teradata.

Teradata Database

Vantage

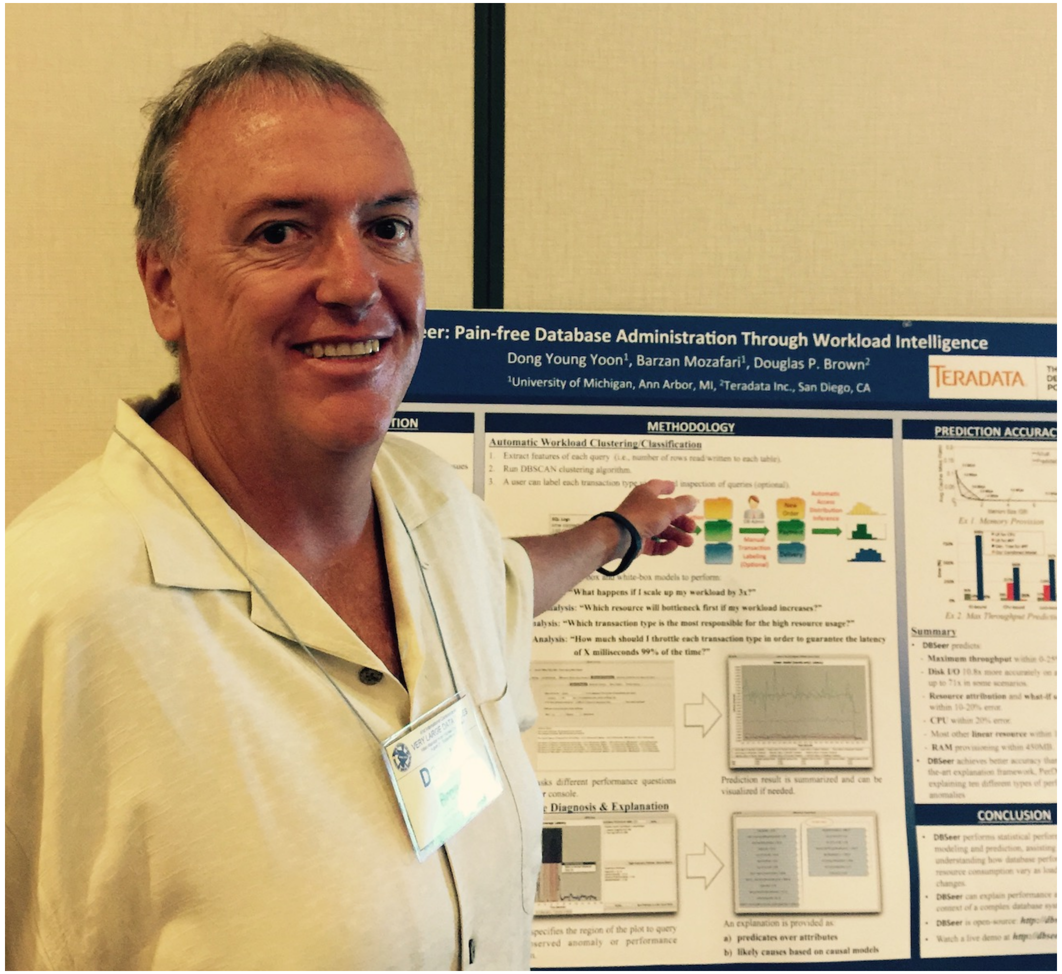
Douglas Brown

May 12, 2022

Agenda

- Who Am I - Background
- Teradata Database High Level Overview
  - Key Components in the Teradata Architecture
- Teradata MPP architecture
- Data Management
- Query Execution
- Optimizer
- Workload Management
- Technology Research projects

Doug Brown
Engineering Fellow - Technology Innovation Office (TIO)



Doug is currently a member of the Technology Innovation Office (TIO) in San Diego. He is presently the lead researcher and Product Owner (PO) for the workload management components of the Teradata database aka Teradata Active Systems Management (TASM).

Doug has been working in the Teradata database engineering team since 1989 and has contributed over 75 US Patents.

teradata.

1979
Birth of Teradata

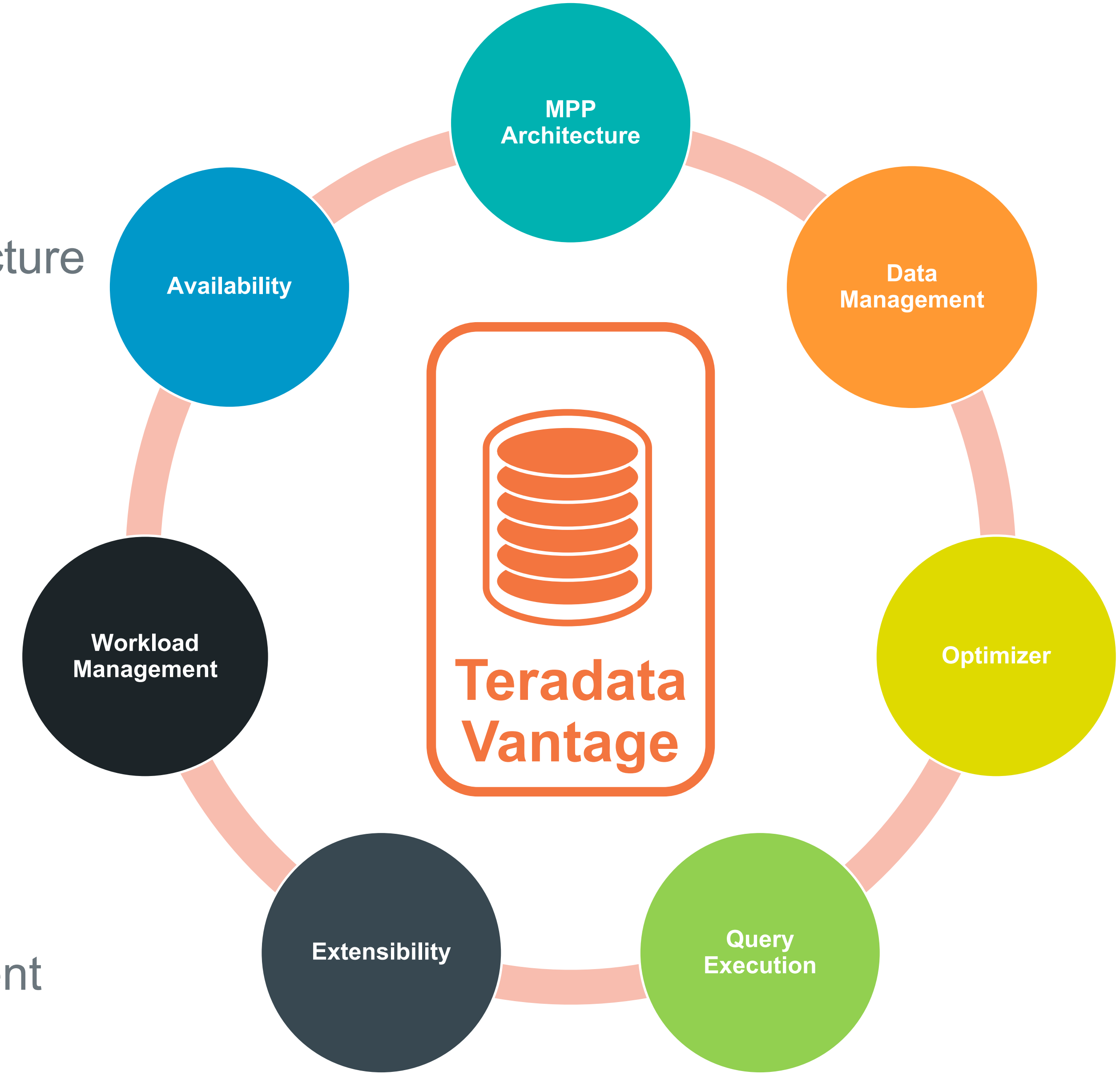
- Microprocessor - 8086
- Personal computer - Commodore 64, Apple I, TRS80
- Disk drive - 200MB weighs 30lbs and requires a washing machine sized cabinet
- Enterprise computing - Blue or Bunch mainframe
- Client/Server computing - didn’t exist
- Internet and e-mail - Only if you are a university researcher or DoD


Teradata DBC/1012

Teradata was founded to solve enterprise Terabyte problems utilizing microprocessors and commodity disks.

Key Components/Concepts

- MPP Shared-nothing Software-based Architecture
  - Hash-based with partitioning (row and column)
- Data Management
  - Relational, semi-structured, unstructured
- Cost-based Optimizer
- Query Execution
  - SQL Engine, Machine Learning Engines
- Extensibility
  - UDTs, UDFs, procedures, foreign servers
- High-concurrency mixed workload management
- Reliability and availability



Teradata Data Warehouse Technology Is Unique

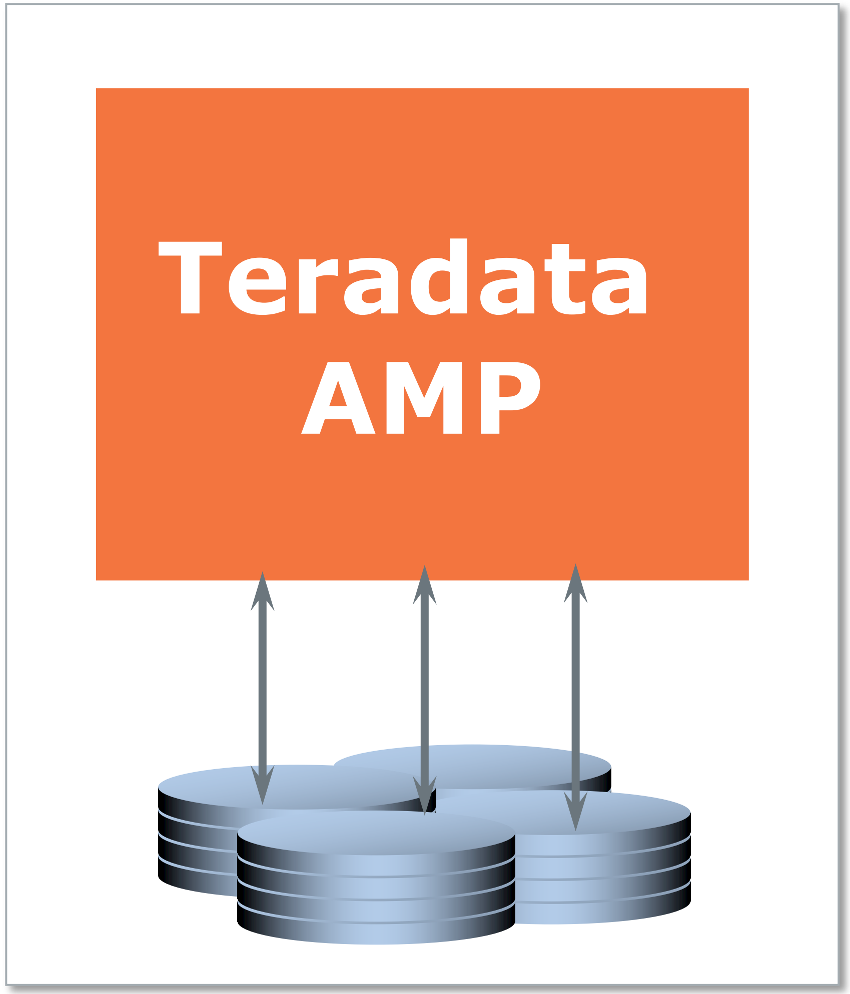
- Parallelism built-in from the ground up
- Dedicated to automatic management and operation
- Easy and fast data movement in and out
- Committed to the highest levels of reliability and availability
- High concurrency mixed workload management
- Parallel extensibility for high performance dynamic functionality
- ***Unequaled scalability in every dimension***

Architected to deliver business value to users

teradata.

Teradata Access Module Processor (AMP)

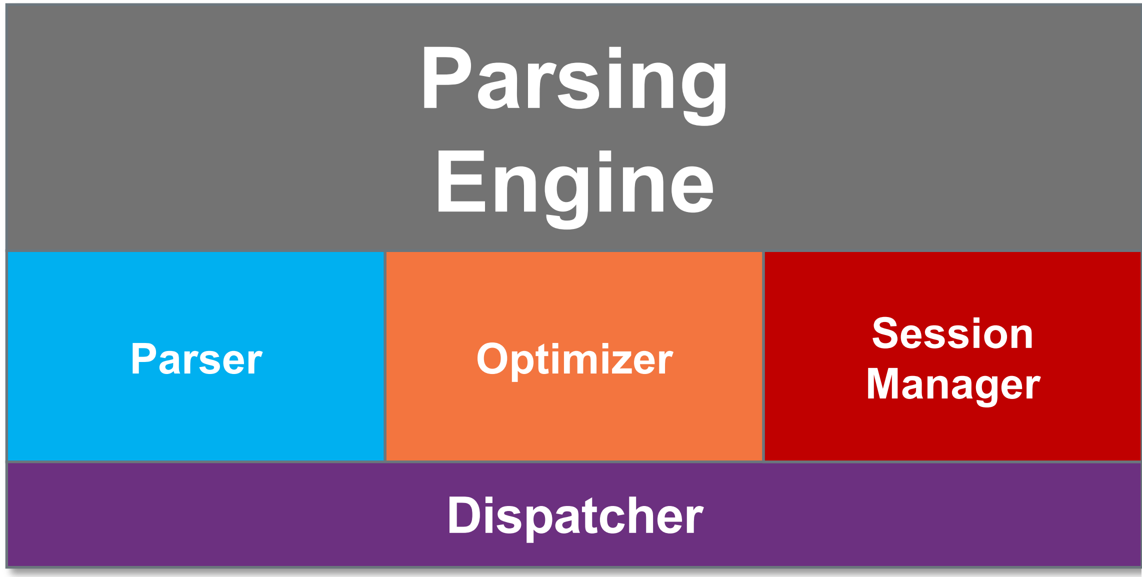
- A single instance of Teradata has many database execution engine virtual processors (vprocs) called Access Module Processors (AMP)
- Each owns a portion of the data (1/# of AMPs)
- Performs all operations on data (select, insert, update, delete, index, join, aggregate,…)
- Each AMP operates independently on steps sent from Dispatcher



teradata.

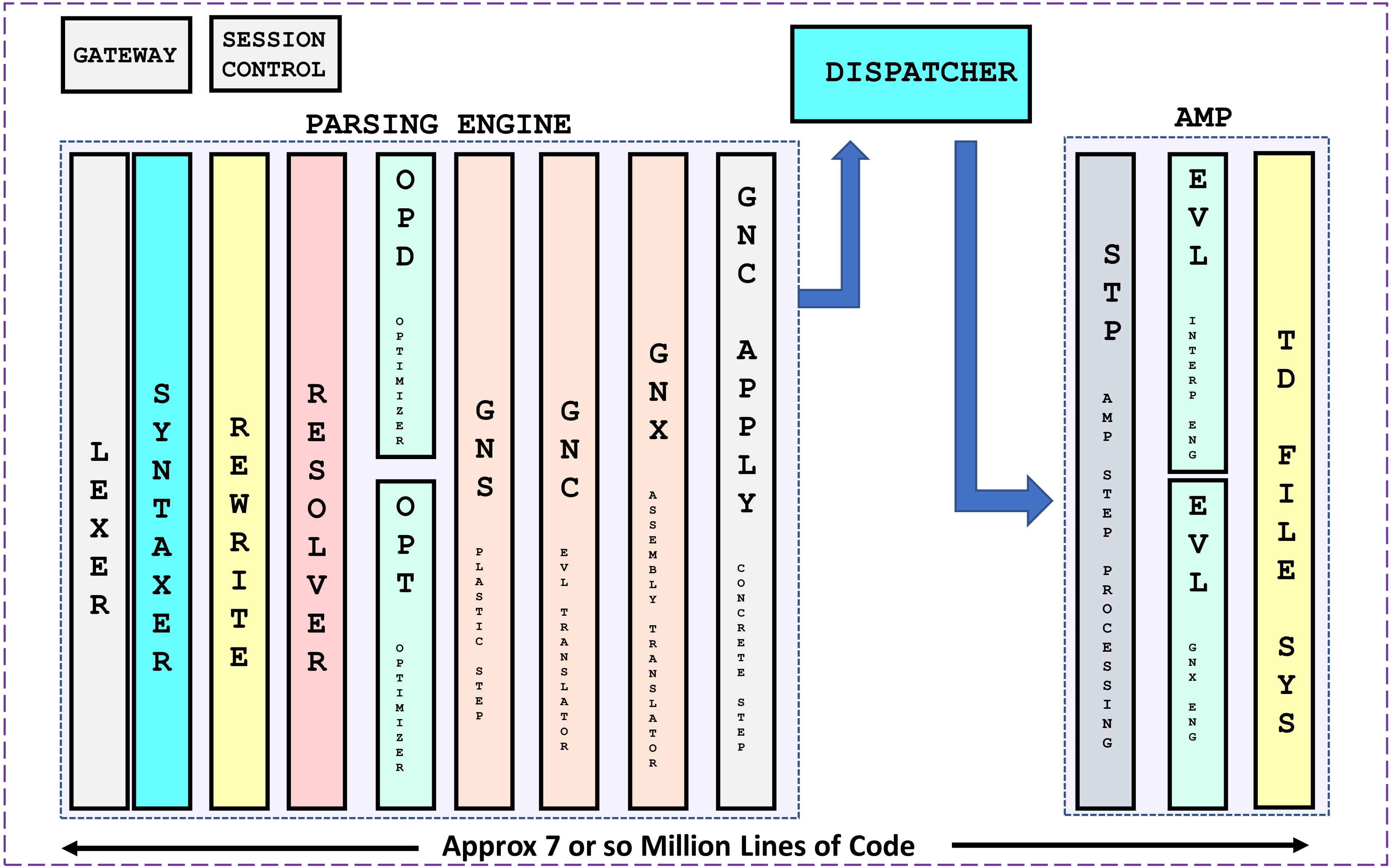
Teradata Parsing Engine (PE)

- **SQL Parser**
  - Analyzes SQL statement for proper syntax
  - Resolves object names
  - Validates that the user has the appropriate access rights to the objects
- **Optimizer**
  - Uses hardware configuration and statistics to generate a step by step execution plan
  - Calculates cost for all reasonable choices
  - Automatically rewrites the query if the cost can be improved



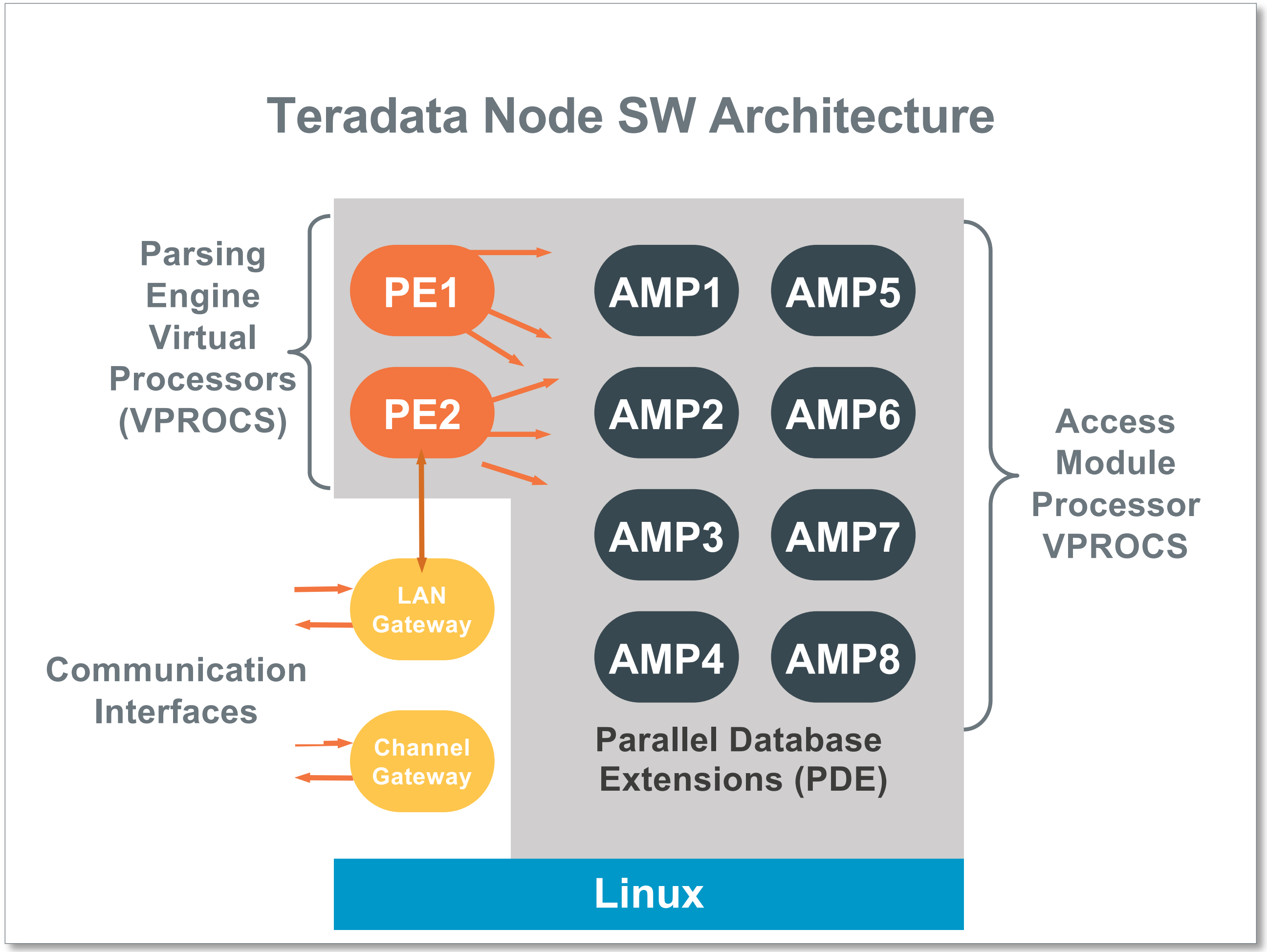
- **Query Step Dispatcher**
  – Sends the steps in the execution plan to a single AMP, a group of AMPs, or all AMPs
- **Session Manager**
  – Manages each session and communication to the client
- Input Data Conversion
  – EBCDIC to ASCII conversion

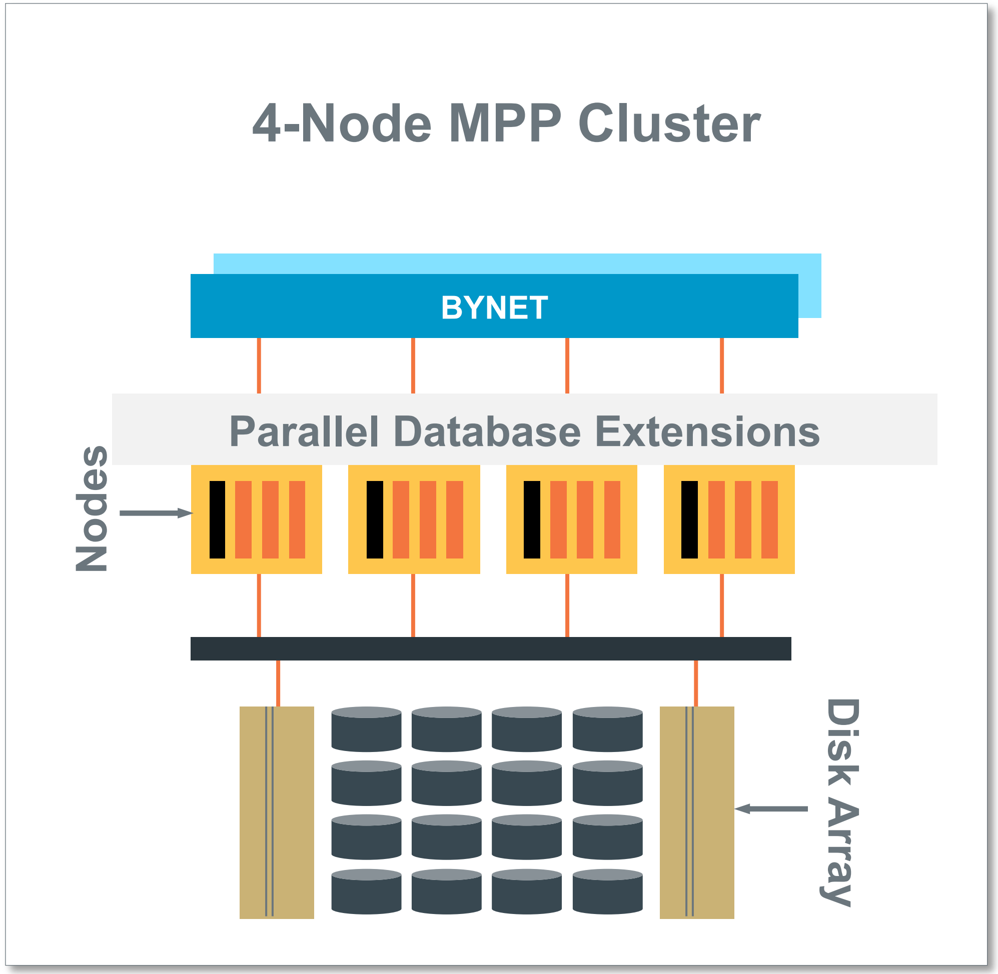
TERADATA "STACK"



teradata.

