system. Database Administrators and system managers use it to provide operational control of the Teradata environment. It is available with Windows 95 and Windows NT platforms for Version 2 systems. It provides performance monitoring and database utilization statistics that are collected, analyzed, and displayed. The product also includes WINDDI (Windows-based Data Definition Interface) that permits the coding of DDL statements by filling in forms. Teradata Manager can produce its reports either in standard report format or in graphical format.

## Database Query Manager (DBQM)

Database Query Manager (DBQM) is a query management utility which dynamically tunes the Teradata RDBMS. Knowledge worker queries are scheduled based on the complexity of the query. This ability is important because it accommodates mixed workloads (both sophisticated and simple queries) in real time. A Teradata knowledge worker has the freedom to explore any aspect of his/her data warehouse without experiencing unnecessary system delays.

# Administrative Tools

## Teradata Manager

- **Graphical system management tool**
- PC based (Windows NT, Windows 95)
- Collects, analyzes, and displays:
  - Performance information
  - Database utilization
- WinDDI for easy coding of DDL commands
- Graphical or report based
- Provides standard DBA functions

## Database Query Manager (DBQM)

- **Query workload management tool**
- Runs, suspends, schedules later or rejects query based on current workload
- Restricts queries based on analysis control:
  - Too long
  - Too many rows
- Restricts queries based on object control:
  - User ID
  - Table
  - Day/time
  - Group ID
- Logs workload performance for analysis
- Restricts based on environmental factors:
  - CPU
  - Disk utilization
  - Network activity
  - Number of users

# Programming Tools

NCR makes available several programming tools with the Teradata database.

## Preprocessor 2 (PP2)

PP2 is a product that permits the coding of embedded SQL into programs written in COBOL, PL1, or C. PP2 permits existing third generation language applications to access rows from Teradata and process them as records from a flat file. PP2 supports dynamic SQL that allows the program to dynamically construct SQL, which is created and prepared for execution at runtime. PP2 also supports multiple sessions and updateable cursors.

## Call Level Interface (CLI)

The CLI is the lowest level programmable interface to the Teradata RDBMS. It is a library of service routines, needed to access rows of data on the database. CLI can be used with most client languages for platforms supporting Teradata. CLI permits the threading of multiple requests against multiple sessions, providing a high level of performance. Parallel CLI permits the parallelization of the client-side application for total parallel performance on both the client and server side.

## Extended Call Level Interface (ECLI)

Extended Call Level Interface is a programming interface to the CLI that permits fewer and higher-level calls, reducing some of the complexity of coding at the CLI level. It is designed for large-volume, multisession applications.

## ODBC Interface

NCR provides the drivers and driver manager necessary to support ODBC applications using the Teradata RDBMS.

## Oracle Transparent Gateway

Allows transparent access to Teradata tables via Oracle front-end tools. Developed, provided, and supported by Oracle.

## WinCLI

A Call Level Interface for MS-DOS and Windows-based applications.

## TS/API

Transparency Series/Application Programming Interface permits ease of migration of IBM mainframe applications to the Teradata RDBMS.



# **Programming Tools**

## **Preprocessor 2 (PP2) support:**

- Use of embedded SQL in application
- COBOL, C, PL/1 languages
- Single or multi-sessions
- Dynamic SQL
- Updateable cursors

## **Call Level Interface (CLI)**

- A library of service routines to interface to Teradata RDBMS
- Requires low-level programming, but provides high performance
- Supports single or multi-sessions
- Supports all languages and platforms
- Parallel CLI allows parallelizing on client and service side

## **Extended Call Level Interface (ECLI)**

- Higher level CLI calls
- Easier to program than CLI
- Better workload balancing
- Designed for large data volume, multi-session applications

## **ODBC Interface**

## **Oracle Transparent Gateway**

## **WinCLI**

## **TS/API**

## **Platform Support**

Teradata supports all of the client platforms, database platforms, and channel-attached mainframe environments listed.

# Platform Support

## Client Platforms

<u>Microsoft</u>	
	MS-DOS Windows Windows NT
<u>IBM</u>	MVS VM OS2
<u>NCR</u>	UNIX SVR4 MP-RAS
<u>Other UNIX platform support</u>	
	HP-UP IBM AIX SUN Solaris DEC Alpha SGI/IRIX SUN OS Pyramid

## Channel-Attached Mainframes

IBM UNISYS Hitachi Amdahl Bull
--

## Database Server Platforms

NCR	UNIX SVR4 MP-RAS
Microsoft	Windows NT

# Teradata SQL

**Teradata SQL** added many new features and functions in Version 2.

## ANSI Compliance

Compliant with SQL92 at the Entry level and many Intermediate and Full features.

## Referential Integrity

Referential integrity may be elected at the table level to ensure the integrity of Primary Key and Foreign Key relationships.