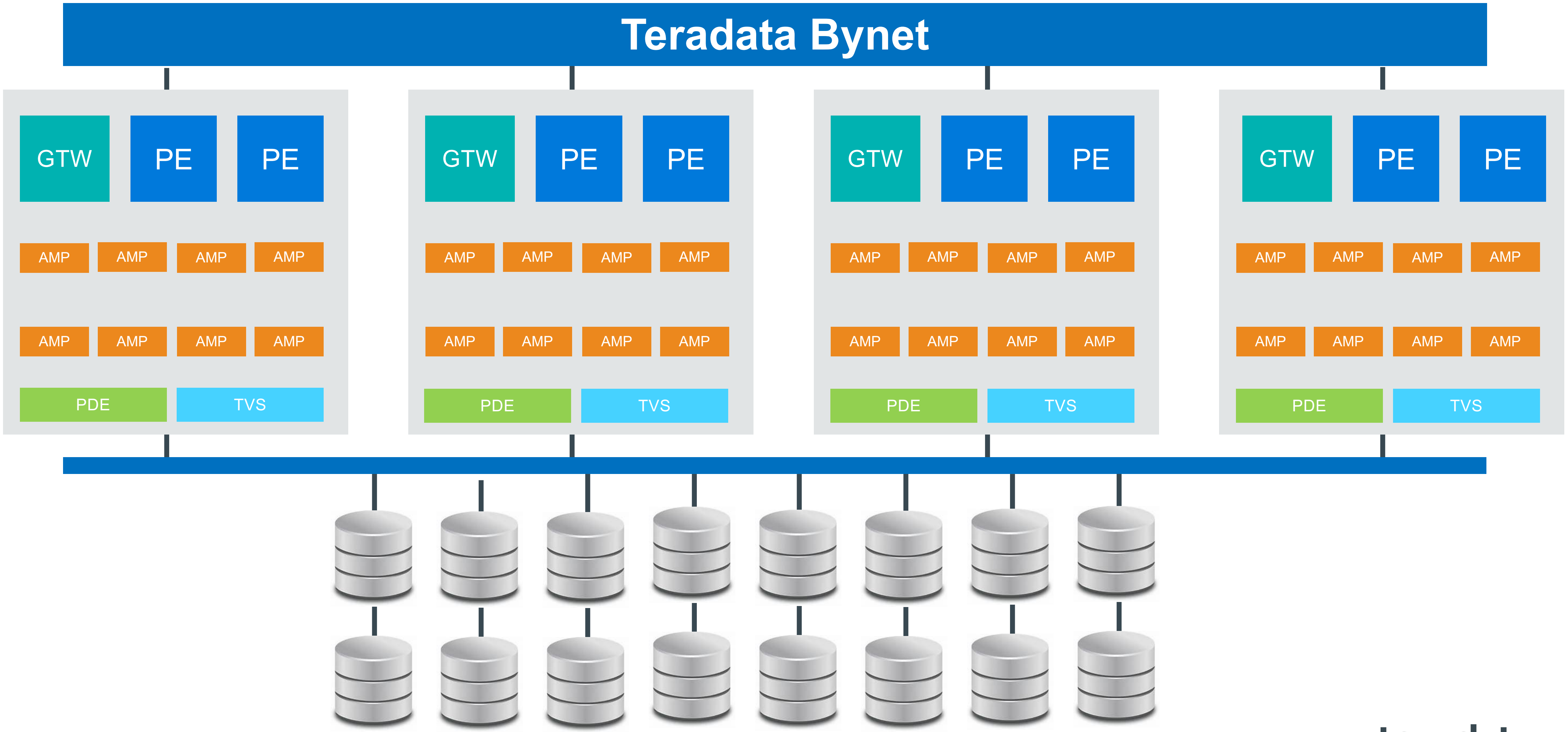
Parallel Database Extensions (PDE) Ties It All Together





teradata.

Unlimited Parallelism

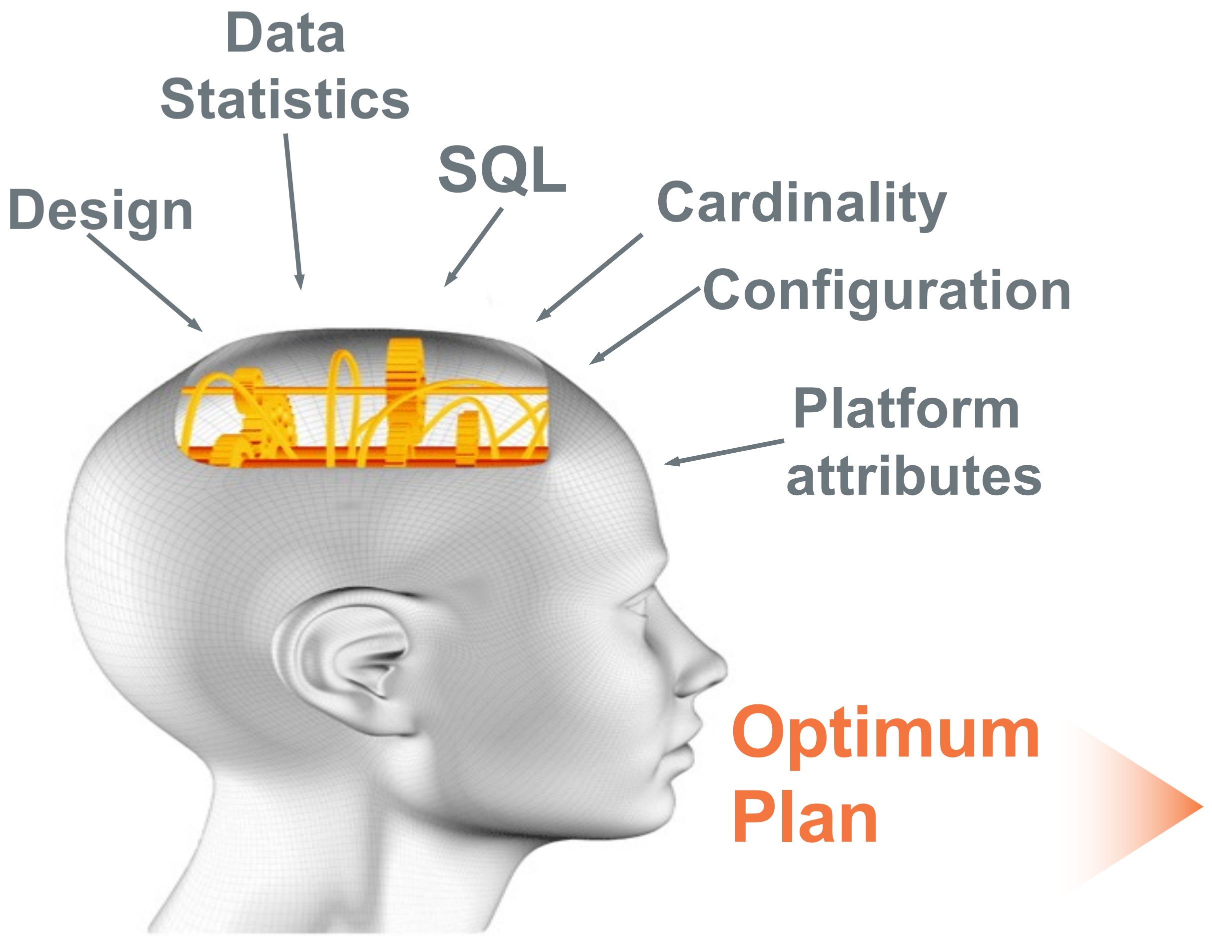


teradata.

Teradata Optimizer

Optimizer

- Cost based query planning
  - Balance costs of
    - Cardinalities
    - Sorts and joins
    - Redistribution
    - Disk I/Os
    - Networks and nodes
    - Compression
  - Lowest resource costs
  - Fastest elapsed time
- Automatic rewrites
- Automatic optimization
  - No hints
  - No degrees of parallelism

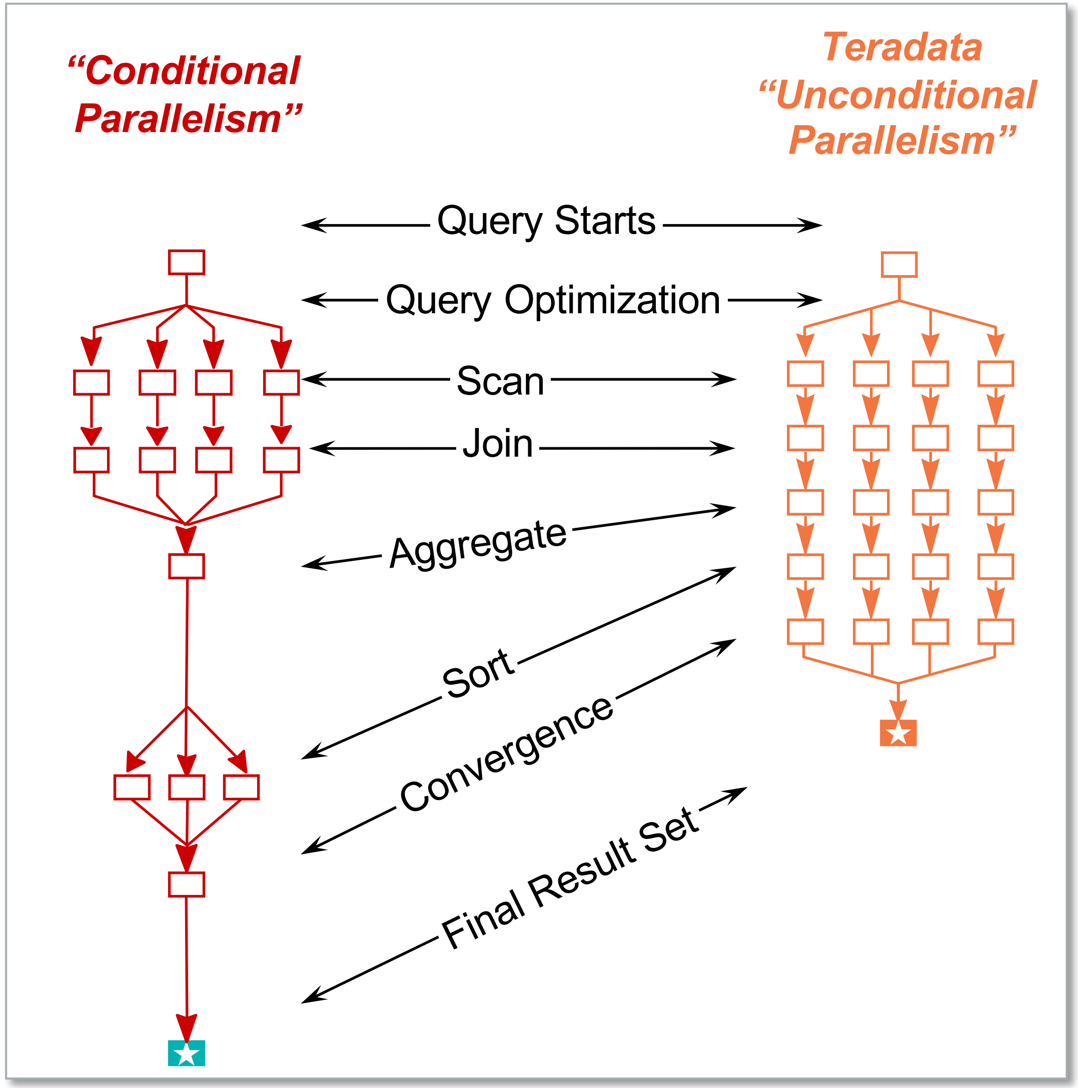


teradata.

Query Parallelization

Query Execution

- Query parsing, management is fully distributed across the nodes
  - No head node/coordinator node
- All operations fully parallel
  - No single threaded operations
  - Scans, Joins, Index access, Aggregation, Sort, Insert, Update, Delete
  - Ordered Analytics
  - Extensibility functions
  - Result return

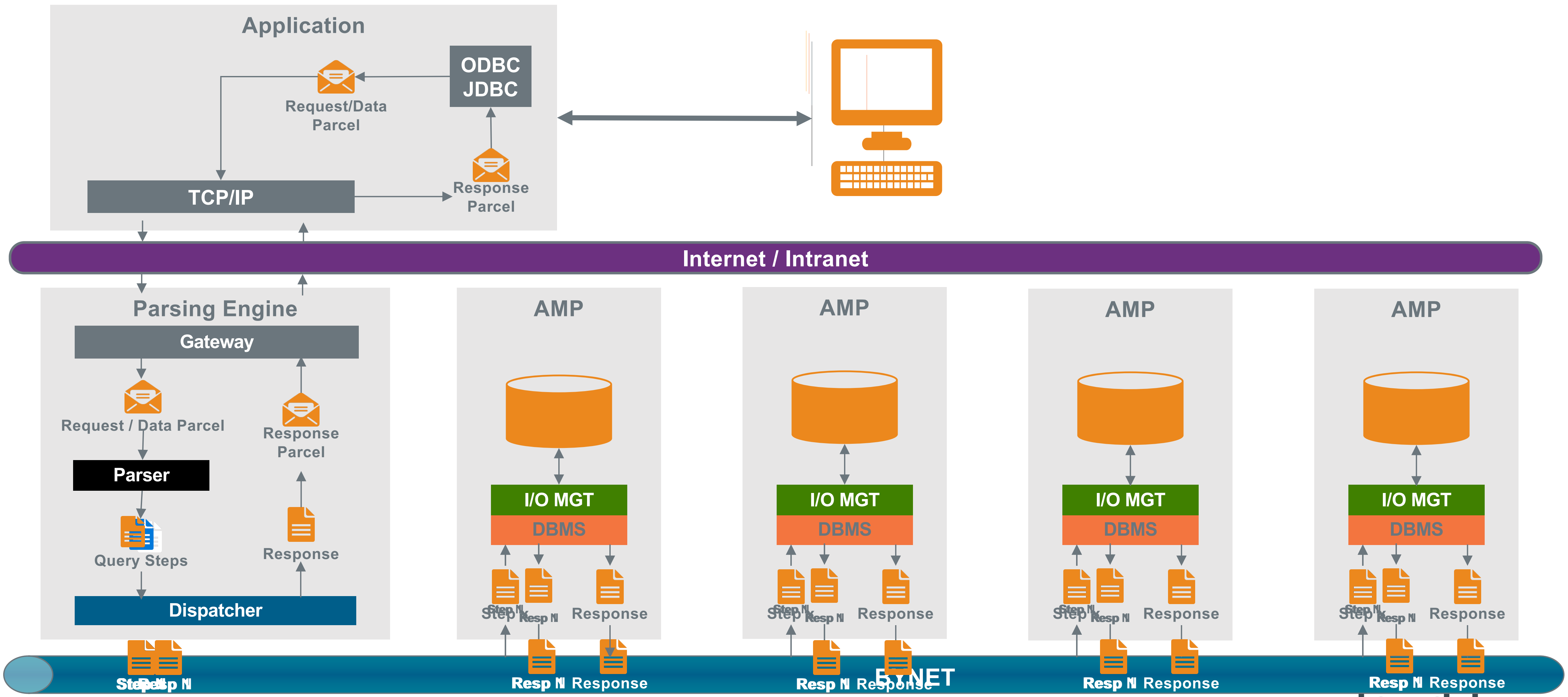


teradata.

Query Execution

The Life of a Teradata Query

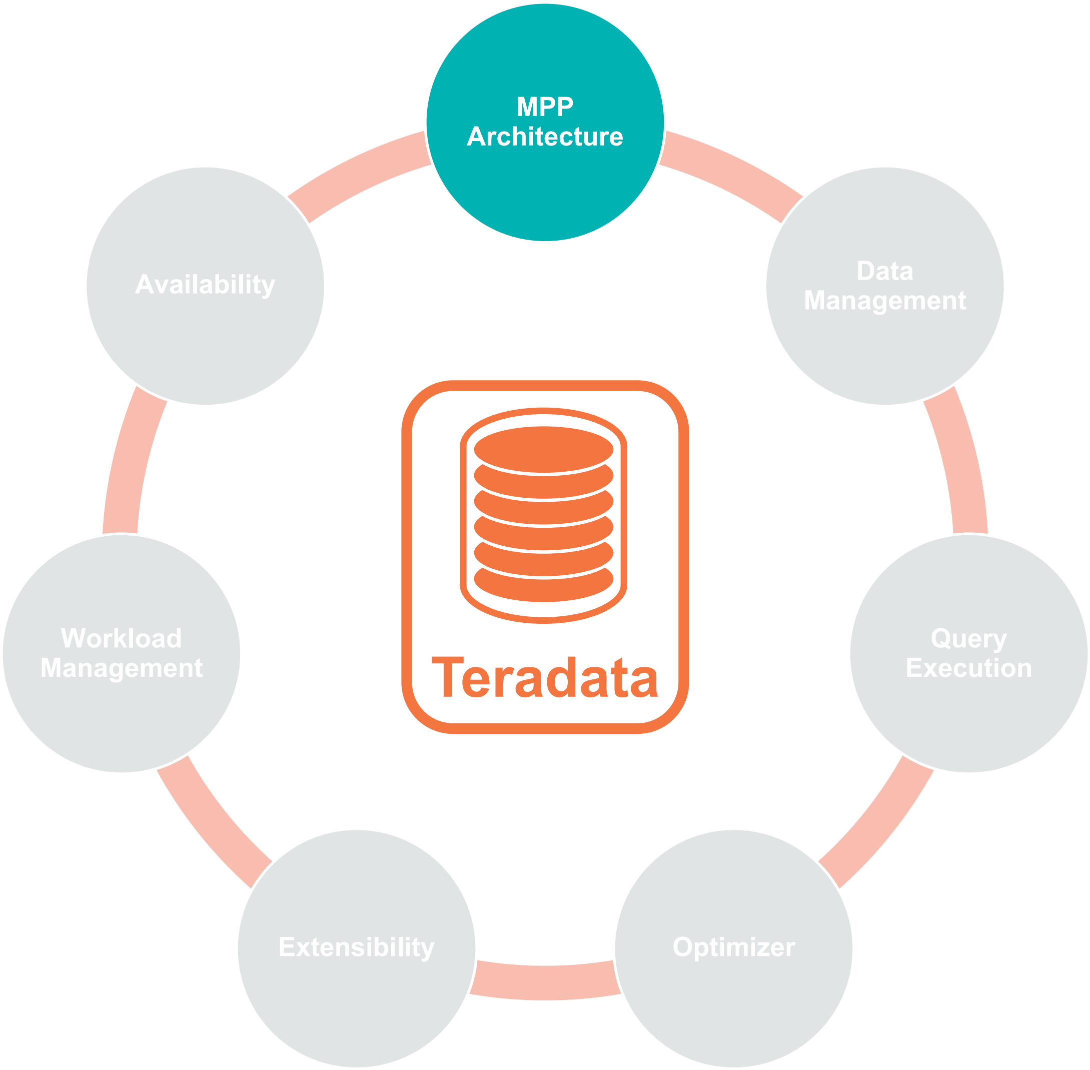
Query Execution



teradata.

What Makes Teradata Unique

Key Components/Concepts



teradata.

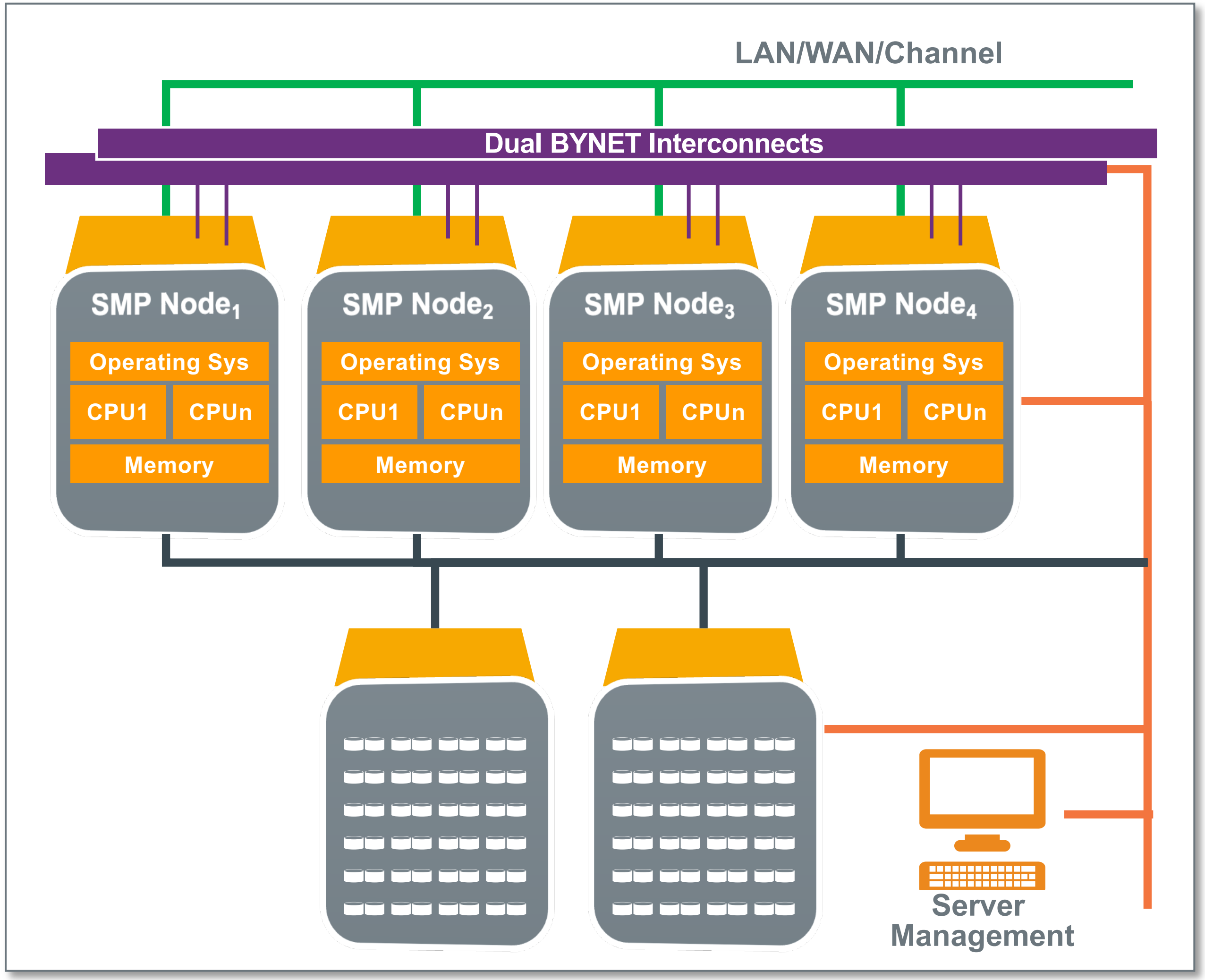
Teradata MPP Architecture
Key Concepts

- As the system scales (e.g. add more nodes), it should be no additional administrative overhead.
- As it scales, there should be linear improvement of performance.
- The rest of the world is catching on to our MPP concept. We've done it from the day one.
- Teradata no longer assumes that we always provide the hardware/platform that our database runs on.

Teradata MPP Server Architecture

MPP Architecture

- Nodes
  - Incrementally scalable to 2048 nodes
- Operating System
  - Linux (SUSE)
- Storage
  - Independent I/O
  - Scales per node
- BYNET Interconnect
  - Fully scalable bandwidth
- Connectivity
  - Fully scalable
  - Traffic spread across all nodes
  - Channel – ESCON/FICON
  - LAN, WAN
- Server Management
  - One console to view the entire system



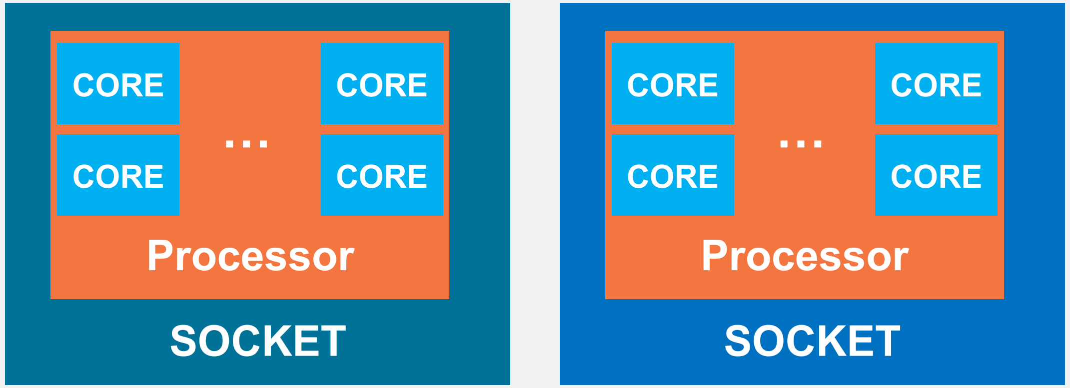
teradata.

Industry-Leading Intel® Xeon Technology

MPP Architecture

Teradata leverages leading Intel technology to deliver industry-leading performance:

- Leverage Moore's Law and use massive numbers of transistors to build multiple core processors
- High-performance Intel Xeon processors
  - We stay very current with the Intel roadmap



For example:

- Twelve-Core Processor
- Two Processors = 24 Cores/node
- Two Hyper-threads per Core = 48 virtual cores

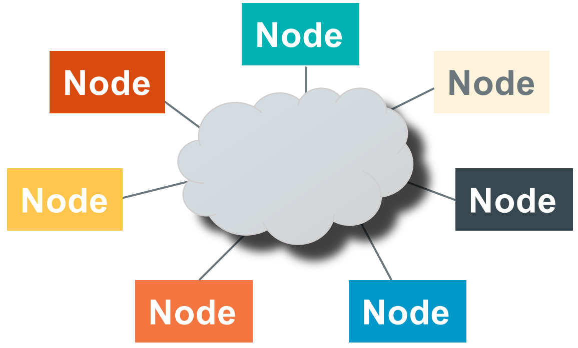
BYNET™ Software

MPP Architecture

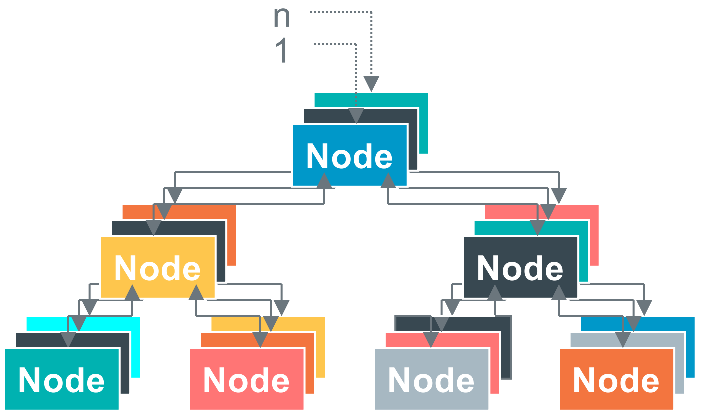
- **Optimized for Teradata Active Data Warehouse Performance**
  - Linearly scalable bandwidth
  - BYNET Low Latency Interface (BLLI) – Streamlined communication protocol
  - BYNET software provides unique Teradata features: broadcast message support, row merge support, multi-fabric message traffic shaping, software guaranteed message delivery (point to point and broadcast)
  - Broadcast reliability is implemented in software in the form of multiple virtual broadcast trees.
  - Supports several physical fabrics; InfiniBand , Ethernet (1Gb & 10Gb), Native BYNET.
- **Proven High-Availability**
  - Each fabric is fault tolerant (multiple paths, redundant power & cooling)
  - 2 active and independent fabrics (no single point of failure)

The **Teradata Optimizer** chooses between Point-to-Point and Broadcast Messaging to select the most effective communication.

Multiple Simultaneous Point-to-Point Messaging



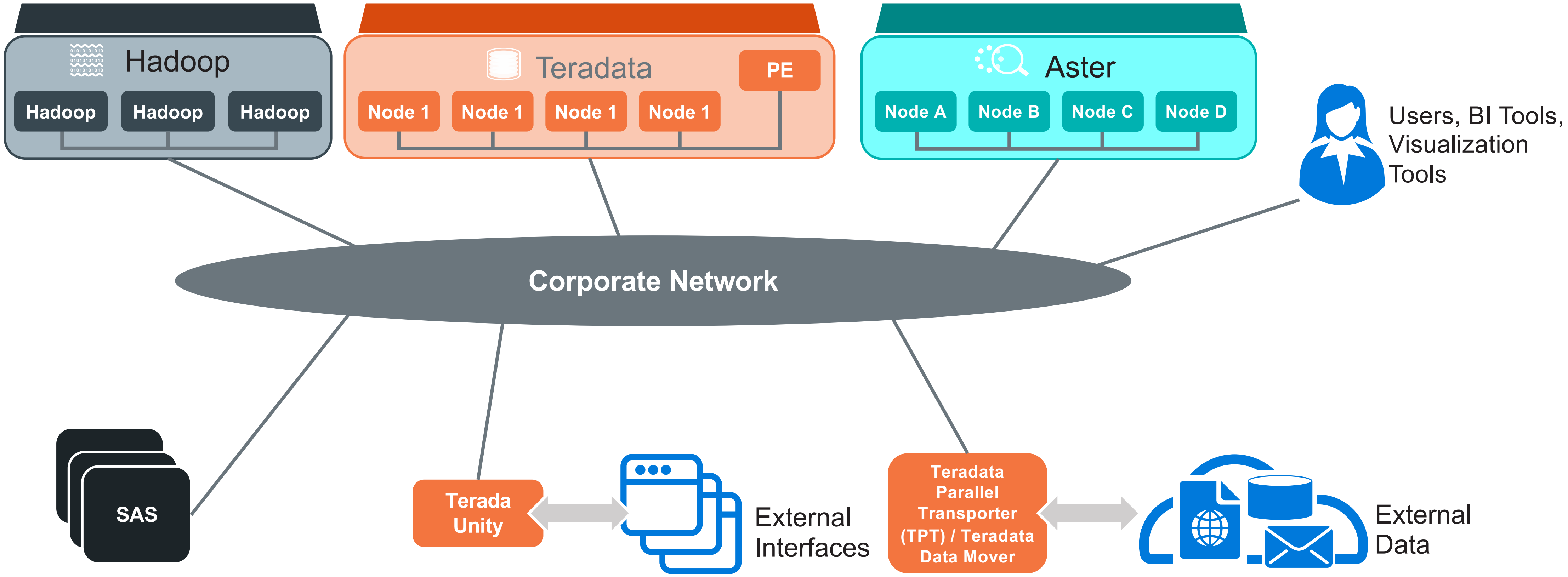
Broadcast Messaging (Multiple Virtual Broadcast Trees)



teradata.

Network Connected Teradata Unified Data Architecture

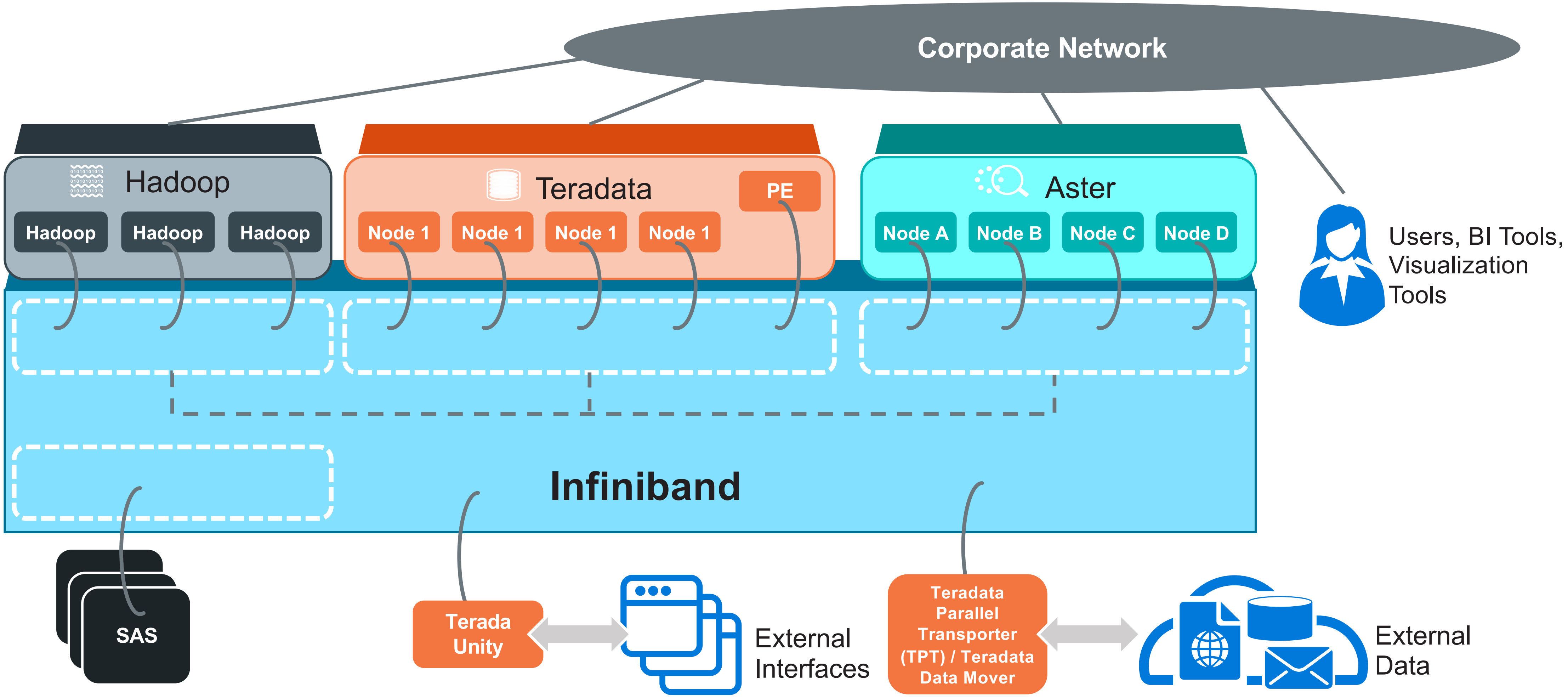
MPP Architecture



Infiniband

Integrated Communication for Teradata Unified Data Architecture

MPP Architecture

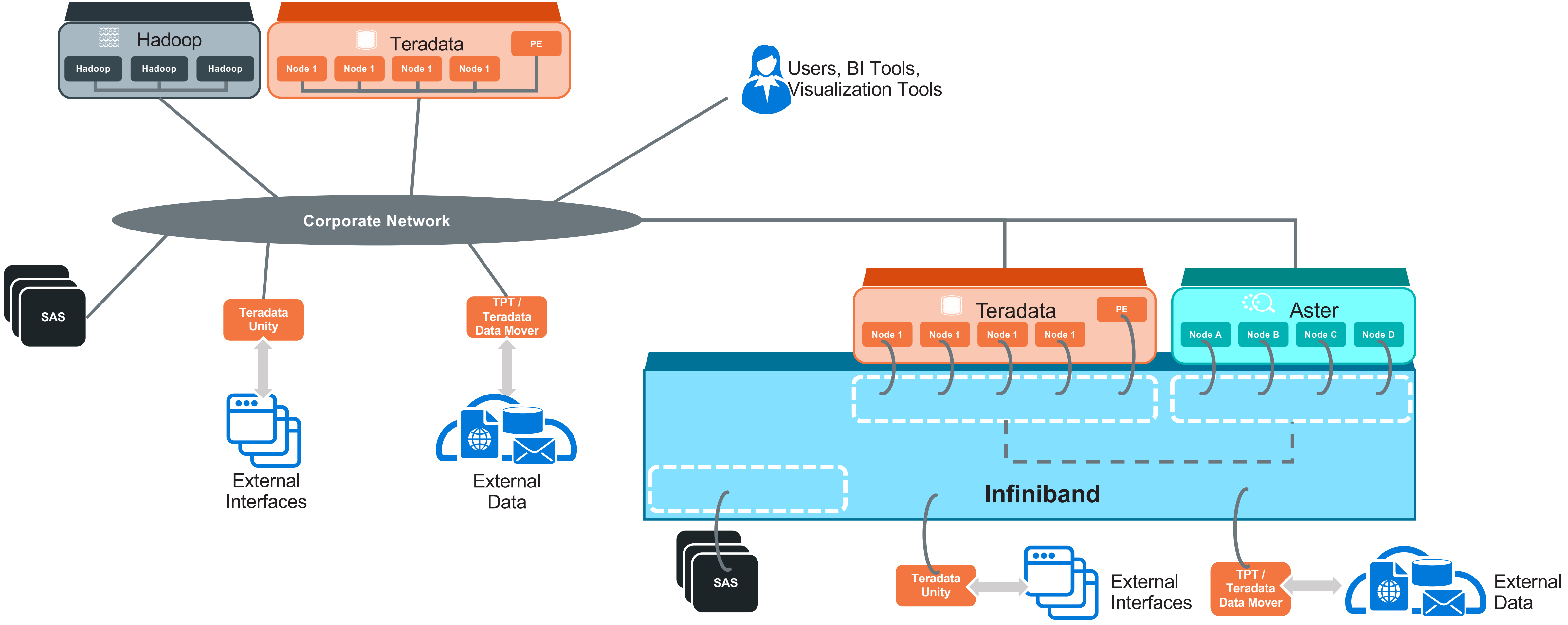


teradata.

Infiniband

Hybrid Architecture

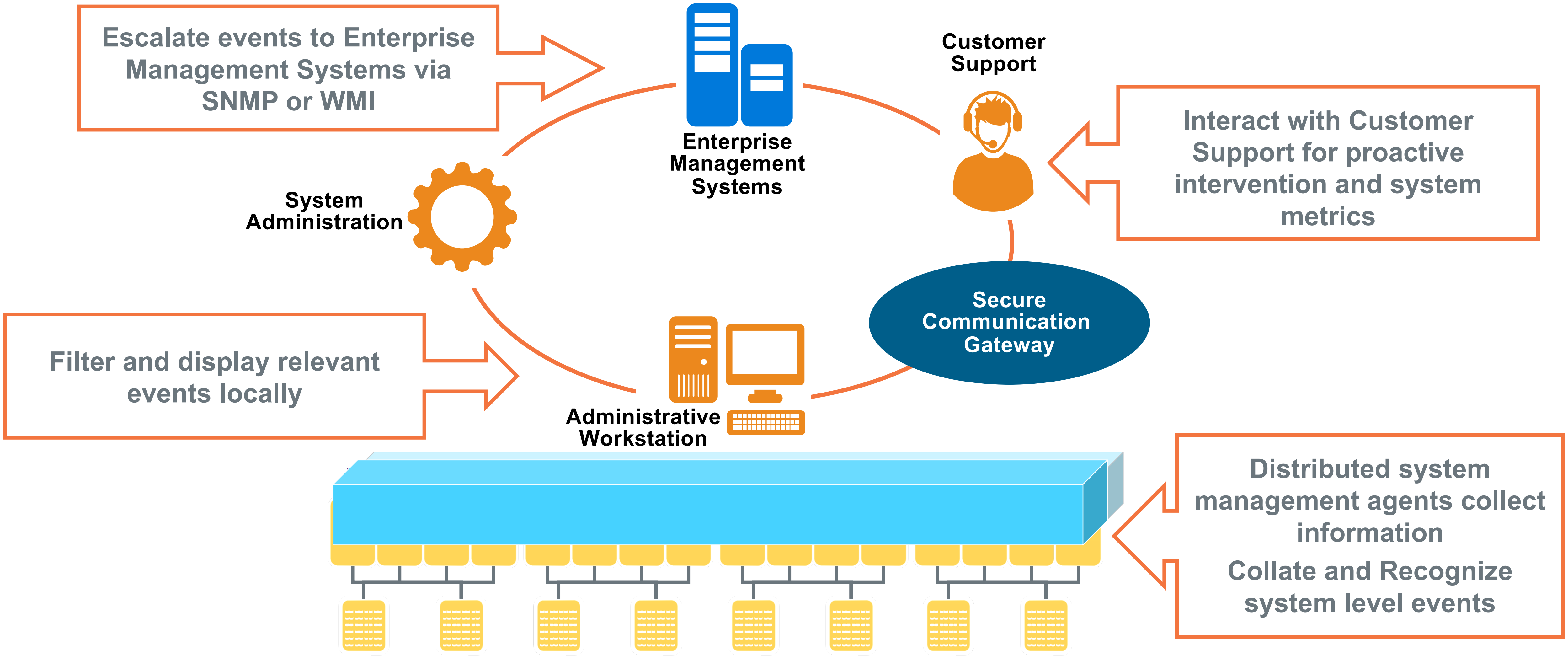
MPP Architecture



teradata.

Teradata Systems Management

MPP Architecture



teradata.