## Updateable Cursors

Rather than having to re-read data, cursors declared in a preprocessor program may now do in-place updates.

## Statistical Functions

Traditional analytical processing functionality with support for moving averages, rankings, and forecasting among others.

## Data Sampling

Allows analysis to be performed on data samples, which can then be extrapolated to greater volumes.

## Global and Volatile Temporary Tables

Add to the derived table capability of Version 2. Support temporary tables that can be passed on to different steps of a job stream.

## Database Triggers

Cause an update to propagate downstream by either cascading the update or preventing it until further changes have occurred.

## Join Index

Allows a permanent index to be created that serves as a pre-joined table and is available to the Optimizer as a tool for resolving the query.

## Correlated Subqueries

Permit subqueries, which are correlated to information in the outer query.

# Teradata SQL

**Under Version 2 (V2), Teradata SQL has added:**

- **ANSI compliance**
  - Entry level SQL92
- **Referential integrity**
  - PK, FK consistency
- **Updateable Cursors**
  - Update in place without index
- **Statistical Functions**
  - Moving averages, rankings, forecasting
- **Data Sampling**
  - Data mining support
- **Global and Volatile Temporary Tables**
  - Automatically created and deleted
- **Database Triggers**
  - Parallel and optimized
  - Cascade or prevent
- **Join Index**
  - For prejoining of tables
- **Correlated subqueries**

# Why Teradata?

## Customer comments:

“We selected Teradata because it was the only solution that worked in the 1+ terabyte space. When we compared it with competing products, there was nobody else with the utilities and durability in the installed base to support that type of installation.”

“In Teradata, we’re talking about one DBA maybe 75% of the time who is actually managing the environment. There is no notion of defrag or reorg or things like that actually taking place in the Teradata environment. So that is a big plus on our part—it saves a whole lot of weekends.”

“The secret to data warehousing is not just the ability to lay down a terabyte of data onto disk. Lots of vendors can do that. **The trick is to be able to access it and use it in a meaningful way.** That means lots of users doing lots of queries, some ad hoc, some planned, all getting what they need in **a timely fashion.**”

If you can’t do that, your data warehouse might as well have a big lock on the door.”

## Why Teradata?

NCR/Teradata is the world leader in large scalable data warehousing.

NCR/Teradata:

- **Supports more warehouse data** than all competitors combined.
- Supports easy scalability from a small (10 GB) to a massive (100+TB) database.
- Provides a query optimizer with approximately 20 years of experience in large-table query planning.
- Does not require complex indexing schemes, complex data partitioning or time-consuming reorganizations.
- Supports ad hoc querying against the detail data in the warehouse, not just summary data in the data mart.
- Designed and built with parallelism from day one (not a parallel retrofit).

# Notes



## Review Questions/Solutions

# **Module 1 Review Questions**

## **Solutions**

1. When was the first Teradata product sold?  
**1984**
2. One trillion is written as a 1 (one) followed by how many zeroes?  
**12**
3. Which feature of Teradata permits high performance even against enormous databases?  
**Parallelism**
4. Which two operating systems can be the platform for Teradata?  
**UNIX and NT**
5. Is Teradata a client or a server?  
**Database server**
6. What operating systems can be used with Teradata?  
**UNIX, Windows NT**
7. Name at least 3 features of Teradata.

**Relational, Large capacity database machine, Performance, Single data store for multiple client hosts, Network connectivity, Standard access language (SQL), Manageable growth, Fault tolerance, Data integrity**

# **Module 2 Review Questions**

## **Solutions**

**Match each term with its definition:**

**f** 1. Database

**e** 2. Table

**b** 3. Relational database

**a** 4. Primary Key

**d** 5. Null

**c** 6. Foreign Key

- a. A set of columns that uniquely identify a row
- b. A set of logically related tables
- c. One or more columns that are a PK somewhere in the database
- d. The absence of a value
- e. A two-dimensional array of rows and columns
- f. A collection of permanently stored data

# Module 3 Review Questions

## Solutions

1. Name the three major Teradata components and their purposes?

**PEs—Parse, Optimize and Dispatch queries**

**AMPs—Data storage and retrieval**

**BYNET—Communication between AMPs and PEs**

2. What are three Teradata database objects?

**Tables, views, and macros**

3. What are views?

- **Subset of rows and columns of one or more tables**
- **Virtual tables**
- **Window into one or more tables**

4. What are macros? What privileges do you need to work with macros?

- **Predefined, stored set of SQL statements**
- **CREATE, DROP, and EXECUTE privileges**