Summary

Teradata is Teradata

MPP Architecture

Teradata Workload-Specific Platform Family

On-Premises

Teradata Cloud

Managed Cloud

Teradata Database on VMware®

vmware®

Virtual Environment

Teradata Database on …

Microsoft Azure

Google Cloud Platform

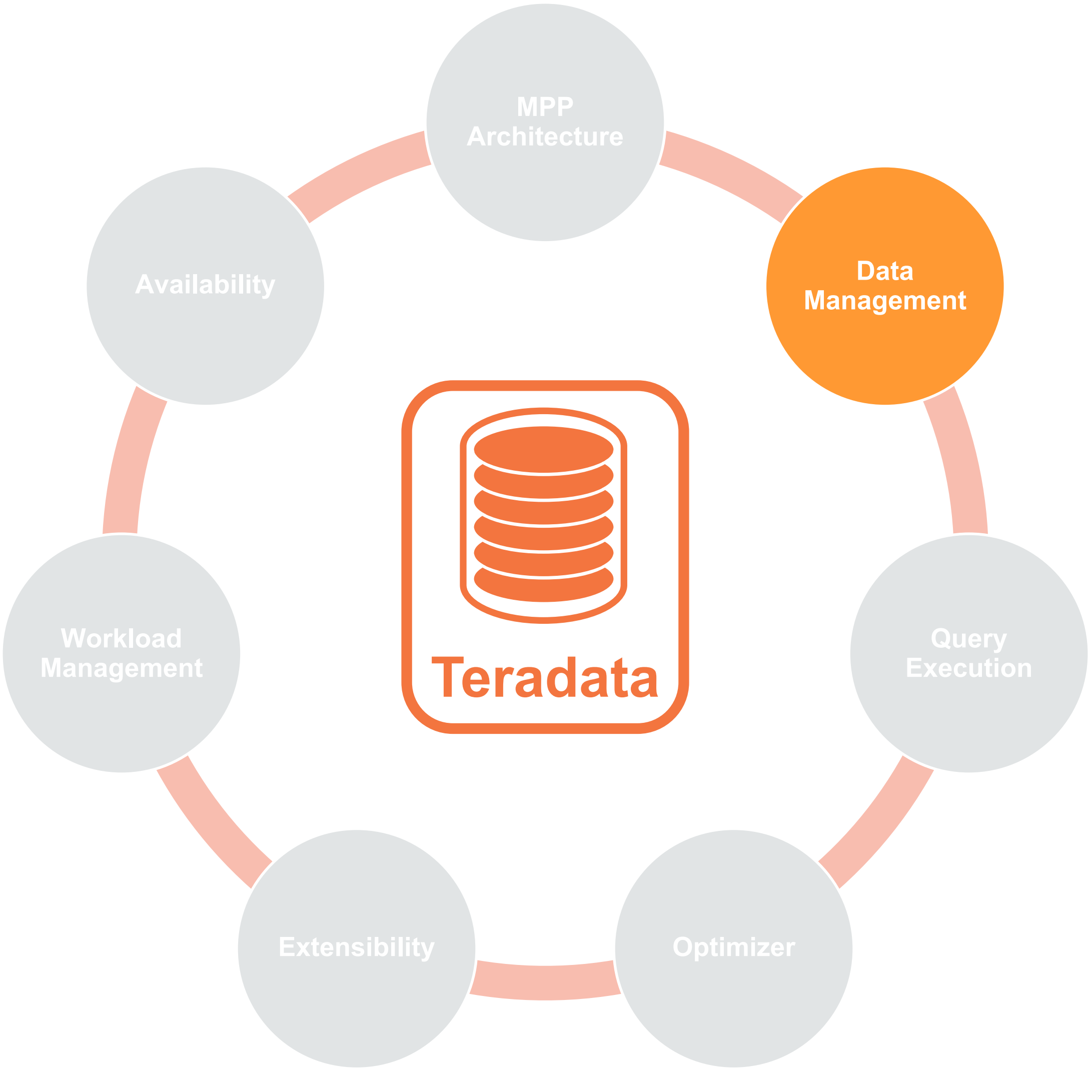
amazon web services™

Public Cloud

teradata.

What Makes Teradata Unique

Key Components/Concepts



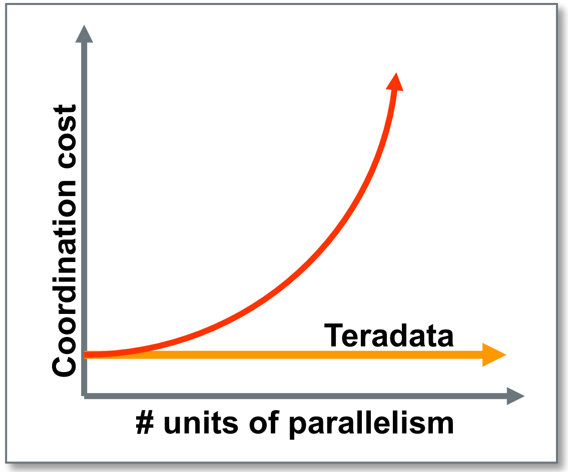
teradata.

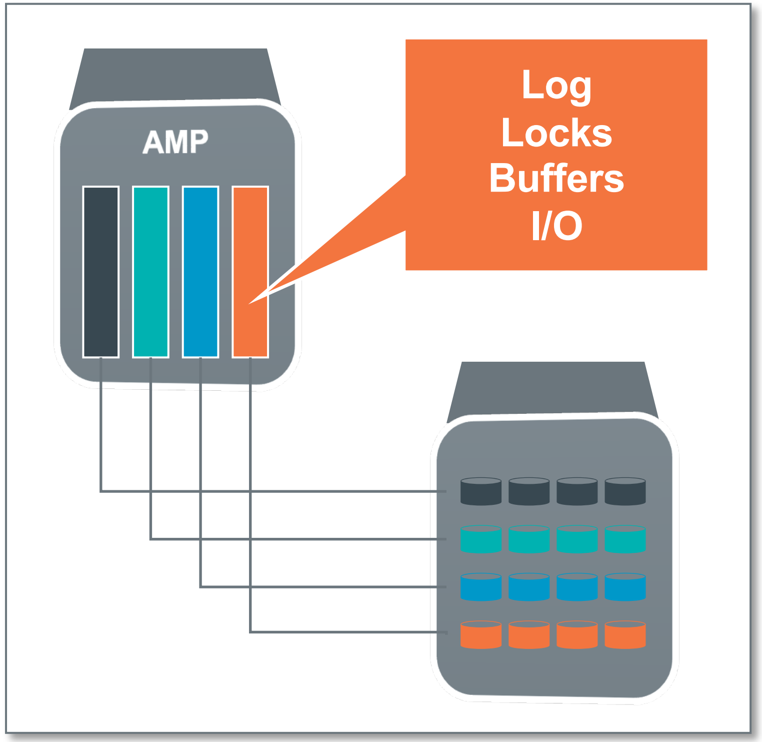
Shared Nothing – Managing the Data

Key Concepts

Data Management

- Basis of Teradata scalability
  - Each AMP owns an equal slice of the disk
  - Only that AMP reads that slice
- No single point of control for any operation
  - I/O, Buffers, Locking, Logging, Dictionary
  - Nothing centralized
  - Exponential communication costs avoided





teradata.

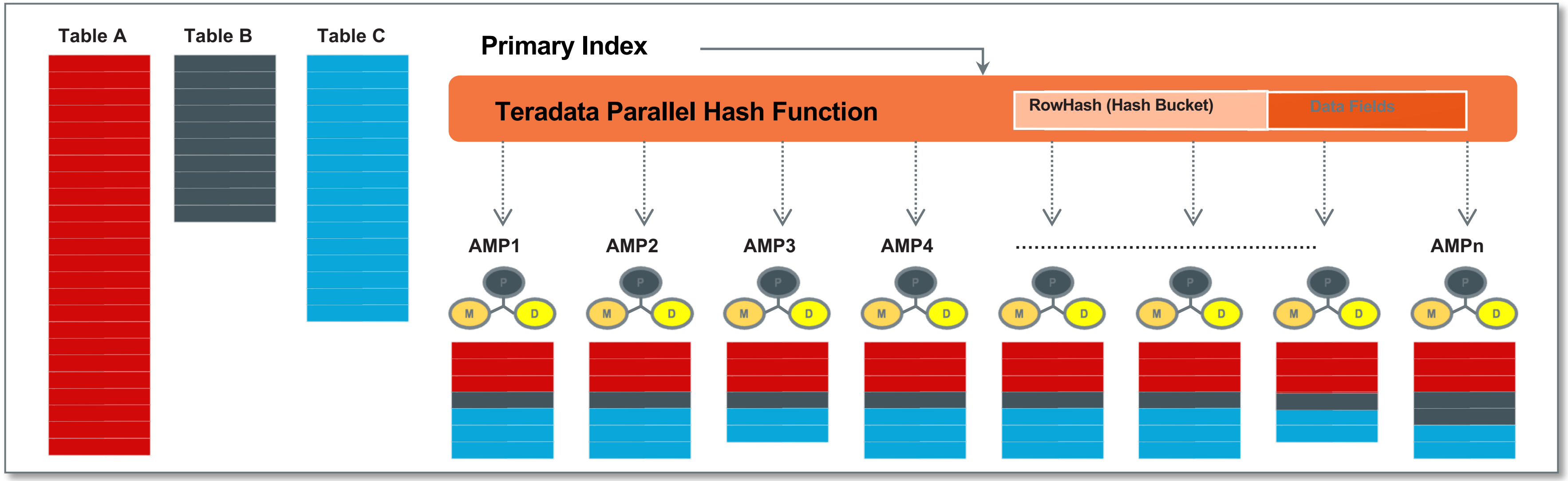
Teradata Hash Maps - Basics

- A Teradata Hash Map
  - A hash map tells Teradata where data belongs in a parallel system.
- Primary Index data values are hashed
  - Hashing algorithm
  - Divide by 1M hash buckets
- AMPs own Hash Buckets
  - The "Hash Map" maps each of the 1M buckets to whatever number of AMPs exist within the system.
  - Rows are assigned to the AMPs via the hash bucket
- Spreads data evenly
  - For performance
  - Reduce skew
  - Enables co-located table rows

Teradata Data Management

Rows automatically distributed evenly by **hash partitioning**

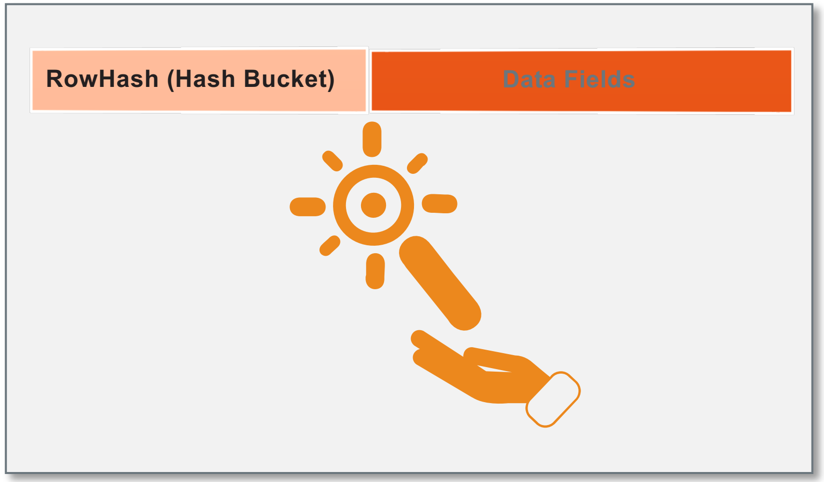
- Even distribution results in scalable performance
- Done in real-time as data are loaded, appended, or changed.
- Hash map defined and maintained by the system
  - 2**32 hash codes, 1,048,576 buckets distributed to AMPs
- Primary Index (PI) column(s) are hashed
- Hash is always the same - for the same values
- No reorgs, repartitioning, space management



The Magic of Row Hash

Data Management

- Store rows in a table in row hash order (logically)
- Direct access via Primary Index
  - For continuous update or single lookup
  - Hash the value
  - Go direct to AMP (routed by BYNET)
  - MI (in memory) -> CI -> data block
  - Very sparse N* tree
  - MI and CI very small structures
  - Guarantees no more than 2 IOs to get to any record via PI
  - CI cached if frequent access
  - “First Index is free”
- Sort merge join with no sort
  - Joins which never require scan of the large table
  - E.g. Star Join doesn’t need to hash join or spool the fact table and sort it to join



Defining a Table in Teradata

Create the table

- Standard SQL syntax

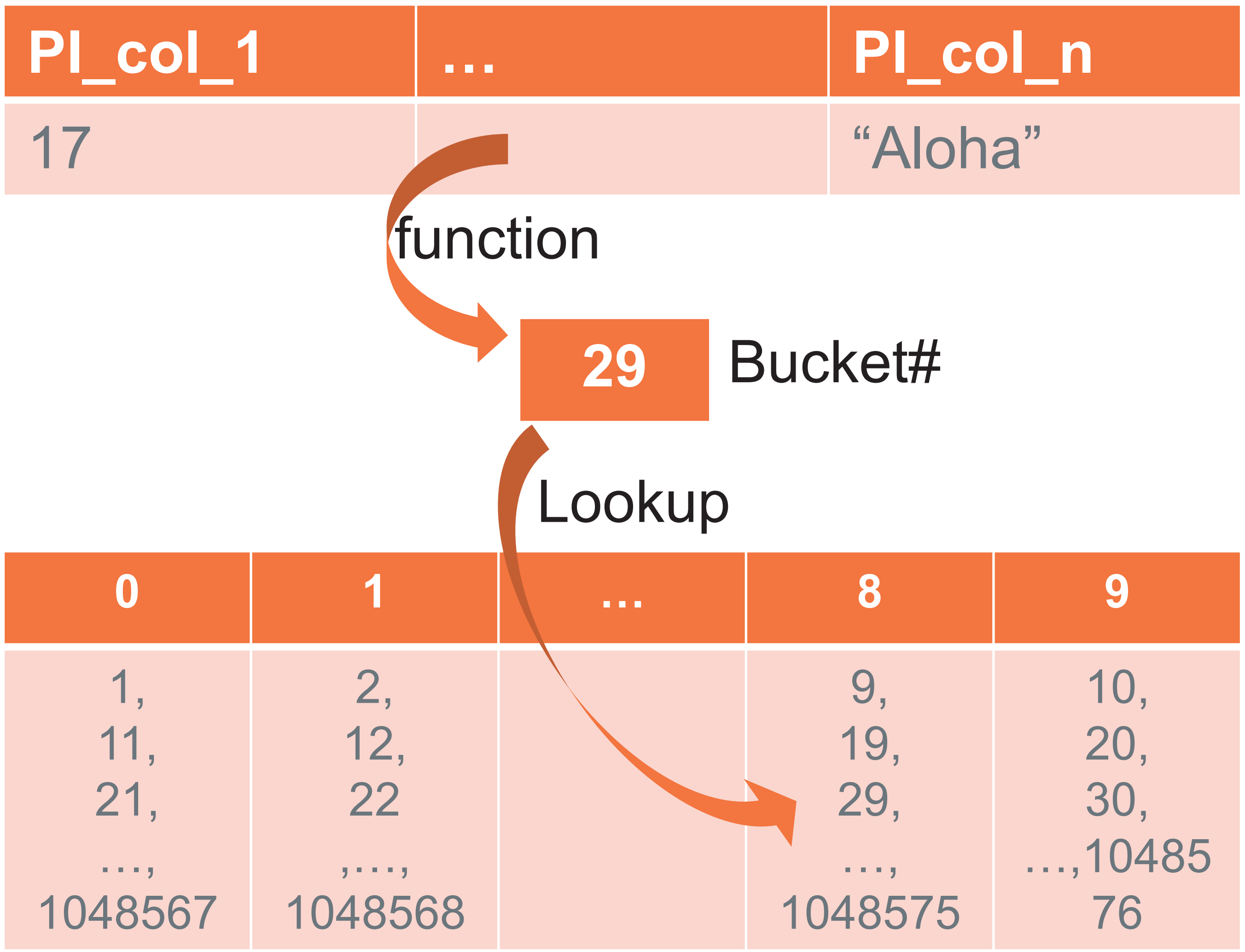
Define the primary index

- Extra line at end of table definition

```
CREATE TABLE LineItem (
        OrderKey INTEGER NOT NULL,
        PartKey INTEGER NOT NULL,
        SupplierKey INTEGER,
        LineNumber INTEGER,
        Quantity INTEGER NOT NULL,
        ExtendedPrice DECIMAL(13,2),
        Discount DECIMAL(13,2),
        Tax DECIMAL(13,2),
        Comment VARCHAR(50)
        )
PRIMARY INDEX (OrderKey);
```

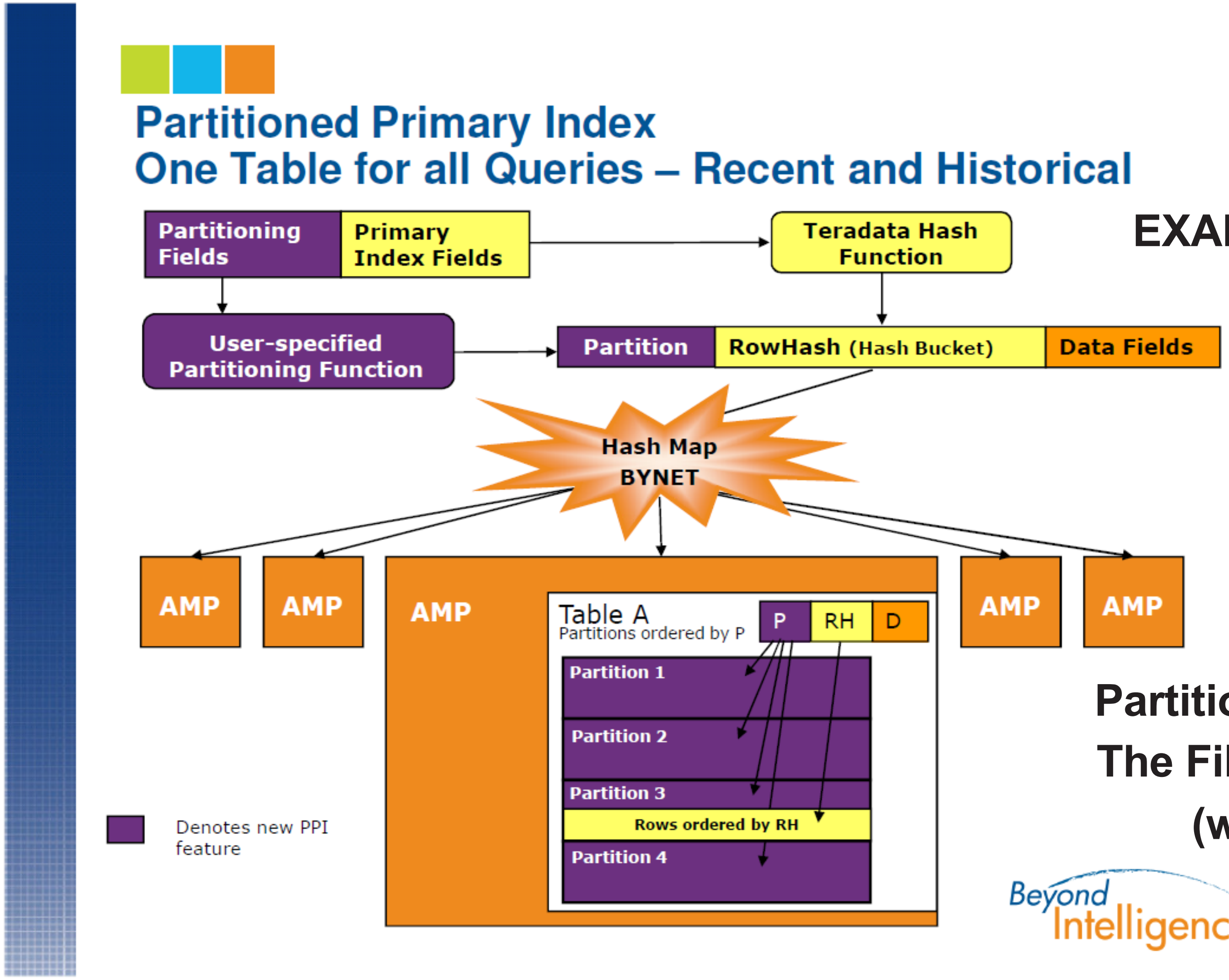
Teradata Hash Maps - Basics

- Teradata Hash Maps
  - Key to the distribution of a table in an MPP System
  - The “PI” Fields are concatenated and run through a mathematical function that returns a “bucket”, a number between 1 and 1,048,576 (20 bits)
  - The “Hash Map” maps each of the buckets to whatever number of AMP’s exist within the system



teradata.

Partitioned Primary Index (PPI)



EXAMPLE: PARTITION BY "MONTH"

Partition plays NO role with regards to which AMP owns the Data

Partition does play a role in establishing The File System Logical Row Sort Order (which is: Partition, then Hash)

Beyond Intelligence

teradata.

File System

Data Management

- Teradata wrote a new rule book
    - Old one written by IBM 40 years ago, used by most mainstream DBMSs today - except Teradata
- File system built of raw slices
- Rows stored in blocks
    - Variable length
    - Grow and shrink on demand
    - Rows located dynamically
        - May be moved to reclaim space, defrag
    - Maximum block size is configurable
        - System default or per table
        - 8K to 1MB
        - Change dynamically
- Blocks stored in 12MB allocation units - “cylinders”
- Indexes are just rows in tables

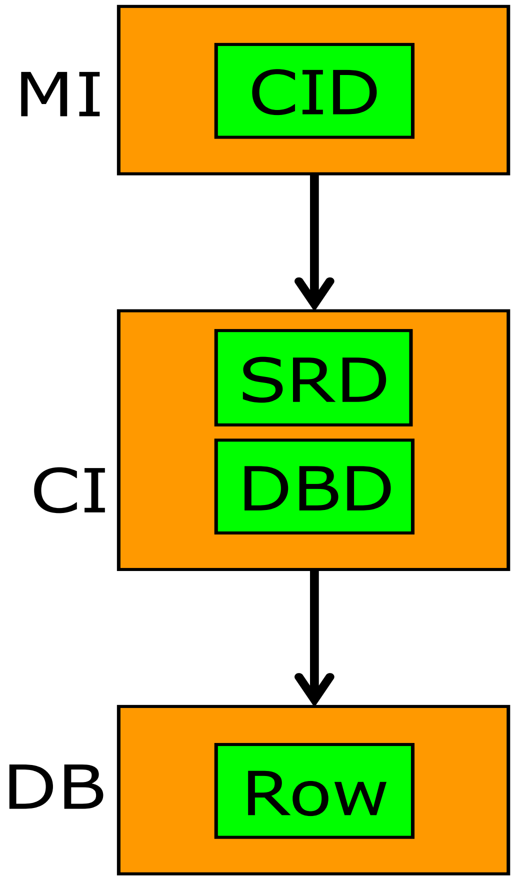
File System Structure

File System Index Structure

- B*Tree

- MI — Master Index
  - CID — Cylinder Index Descriptor
- CI — Cylinder Index
  - SRD — Subtable Refarray Descriptor
  - DBD — Data Block Descriptor
- DB — Data Block
  - ROW — The actual physical row

- 3 level physical tree
  - But MI always in memory



teradata.

LOGICAL ROW ADDRESSES

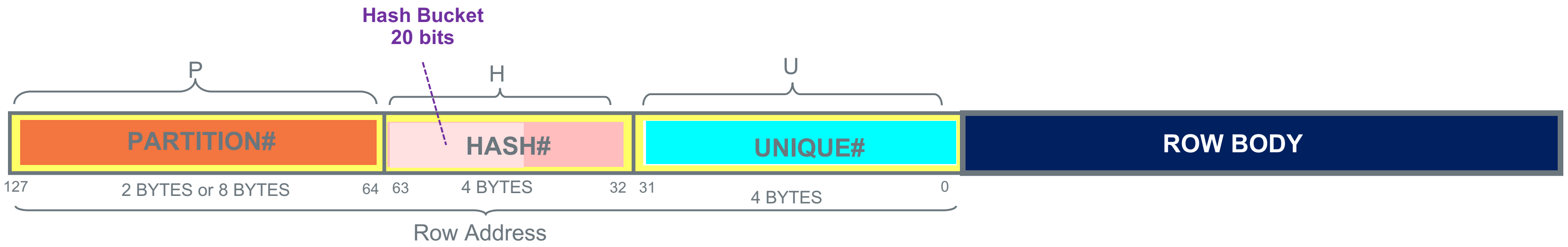


Table Row Address:

- Usage of optional Partition# field is dependent upon row-usage
  - Traditional Primary Index Table Rows: doesn’t use partition# field
  - PPI-2 table rows : uses two bytes
  - PPI-8 table rows : uses 4 bytes
- Usage of the Hash# field:
  - For table rows, this typically contains computed hash associated with the primary index
- Usage of the Unique# field:
  - For table rows, this is utilized to assign unique row address for the case in which there are matching hash values.
- IMPORTANT!: For partitioned tables, a “partition boundary” is a “row address boundary”, not a physical storage boundary. Partitions may cross data block and/or cylinder boundarys

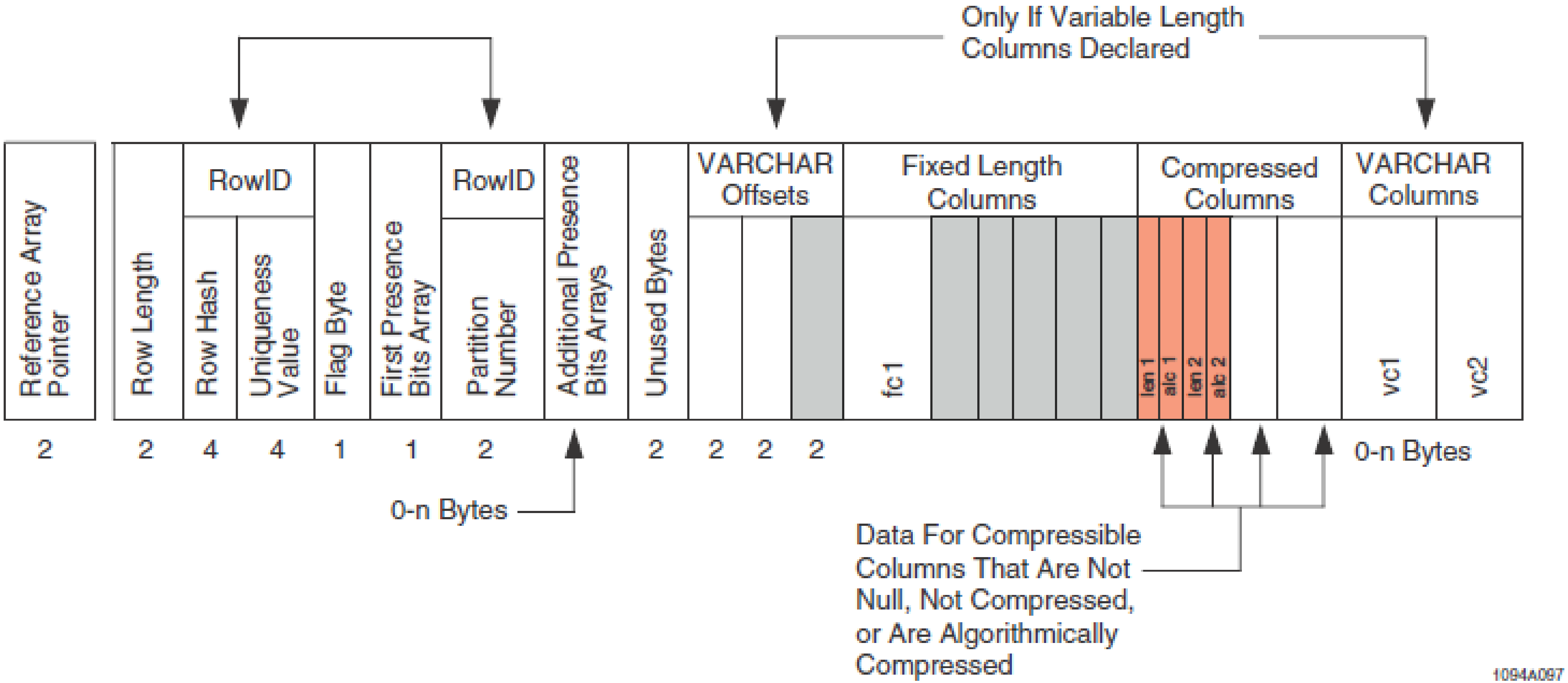
Spool File Row Address:

- May have table-like row address assignments or artificially assigned row addresses

Special Case RTREE-index sub-table rows

- Partition3(4 BYTES) and Hash#(4 BYTES) fields used as a single 8-BYTE field containing the “Virtual Block Identifier”
- Unique# field used to identify rows within a given Virtual Block

Base Table Row Formats



teradata.

Space Allocation

Data Management

- Space allocation is entirely dynamic
  - No tablespaces or journal spaces or any pre-allocation
  - Spool (temp) and tables share space pool, no fixed reserved allocations
- If no cylinder free, combine partial cylinders
  - Dynamic and automatic
  - Background compaction based on tunable threshold
- Quotas control disk space utilization
  - Increase quota (trivial online command) to allow user to use more space

teradata.

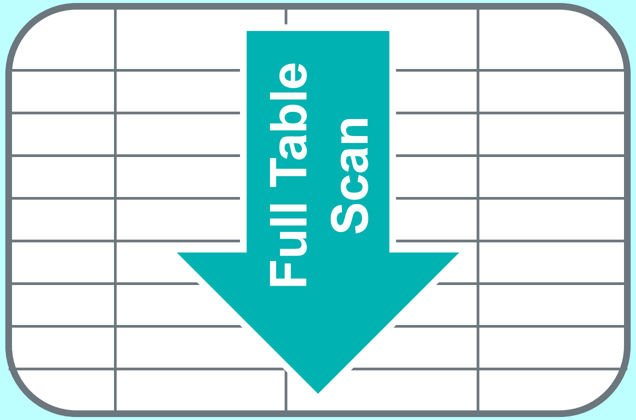
Partitioned Primary Index (PPI)

Comparison

Data Management

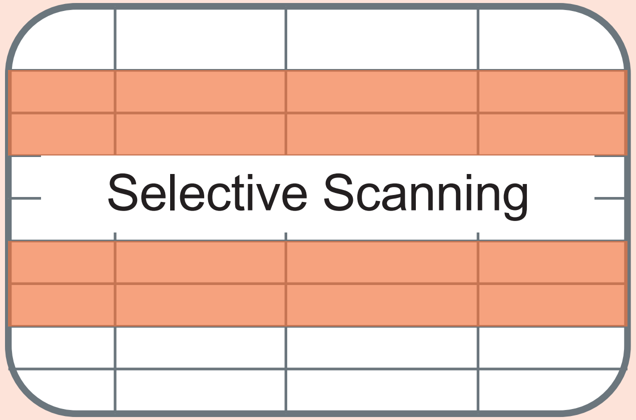
No Partitions

Looking for a customer's transactions across all time and states



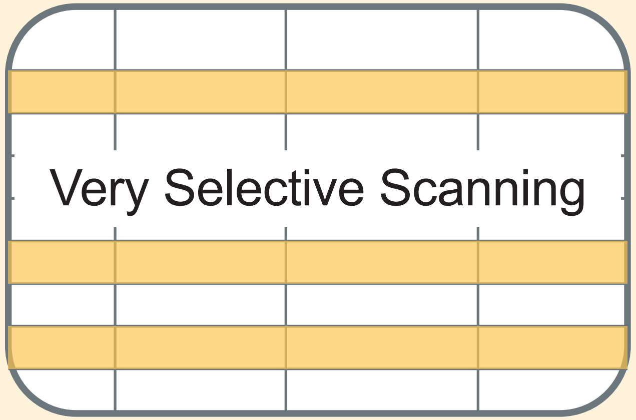
Single Level Partition

Looking for two particular weeks



Multi-level Partitions

Looking for two states in two weeks



teradata.

Teradata Columnar

Data Management

Row Store – Employee Table

Emp ID	Lastname	Firstname	Salary
1	Smith	Joe	40000
2	Jones	Mary	50000
3	Johnson	Cathy	44000

Column Store – Employee Table

Emp ID	Lastname	Firstname	Salary
1	Smith	Joe	40000
2	Jones	Mary	50000
3	Johnson	Cathy	44000

- Key advantages:
  - Better query **performance** – read only the columns you need
  - Smaller **data size** – better compression on homogeneous column values
- Query execution optimized for column processing
- Full hybrid columnar
  - Row tables, column tables, mixed tables, and mixed index and table
  - Physical design optimization option

Multi-Level Partitioned Primary Index (PPI)

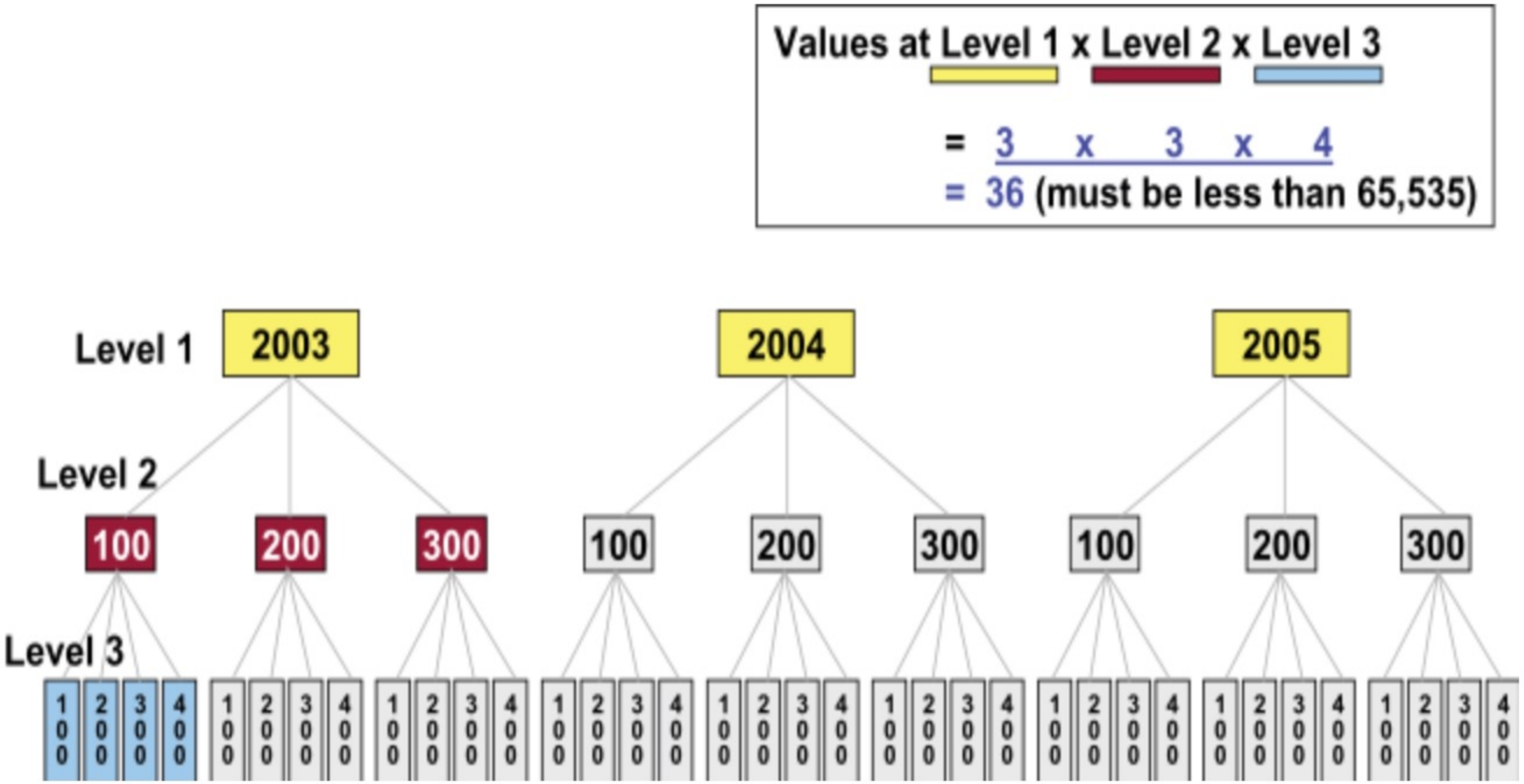
Data Placement

```
CREATE TABLE Sales
    (storeid INTEGER NOT NULL,
     productid INTEGER NOT NULL,
     salesdate DATE FORMAT 'yyyy-mm-dd' NOT NULL,
     totalrevenue DECIMAL(13,2),
     totalsold INTEGER,
     note VARCHAR(256))
UNIQUE PRIMARY INDEX (storeid, productid, salesdate)
    PARTITION BY (
      RANGE_N(salesdate BETWEEN DATE '2003-01-01' AND DATE '2005-12-31'
                                EACH INTERVAL '1' YEAR),
      RANGE_N(storeid BETWEEN 1 AND 300 EACH 100),
      RANGE_N(productid BETWEEN 1 AND 400 EACH 100));
```

Input to partition function				Sales					
L1	L2	L3		salesdate	storeid	productid	total revenue	totalsold	note
1	1	1		2003-04-15	96	10	4158	42	Good day
1	1	2		2003-07-06	71	184	1972	68	Marginal
2	2	3		2004-11-09	175	241	3055	47	Slow day
1	1	4		2003-12-24	82	363	1261	13	Promotion

Multi-Level Partitioned Primary Index (PPI)

Data Placement



How Rows for the Sales Table Are Grouped on an AMP

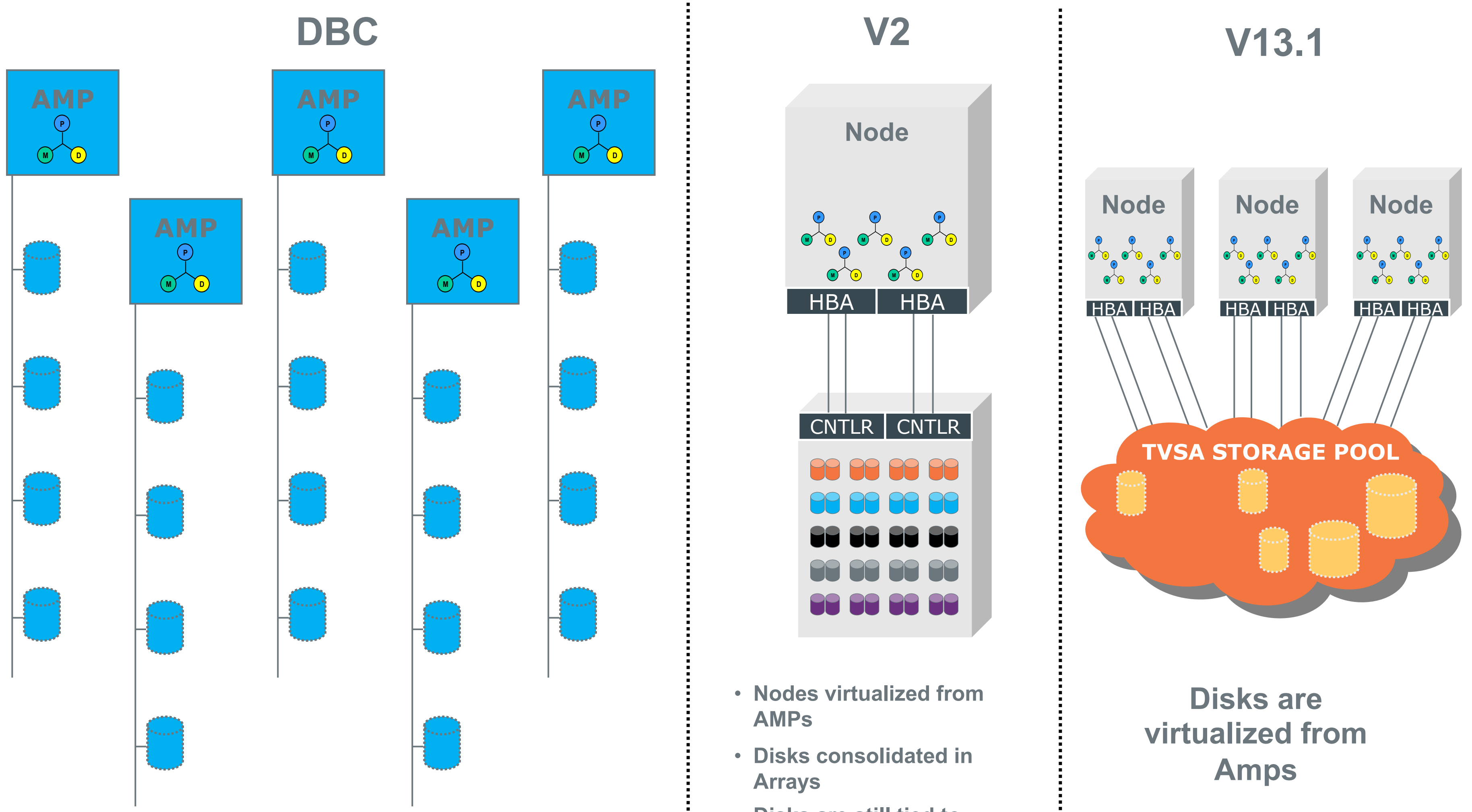
Compression

Data Management

	Multi Value	Algorithmic	Data Block
What it is	• Dictionary-based compression that replaces values specified by the user	• Make your own - It applies a compression / decompression algorithm to a column of data	• It compresses all types of data in a data block before it's stored on a disk
Suited for	• Data with repeated values	• Data with well-known attributes	• Data with a low frequency of access (cold data)
Type of Data	• NULLs + Common values	• UTF-16 to UTF-8, long strings	• Works on any/all data
Characteristic	• Zero CPU overhead	• Can write custom procedures	• Built-in library
Compression Ratio	• 30-40%	• 50% (2x)	• 3X or more

teradata.

TERADATA FILE SYSTEM: The Evolution of TVSA



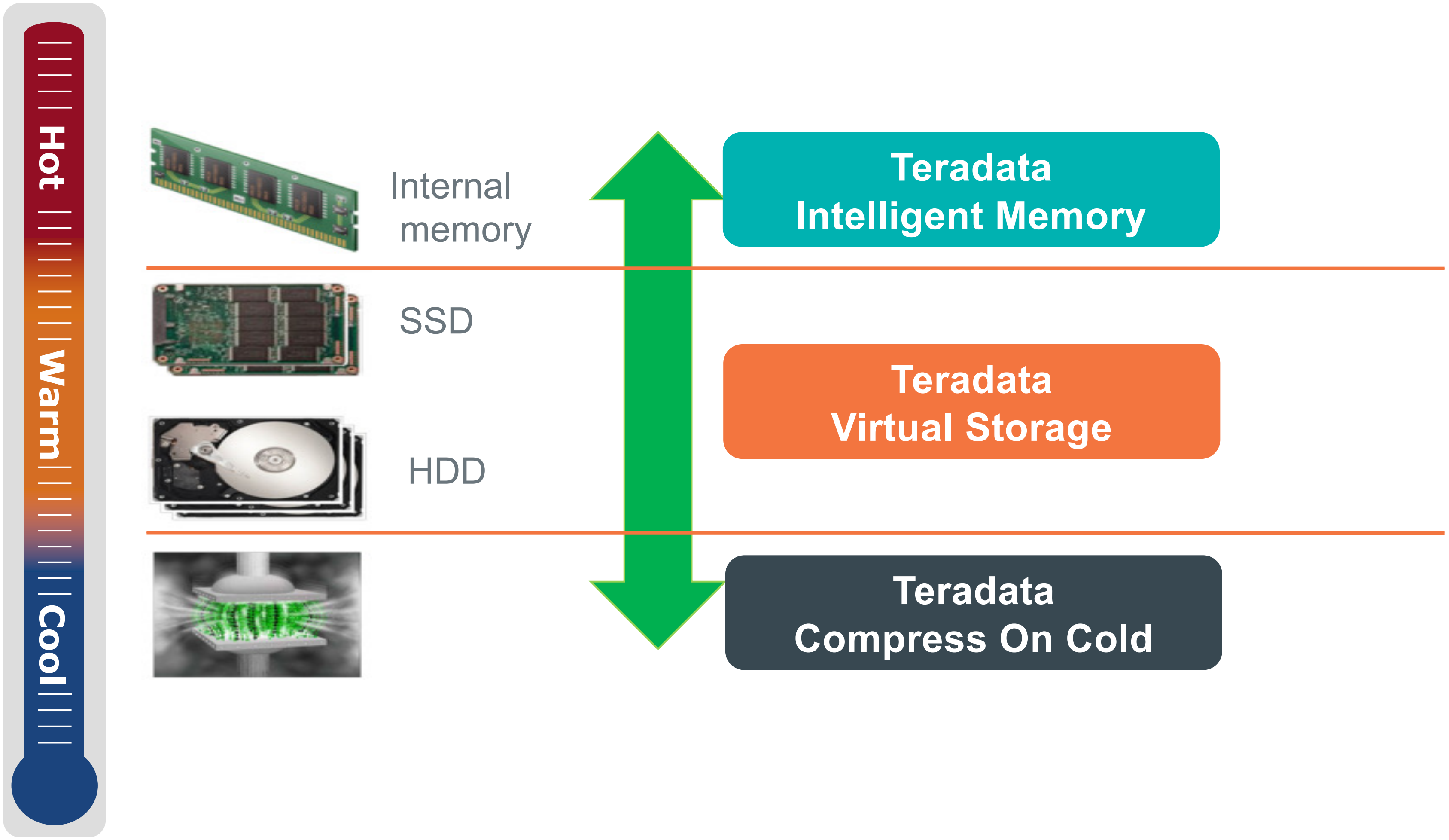
- Nodes virtualized from AMPs
- Disks consolidated in Arrays
- Disks are still tied to AMPs

Disks are virtualized from Amps

teradata.

Integrated Management of Hybrid Storage

Data Management



- Automatic
- Transparent
- No DBA effort
- No SQL changes
- Maintain user access to ALL data for analysis
- Avoid separate systems and copies of data for each use case

teradata.