5. How many sessions can a PE support?

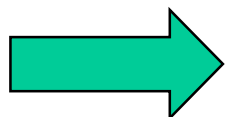
**120**

6. What happens to performance if you double the number of AMPs?

**It doubles**

7. Name the 2 types of virtual processors.

**AMP and PE**



# Module 3 Review Questions—Continued

## Solutions

### MATCH QUIZ 1

- |                 |          |   |
|-----------------|----------|---|
| 1. CLI          | <u>i</u> | a. Does locking, sorting, inserting, etc.                             |
| 2. MTDP         | <u>f</u> | b. Handles up to 120 sessions   |
| 3. MOSI         | <u>e</u> | c. Handles communication between AMPs and PEs                         |
| 4. PE           | <u>b</u> | d. Balances sessions across PEs for channel-attached systems          |
| 5. SQL          | <u>k</u> | e. Provides client-side OS independence                               |
| 6. AMP          | <u>a</u> | f. Manages data communication for network clients                     |
| 7. BYNET        | <u>c</u> | g. Plans most efficient steps to return response                      |
| 8. TDP          | <u>d</u> | h. At the client, packages queries into blocks and unpackages results |
| 9. Optimizer    | <u>g</u> | i. Library of Teradata Service Routing                                |
| 10. Dispatcher  | <u>h</u> | j. Foundation of Teradata architecture                                |
| 11. Parallelism | <u>j</u> | k. Language used to access data on tables                             |

### MATCH QUIZ 2

- |            |          |  |
|------------|----------|--|
| 1. HELP    | <u>b</u> | a. Displays the DDL for an object                  |
| 2. EXPLAIN | <u>d</u> | b. Displays information about objects              |
| 3. SHOW    | <u>a</u> | c. Displays path PE optimizer will use for a query |
| 4. DML     | <u>e</u> | d. Allows you to create an object                  |
| 5. DDL     | <u>c</u> | e. Allows you to retrieve a row (e.g. SELECT)      |

# Module 4 Review Questions

## Solutions

1. Name the four types of enterprise data processing and give examples of each.

**OLTP**      Withdraw cash from ATM

**DSS**      How many blue jeans were sold across all of our Eastern stores in the month of March in child sizes?

**OLCP**      Instant credit - How much of a credit line can be extended to this person?

**OLAP**      Show the top ten selling items for 1997 across all stores.

2. What is the difference between a data warehouse and a data mart?

**Data Warehouse**      Central enterprise-wide, detail data, historically unlimited

**Data Mart**      A special purpose subset of enterprise data for a particular function or application. It may contain detail or summary data or both. Limited history.

3. Which type of data mart gets its data directly from the data warehouse?      **Logical**

4. Which type of data mart should be avoided if you want to have only one version of the truth?  
**Independent**

5. What distinguishes the data found in the warehouse from the data found in a mart?

**Summary vs. Detail data**

# **Module 5 Review Questions**

## **Solutions**

- 1. What is the communication layer in a single node Teradata system?**

**Boardless BYNET**

- 2. What are cliques?**

**A group of nodes that share a disk array.**

- 3. What type of failure does a clique protect the system from?**

**The loss of a node**

- 4. What is the maximum number of vprocs in a clique?**

**128**

# **Module 6 Review Questions**

## **Solutions**

**Indicate whether a statement is True or False.**

1. A database will always have tables. **False**
2. A user will always have a password. **True**
3. The super-user at the top of the hierarchy of databases is always "SYSDBA." **False**
4. A user creating a subordinate user must give up some of their Perm Space. **False**
5. Tables must always be assigned a Primary Index at creation. **True**
6. Perm Space is not pre-allocated. **False**
7. The sum of all user and database Perm Space will equal the total available space on the system. **True**
8. The sum of all user and database Spool Space will equal the total available space on the system. **False**
9. Before a user can read a table, a table SELECT privilege must exist in the DD for that user. **True**
10. Deleting a macro from a database reclaims Perm Space for the database. **False**
11. Secondary indexes may be created or dropped at any time during the life of the table. **True**



# **Module 7 Review Questions**

## **Solutions**

1. For each statement, indicate whether it applies to: UPIs, NUPIs, Either, or Neither
  - a. Specified in CREATE TABLE statement. **Either**
  - b. Provides uniform distribution via the hashing algorithm. **UPI**
  - c. May be up to 16 columns. **Either**
  - d. Always a one-AMP operation. **Either**
  - e. Access will return a single row. **UPI**
  - f. Used to assign a row to a specific AMP. **Either**
  - g. Allows nulls. **Either**
  - h. Value cannot be changed. **Neither**
  - i. Required on every table. **Either**
  - j. Permits duplicate rows. **NUPI**
  - k. Can never be the PK. **NUPI**
2. Why is the choice of a Primary Index important?  
**It affects the distribution of rows, and thus performance.**

# Module 8 Review Questions

## Solutions

Fill in the blanks.