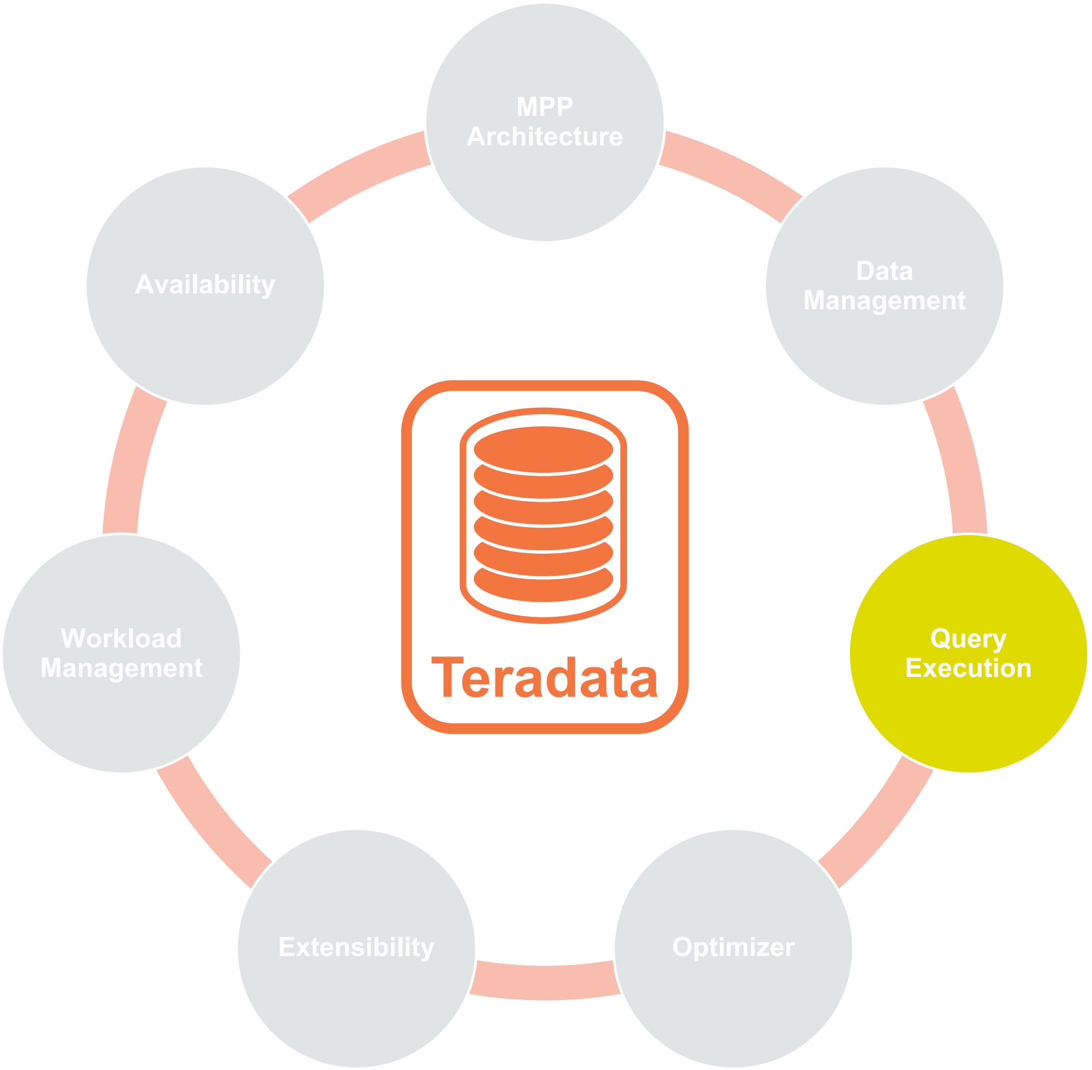
Summary
Automate and Eliminate

- Database should manage the data, not DBA or the user
- Data management should be as invisible as possible
  - Define and forget
- Enables user self-service

- No reorgs
  - Don't even have a reorg utility
- No index rebuilds
- No re-partitioning
- No detailed space management
- Easy database and table definition
- Minimum ongoing maintenance
  - All performed automatically

What Makes Teradata Unique

Key Components/Concepts



Overview: Query Execution

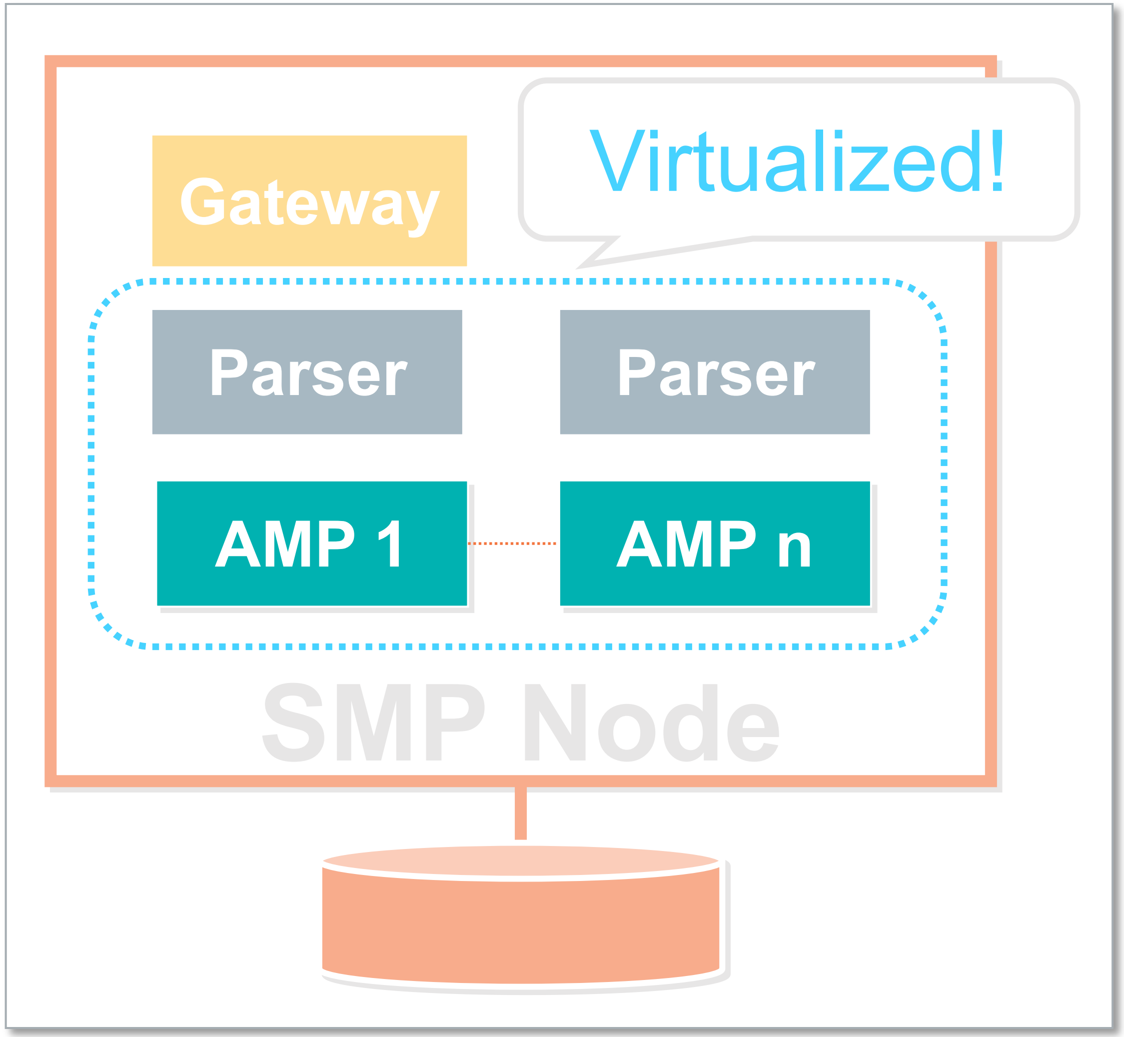
- Teradata query execution is designed to make every analytic query scalable
- Enabling in-database execution is a primary goal
- Parallelization is built-in throughout all operations and is fully automatic

teradata.

What's on a Node

Query Execution

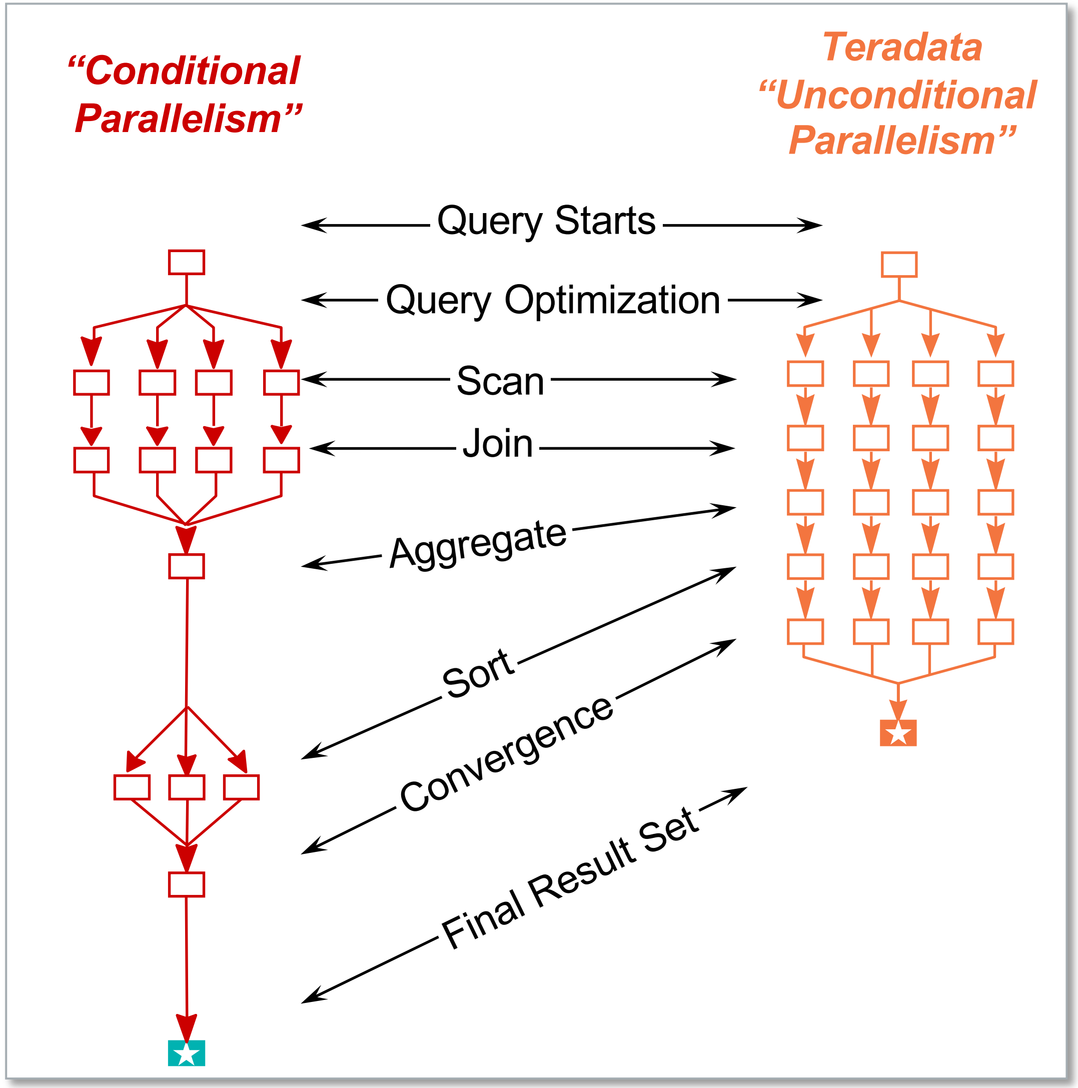
- **Gateway**
  - Connect sessions to outside world
  - Balance external traffic workload across nodes
- **Parsing Engine (PE)**
  - Parse & Optimize
  - Dispatcher to AMPs
- **AMP (Access Module Processor)**
  - Execution engine
  - Logs & locks
  - Data dictionary
  - I/O management
- **"Vprocs"**
  - Virtual "processors" sharing one physical node



teradata.

Query Parallelization

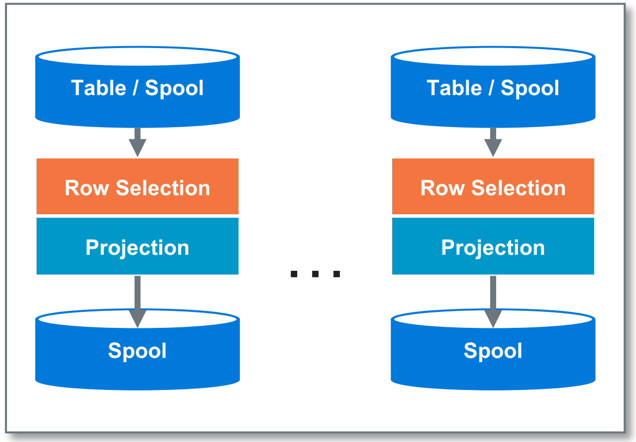
- Query parsing, management is fully distributed across the nodes
  - No head node/coordinator node
- All operations fully parallel
  - No single threaded operations
  - Scans, Joins, Index access, Aggregation, Sort, Insert, Update, Delete
  - Ordered Analytics
  - Extensibility functions
  - Result return



Data Access Methods

Query Execution

- Scan
  - Read all rows of a table
  - Executed on all AMPs simultaneously, each AMP reads the rows hashed to that AMP
  - A table or spool will be read, rows will be qualified (Row Selection)
  - Unneeded columns will be removed (Projection)
  - Result will be written to a spool file
  - Only interaction between the AMPs is for each of them to report completion
- Primary Index
  - Unique (UPI) and Non-Unique (NUPI)
  - Allows efficient direct access by value or via join
- Additional Indexes
  - Allow optimization of access
  - Used to support workloads that are high frequency, high SLA
  - Selected for use automatically by the query optimizer
  - PI, PPI and high performance scan eliminate need for many indexes



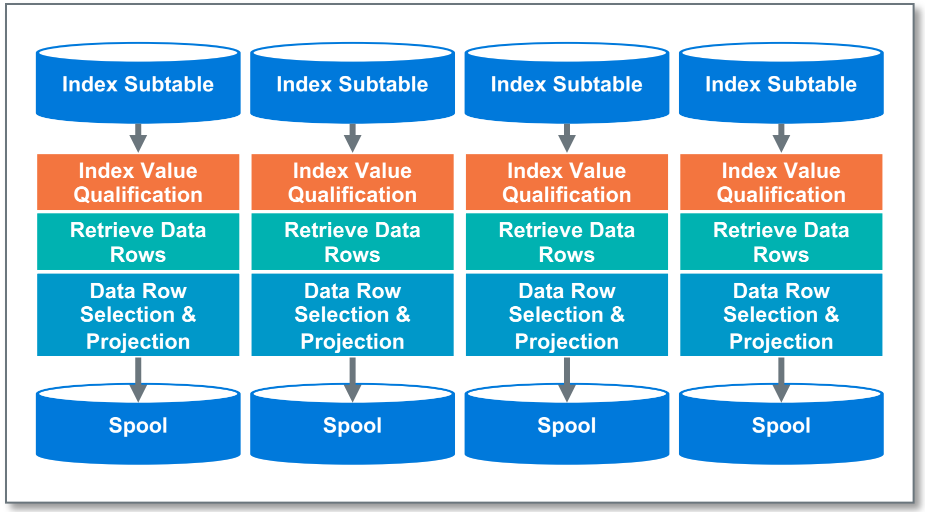
teradata.

Secondary Index: Non Unique Secondary Index (NUSI)

Query Execution

Non Unique Secondary Index (NUSI)

- Designed for accessing one or a small number of values where many rows have the same value
- Index entries are stored on the same AMP with the data rows
- An index entry is a value followed by a list of rowids of rows containing that value
- Ordered by hash of value being indexed
- Execution via a NUSI reads the index first, then uses the rowids to retrieve the data rows
  - Executed on All-AMPs simultaneously

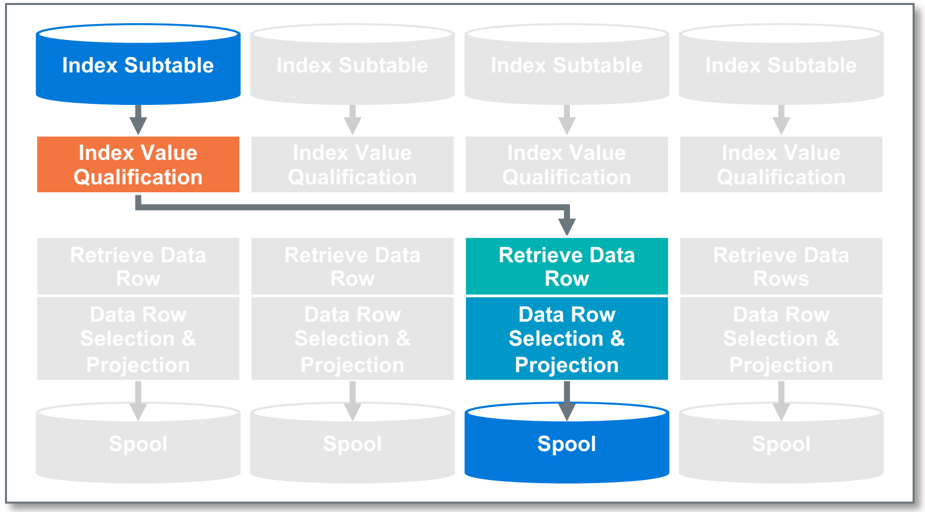


Secondary Index: Unique Secondary Index (USI)

Query Execution

Unique Secondary Index (USI)

- Designed for accessing one or a small number of rows by value
  - Enforces Uniqueness
- Index entry is stored on hash AMP of value
- Index entry contains the value and the single rowid
- Execution via a USI accesses the index entry on the hash AMP(s) of the index value(s), then sends a message to the AMP of the rowid to retrieve the row
  – Executed on the minimum number of AMPs possible



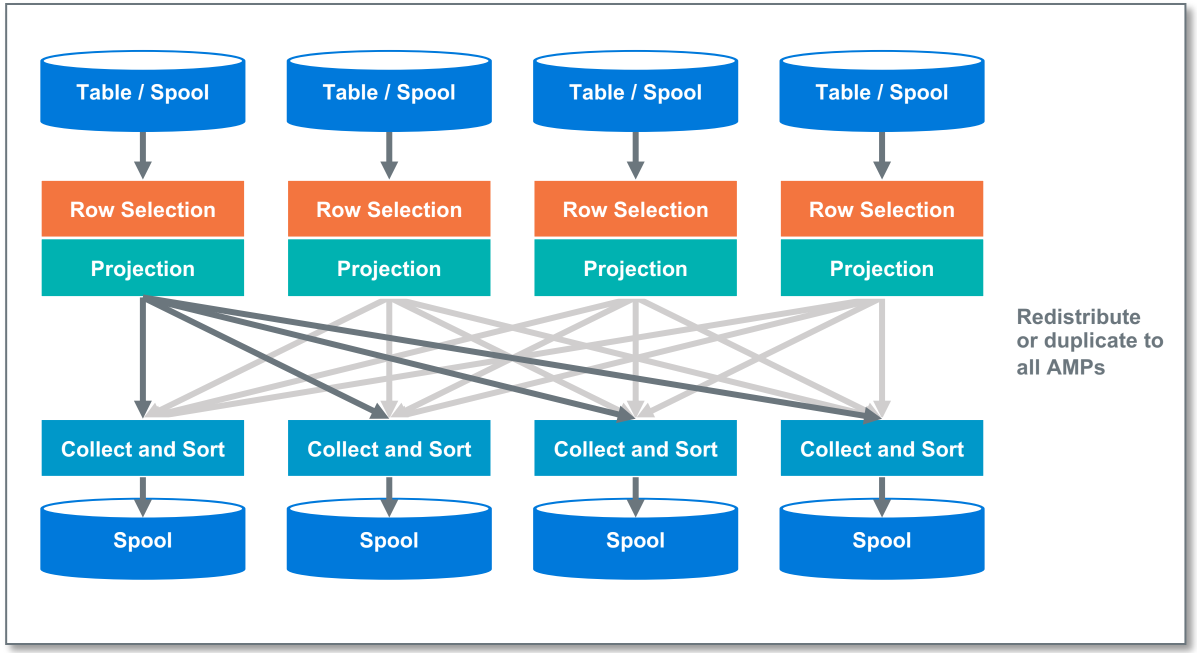
teradata.

Join Preparation

Query Execution

- Join preparation works in parallel across all AMPs
  - Selection to eliminate rows
  - Projection to eliminate columns
  - Eliminate bytes as early in the plan as possible
- Leverages BYNET to get data co-located for join
  - Redistribute : Hash by some set of attributes, often the join columns
  - Duplicate: Make a full copy of the prepared set on every AMP
- Collect results in receivers and write to spool or feed directly to first stage of sort if needed

- Uses Spool for intermediate results
  - Intermediate spools can be released after use (marked “last use” in explain)

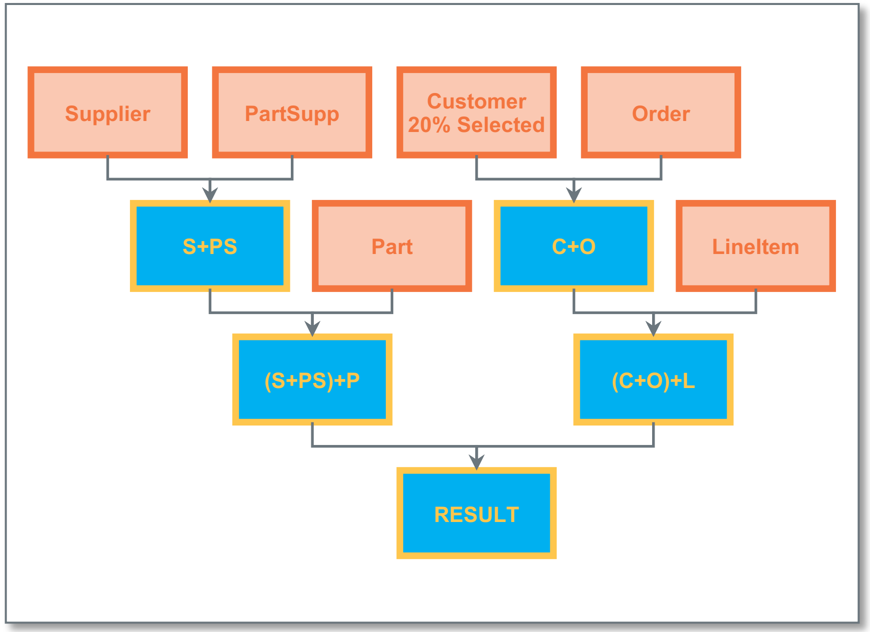


teradata.

Joins

Query Execution

- Joins in Teradata all work in parallel and scale linearly
  - Must have data co-located in an AMP to join
- Join operates on two relations producing a join result which can be spooled locally or piped into redistribution and or sort as required
  - Queries with more than two tables are made into a series of two relation joins
- Many types of join operations available
  - Merge Join
  - Hash Join
  - Product Join
  - Nested Join
  - Outer Join versions of each
  - Exclusion and Inclusion Join for sub-query processing
- Each AMP performs it’s part of the join independently
  - Communicate only when completely done

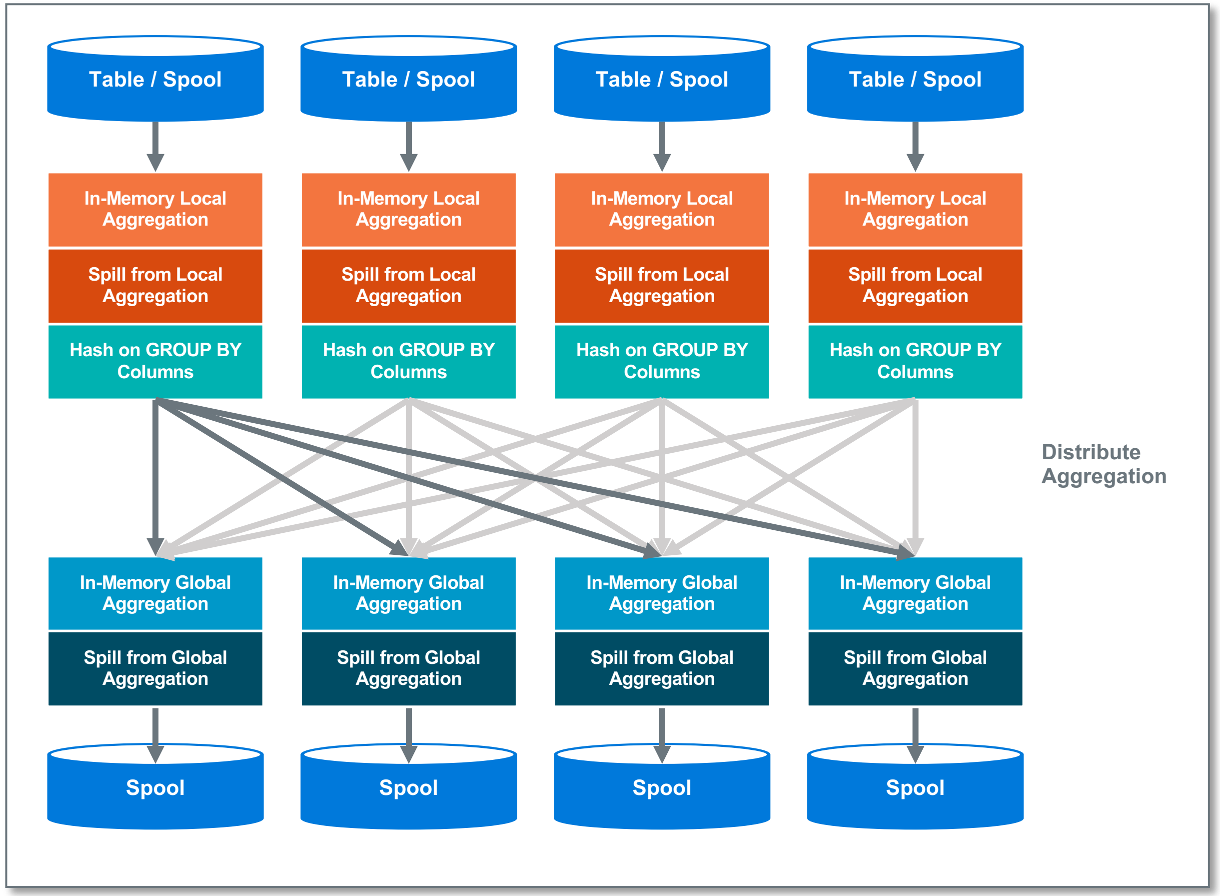


teradata.

Parallel Aggregation Assisted by the Optimizer

Query Execution

- Aggregation makes full parallel use of all AMPs
  - Final result and final aggregation distributed across the platform
  - Linearly scalable across any data size
- Perform local aggregation resulting in (at most) one record per group per AMP
  - Skip if we know that little or no reduction will occur
- Redistribute by hash on GROUP BY columns to get all rows for a group together and distribute groups evenly across all AMPs
  - Skip if we know already distributed by GROUP BY columns
- Receive redistributed rows and perform Global Aggregation
- Caches at both local and global level do aggregation in-memory unless number of groups is very large
  - Cache spill that adapts to data pattern (e.g. most frequent keys don't spill)



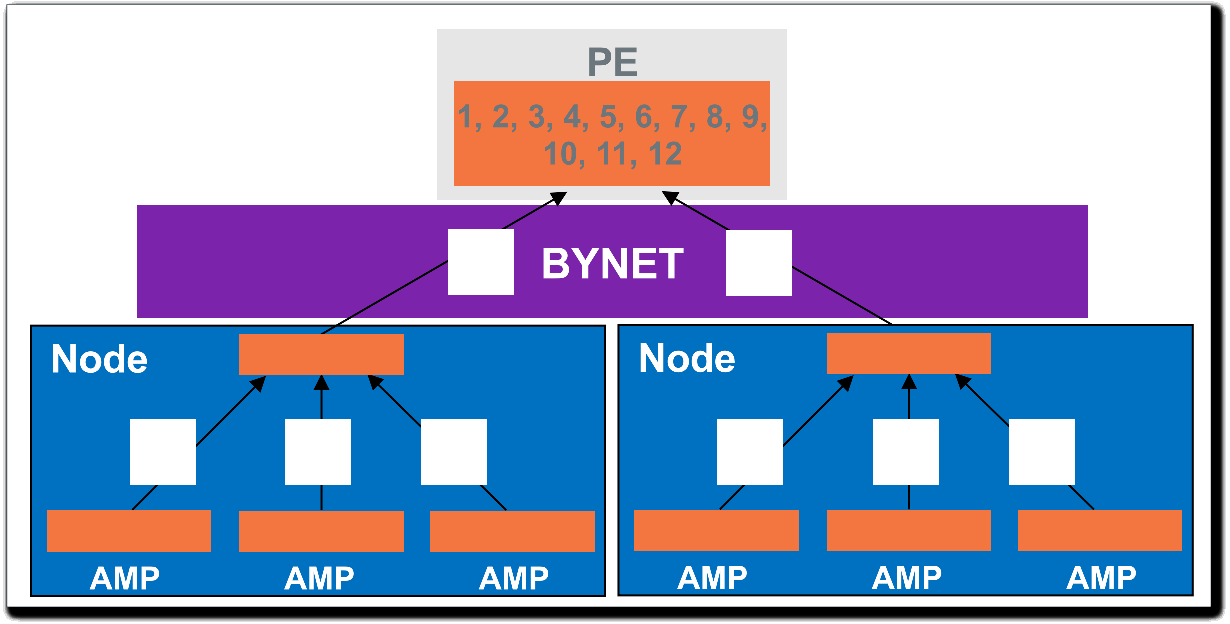
teradata.

Merge of an Answer Set

Query Execution

- Each AMP sorts and spools its part of answer set
- Query complete reported to application
- Application performs fetch, interface (JDBC, ODBC,…) requests buffer of rows
- Each node merges its contributing AMPs answer set into a buffer for the BYNET
- BYNET merges the sorted buffers and fills a buffer to return to the interface

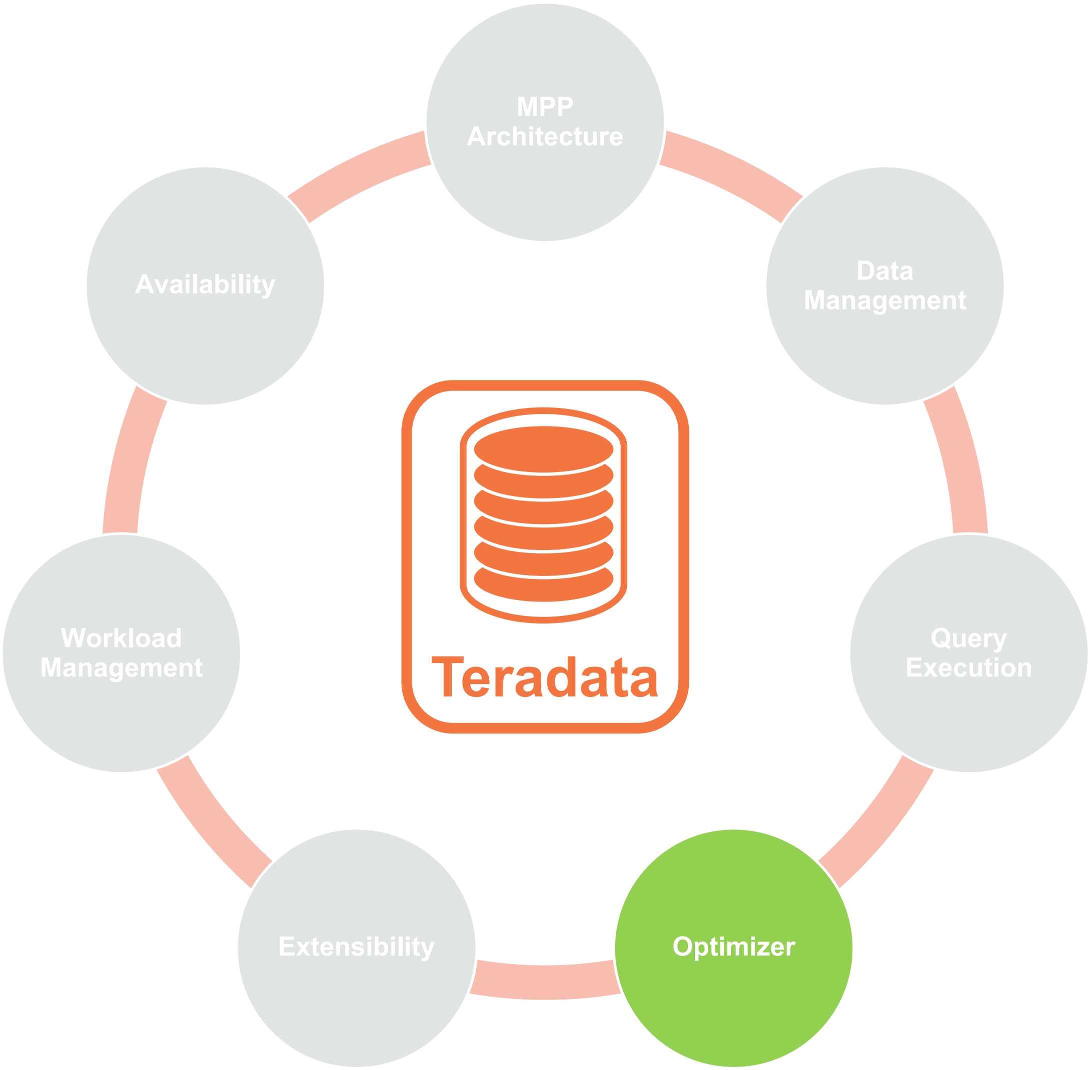
- All levels merge in parallel, and data is transferred using point-to-point messages, maintaining linear scalability as the size of the answer set or number of nodes increases
- The "big sort" penalty has never existed (N(log(N) algorithm, N divided by # of AMPS)
- Final answer set never has to be brought together into a single node for a large final sort – only merge a buffer at a time on demand



teradata.

What Makes Teradata Unique

Key Components/Concepts



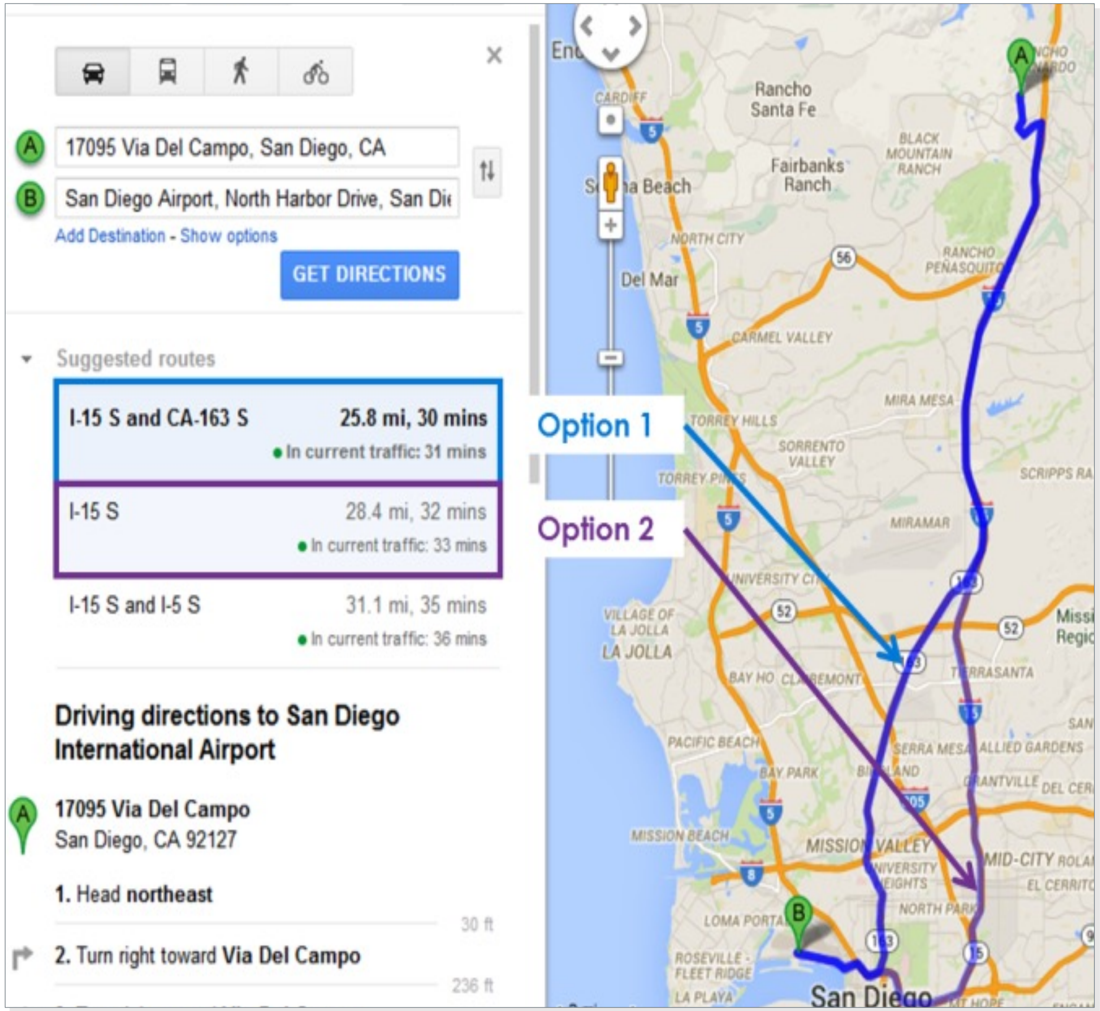
teradata.

Teradata Optimizer Goals

Optimizer

Primary Goals of the Optimizer:

- In-database
- Minimum tuning efforts
  - No hints
  - Only indexes and statistics
- Use the platform effectively and efficiently
- Turn a SQL statement into a series of steps for execution



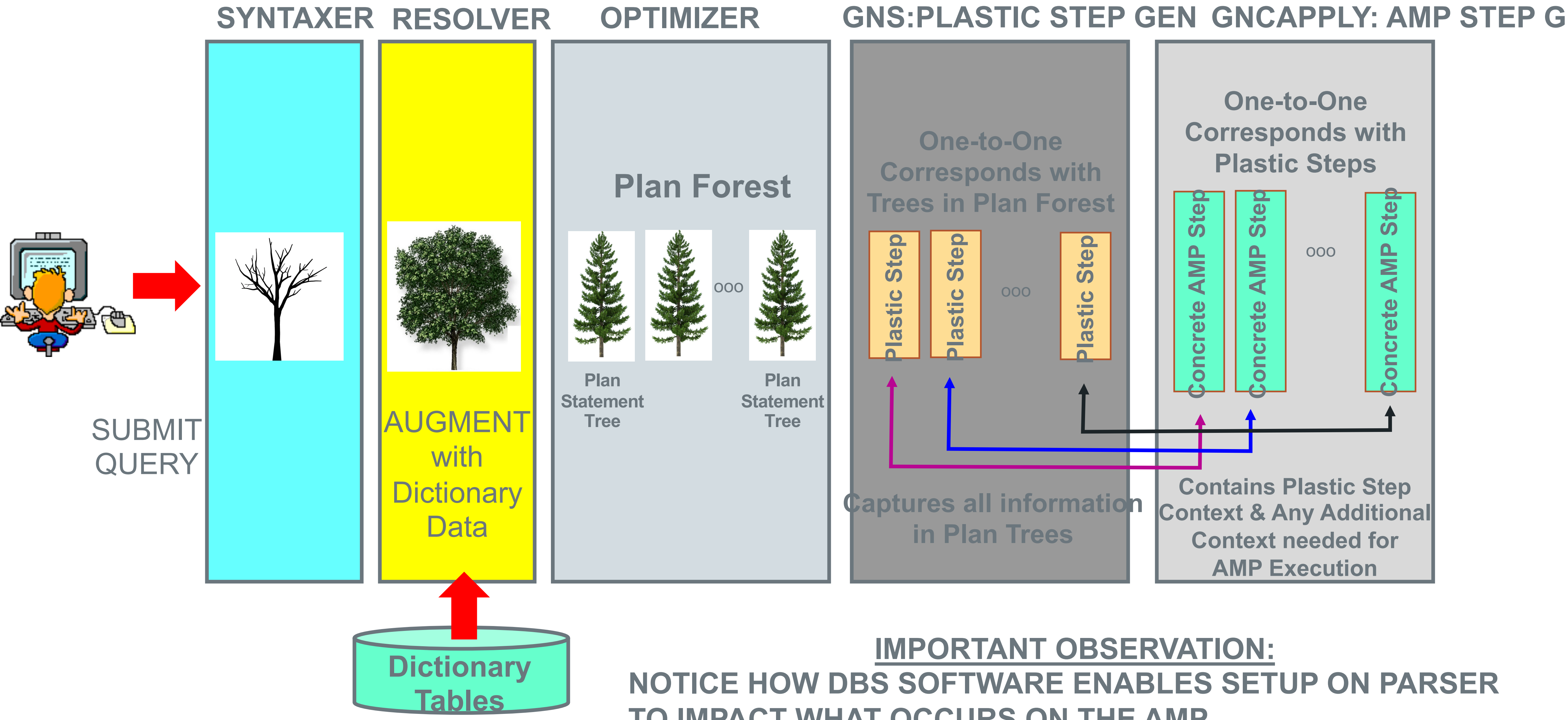
teradata.

Running Queries on Teradata Database

Optimizer

- ANSI compliant SQL
  - Small number of Teradata specific analytic extensions
- High performance algorithms
  - Efficient use of resources not just counting on parallelism
  - Join, Aggregation, Sort, …
  - Compiled expressions
- Complex query features
  - Derived tables, Case expressions, all forms of Sub-queries, Sample, …
  - Big limits: 128 table joins, 128 nesting levels, …
  - 1MB SQL/Views/Macros

CONCRETE STEP (AKA, AMP STEP) GENERATION

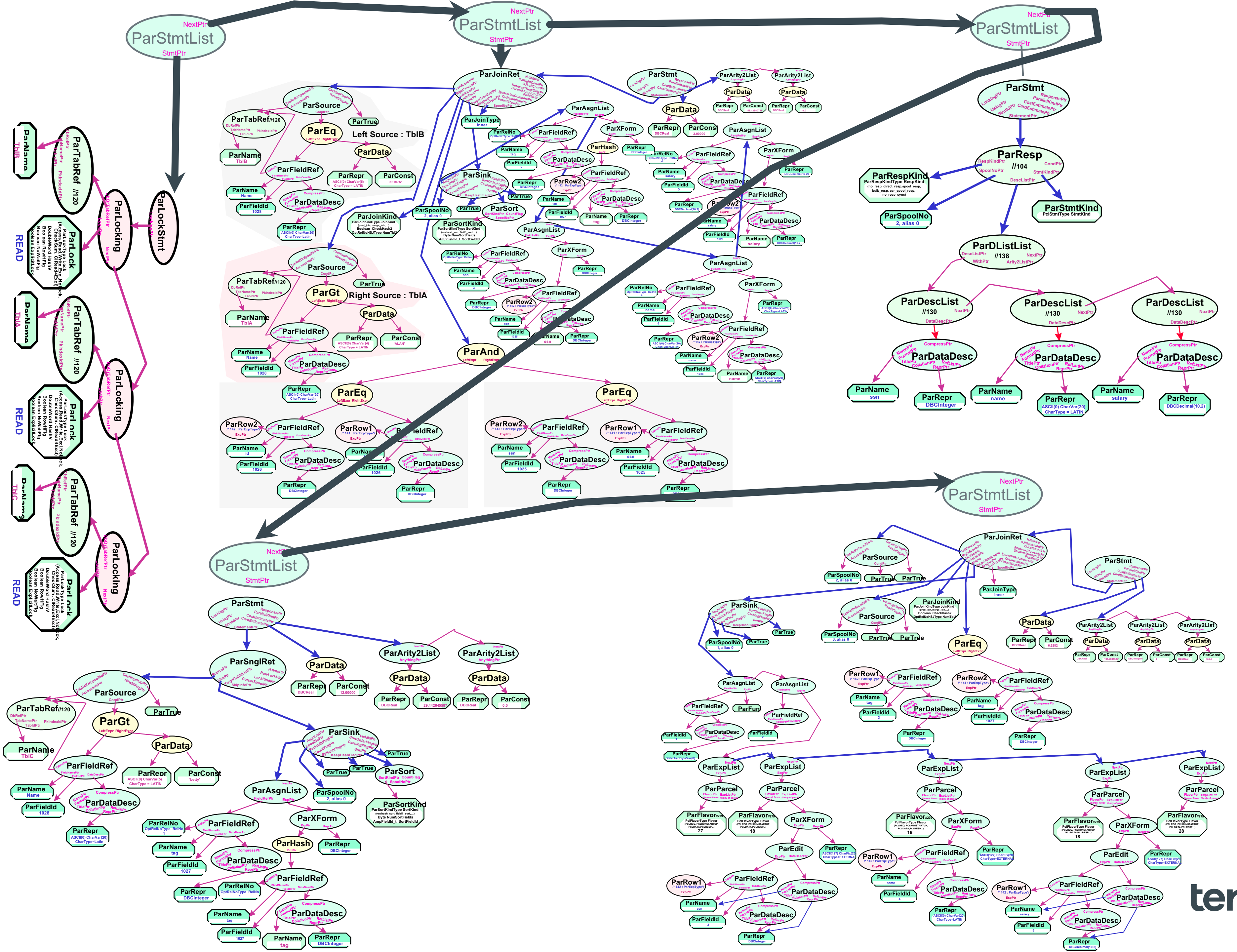


IMPORTANT OBSERVATION:

NOTICE HOW DBS SOFTWARE ENABLES SETUP ON PARSER TO IMPACT WHAT OCCURS ON THE AMP

teradata.