1. To identify the location of a row on an AMP, the system uses the primary index value and row hash.
2. The output of the hashing algorithm is called the row hash.
3. To determine the target AMP, the BYNET must look up an entry in the Hash Map based on the bucket number.
4. Two different PI values which hash to the same value are called hash synonyms.
5. A Row-ID consists of a row hash plus a uniqueness value.
6. A uniqueness value is required to produce a unique Row-ID because of hash synonyms and NUPI duplicates.
7. Rows in a data block are maintained in Row-ID sequence by the pointer array.
8. Because a new block is always allocated for any insert, update, or delete operation, reorganization is never needed.
9. Rows are, by definition, of varying length in a block.
10. Hash maps are configured base on the size of a system.

## Module 9 Review Questions

### Solutions

	USI Access	NUSI Access	FTS
# AMPs	2	All	All
# rows	0 - 1	0 - n	0 - n
Parallel Operation	N	Y	Y
Uses Separate Sub-table	Y	Y	N
Reads all data blocks of table	N	N	Y

Indicate whether a statement is True or False.

1. A query using a secondary index requires at least 2 AMPs. **True**
2. A NUSI avoids a full-table scan by limiting the number of AMPs accessed. **False**
3. A USI can be used to enforce uniqueness on a Primary Key column. **True**
4. You can create or drop USIs and NUSIs at any time. **True**
5. An example of the syntax to create a NUSI is:  
CREATE NUSI (last\_name) on CUSTOMER ; **False**
6. A full-table scan is never efficient because it accesses rows multiple times. **False**
7. A full-table scan can occur when there is a range of values specified for columns in a primary index. **True**

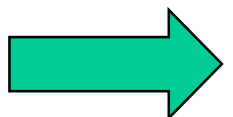
# Module 10 Review Questions

## Solutions

Match each item with its definition:

- f   1. Database lock
- c   2. Table lock
- h   3. Row Hash lock
- i   4. Fallback
- j   5. Cluster
- d   6. Recovery journal
- a   7. Transient journal
- g   8. ARC
- b   9. ASF2
- e   10. Permanent journal
- k   11. Clique

- a. Provides for TXN rollback in case of failure
- b. X-windows-based archive utility
- c. Protects all rows within
- d. Logs changed rows for down AMP
- e. Provides for recovery to a point in time
- f. Applies to all tables and views within
- g. Multi-platform archive utility
- h. Lowest level of protection granularity
- i. Protects tables from AMP failure by using alternative tables
- j. Fault-tolerant set of AMPs
- k. Nodes sharing disk array



# **Module 10 Review Questions—Continued**

## **Solutions**

- 1. FALLBACK is recommended if the application requires constant availability.**

**True**

- 2. FALLBACK is available only when you use CREATE TABLE.**

**False**

- 3. An Exclusive lock applies to rows.**

**False**

- 4. With the Transient Journal, before images are discarded only after the transaction is committed, which occurs only after an END transaction request.**

**True**

- 5. The Permanent Journal does not require user intervention, as it is automatically purged after a set period.**

**False**

# **Module 11 Review Questions Solutions**

**Indicate whether a statement is True or False.**

- 1. An example of a single-AMP request is a primary index retrieval.**

**True**

- 2. It is the role of the AMP to optimize the request.**

**False**

- 3. To locate rows on the vdisk, the AMP uses the hash map.**

**False**

- 4. AMPs do the sorting, the BYNET does the merging.**

**True**

- 5. A full-table scan requires the services of all AMP vprocs.**

**True**

### Module 2 Exercise Solutions

# Exercise—Answering Questions with a Relational Database

EMPLOYEE (partial listing)

EMPLOYEE NUMBER	MANAGER EMPLOYEE NUMBER	DEPARTMENT 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	Kanieski	Carol	870201	680517	3925000
1005	0801	403	431100	Ryan	Loretta	861015	650910	4120000
1004	1003	401	412101	Johnson	Darlene	861015	560423	4630000
1007			432101	Villegas	Arnando	870102	470131	5970000
1003	0801	401	411100	Trader	James	860731	570619	4785000

DEPARTMENT

DEPARTMENT NUMBER	DEPARTMENT NAME	BUDGET AMOUNT	MANAGER EMPLOYEE NUMBER
PK			FK
501	marketing sales	80050000	1017
301	research and development	46560000	1019
302	product planning	22600000	1016
403	education	93200000	1005
402	software support	30800000	1011
401	customer support	98230000	1003
201	technical operations	29380000	1025

1. Name the department in which James Trader works.

**Customer Support**

2. Who manages the Education Department?

**Loretta Ryan**

3. Identify by name an employee who works for James Trader.

**Darlene Johnson**

4. James Trader manages which department?

**Customer Support**