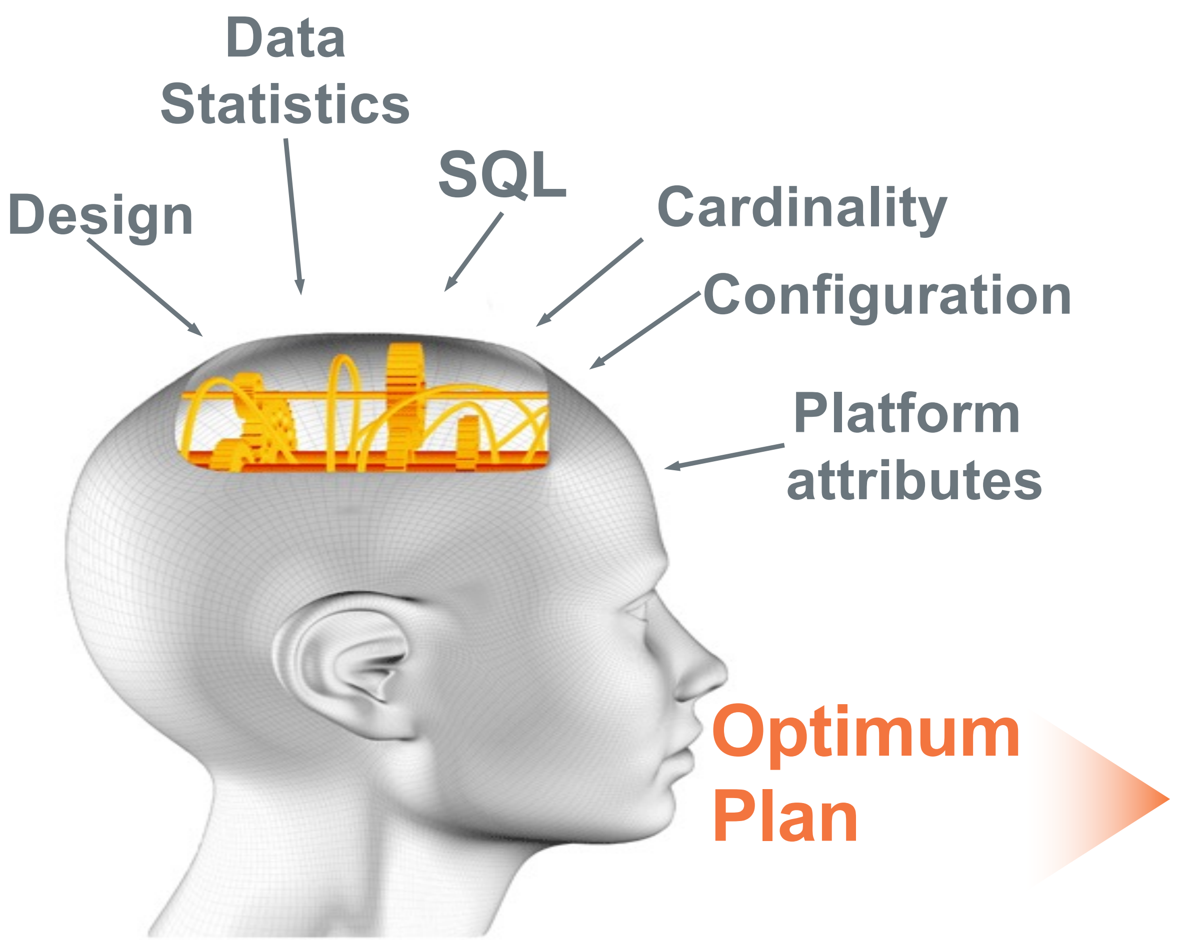
select TblA.ssn, TblA.name, TblA.salary from TblA, TblB, TblC where TblA.ssn=TblB.ssn and TblA.id=TblB.id
and TblA.tag=TblC.tag and TblA.name > 'alan' and TblB.name < 'zebra' and TblC.name > 'betty';



Teradata Optimizer

Optimizer

- Cost based query planning
  - Balance costs of
    - Cardinalities
    - Sorts and joins
    - Redistribution
    - Disk I/Os
    - Networks and nodes
    - Compression
  - Lowest resource costs
  - Fastest elapsed time
- Automatic rewrites
- Automatic optimization
  - No hints
  - No degrees of parallelism

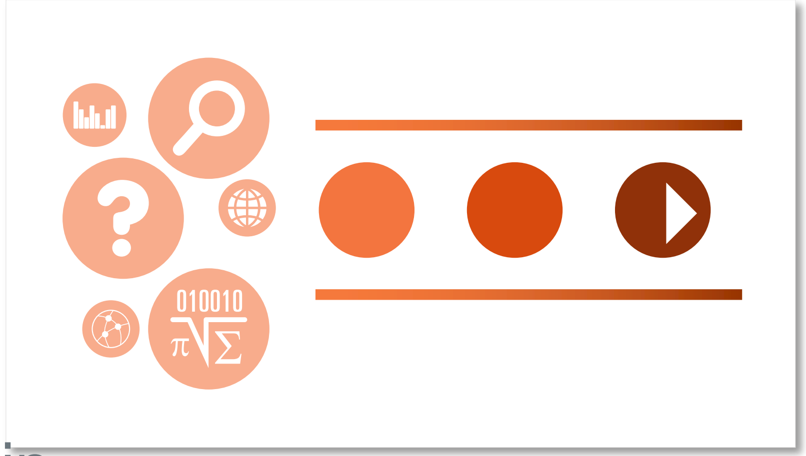


teradata.

Planning

Optimizer

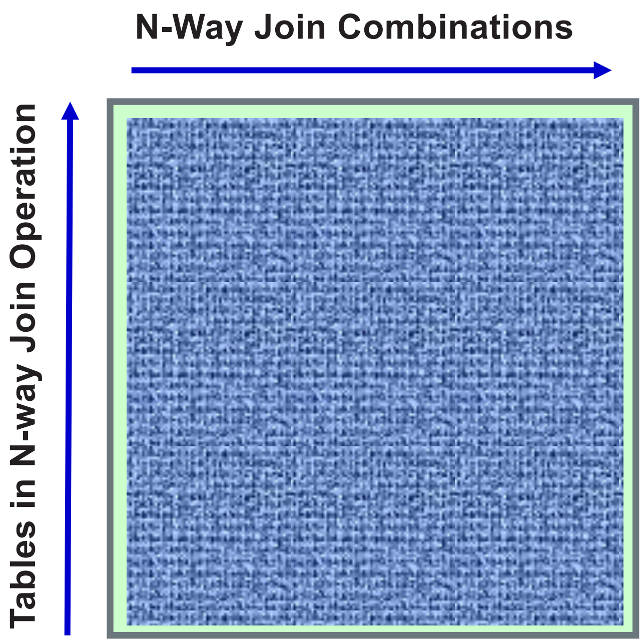
- Process of turning a query into a plan
- Optimizer deconstructs SQL into individual components
  - Relations, Join Conditions, Access Conditions
- Access Method: how to access each relation
  - Table Scan, Index Use, Bitmap Use
- Join Method: how pairs of table are joined
  – Merge Join, Product Join, Hash Join
  – Outer Joins, Inclusion and Exclusion Joins
- Join Order: sequence of table joins
- Join Geography: how rows are relocated prior to the join
  – Redistribute rows, duplicate rows, local spool
- Sort Order: order for merge join, ordered result, distinct



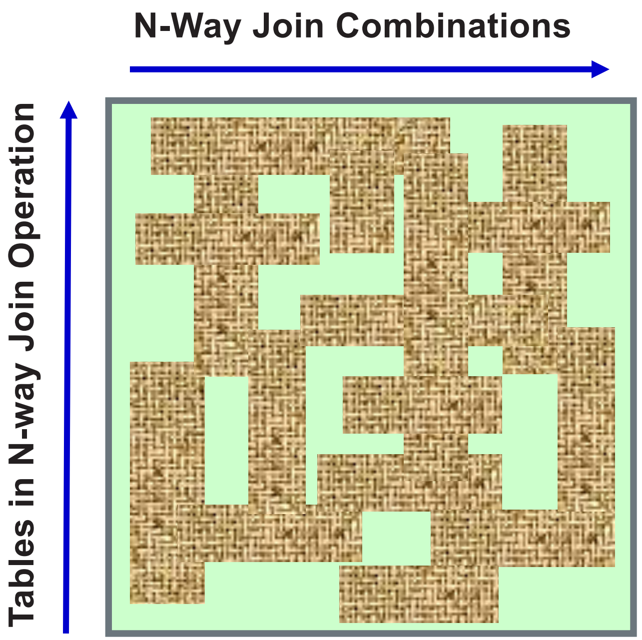
teradata.

Join Planning

Classic System “R”
Exhaustive



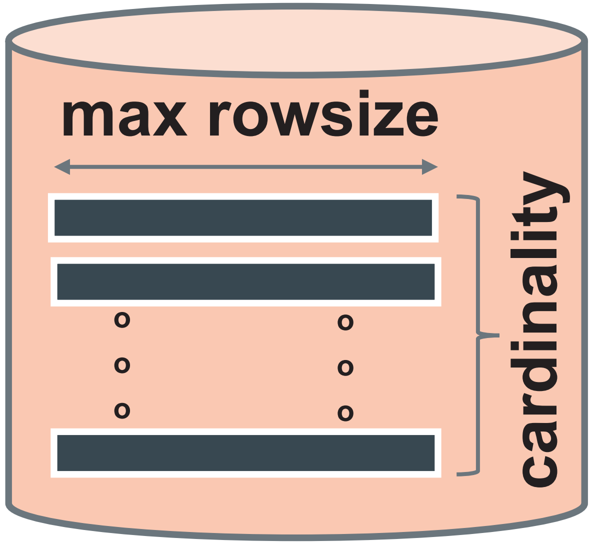
Teradata
Intelligent Plan Navigation



Optimizer Plan Selection: Comes Down to "Cost"

Cost Ξ Time to complete an operation in ms

Costing Building Blocks



+

CPU
cycles-per-row
operation

Allows us to Calculate

OPT PLAN COST

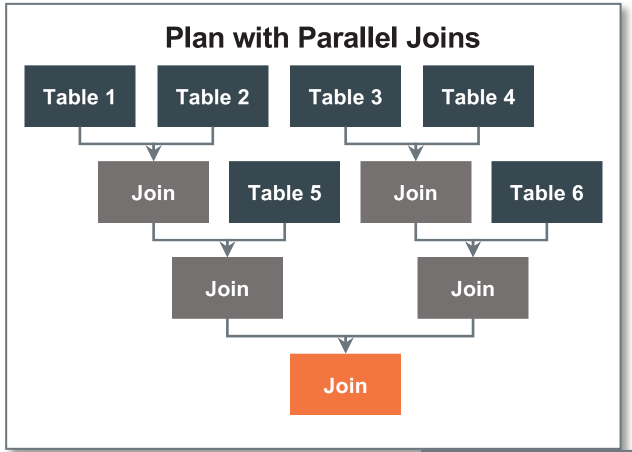
- ΔT : Disk I/O cost
- ΔT : Network (Redistrib) Cost
- ΔT : CPU Processing Cost

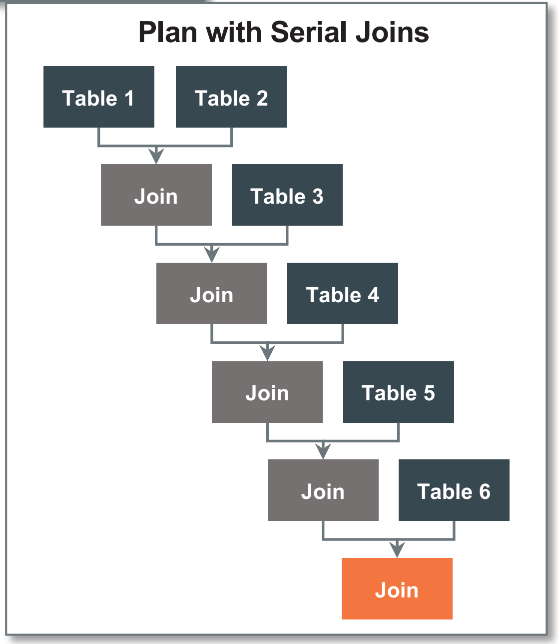
"Cardinality is the "hardest one"
This is where "Statistics"
comes into play

Join Order

Optimizer

- The order of joins is determined by identifying the minimum cost ordering
- Look at size (with conditions applied), distribution, available access methods, available join methods, …
- Create each order and determine cost, pick best
- Thousands or millions of combinations
  - Game theory used to eliminate unlikely combinations
  - Look-ahead limits to control reviewed combinations
  - Heuristics for special cases, starting points
- Dependent on good Statistics!
- And the Optimizer makes the decision – does the hard work of figuring it out

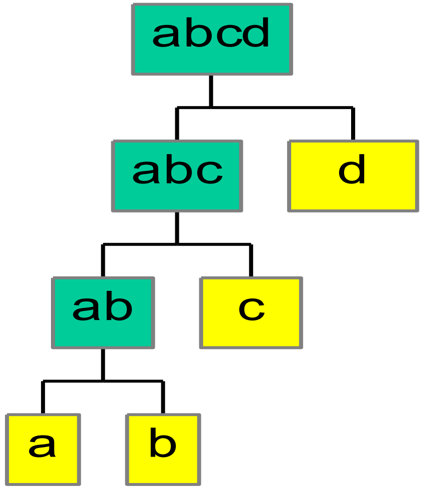


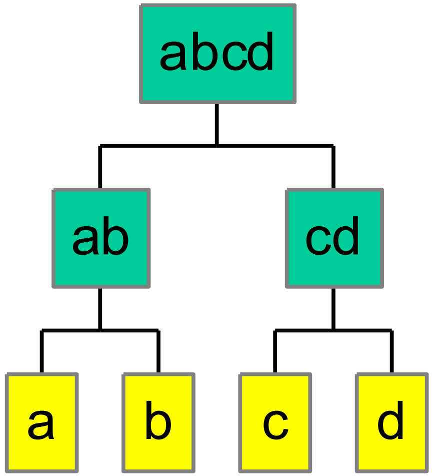


teradata.

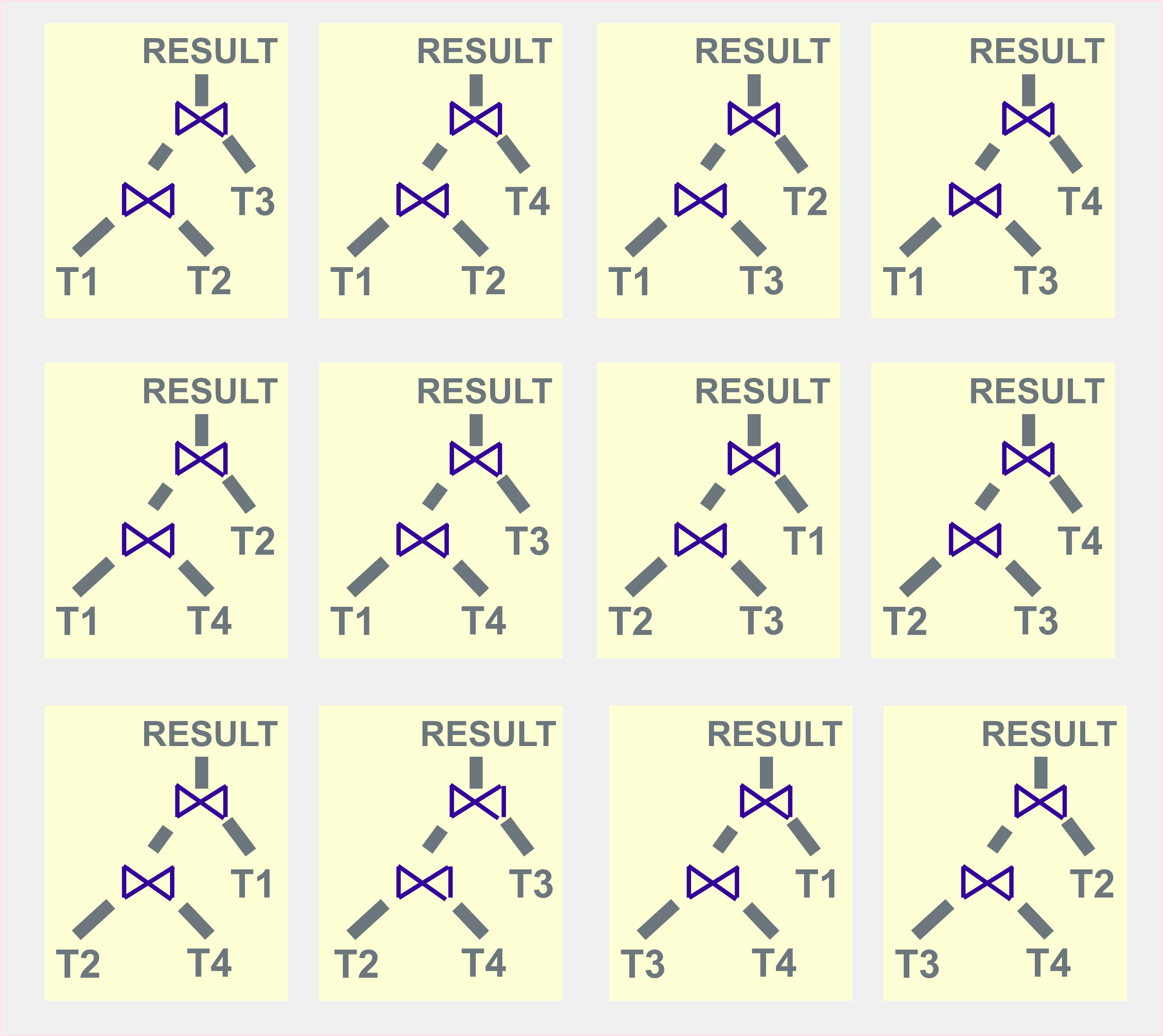
N-Way Join - Combinations

- Sequence of binary joins
- Not all possible combinations considered
  - 10 way join - 17.6 billion possibilities
  - 64 way join - $1.2 * 10^{124}$ possibilities
- Driven by join conditions
  - Look for “connected” relations
  - “Greedy” algorithm





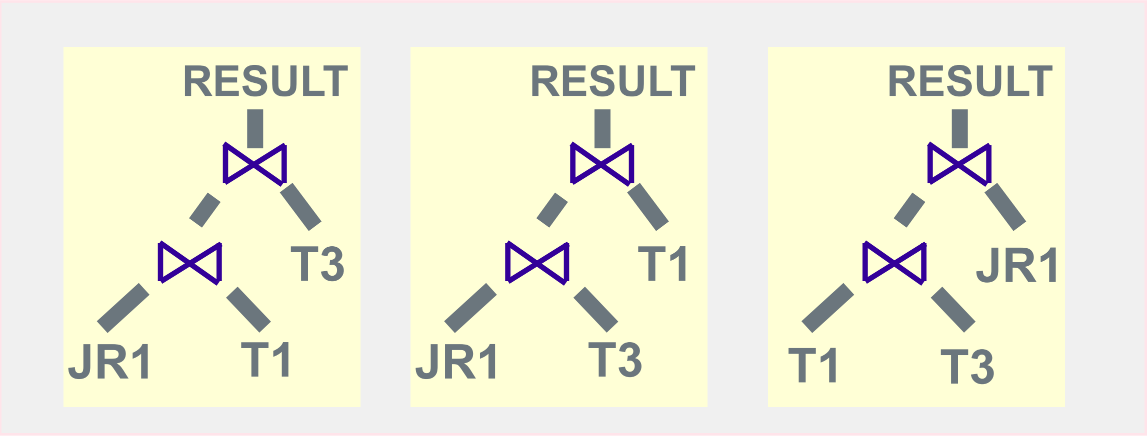
4-Way Join : LookaheadOne Join Planning

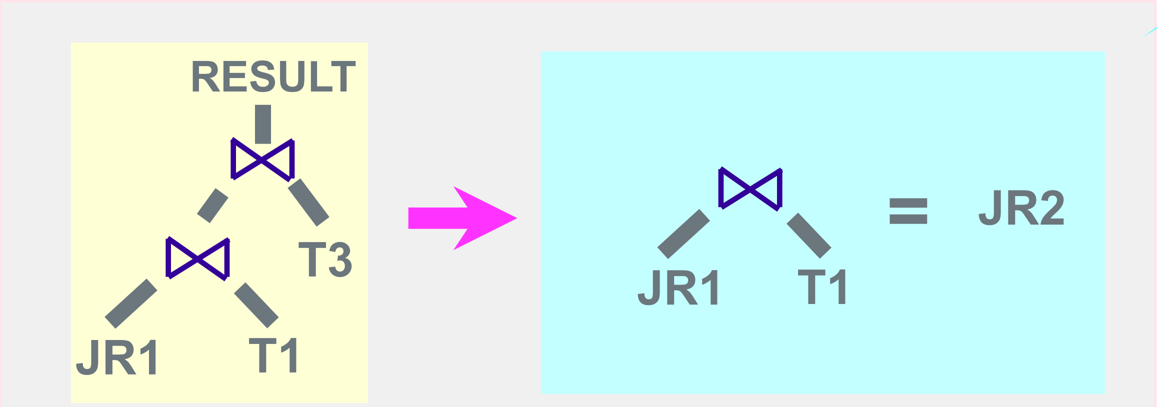


teradata.

4-Way Join : LookaheadOne Join Planning

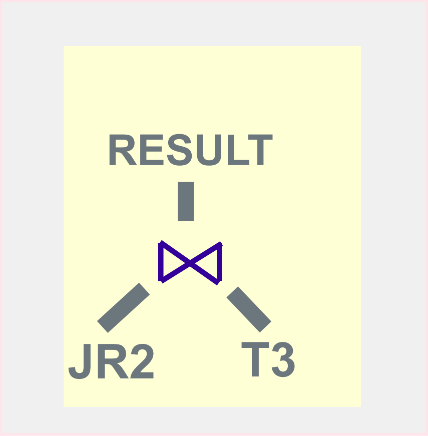
Need to Estimate Cardinality of Committed Pair

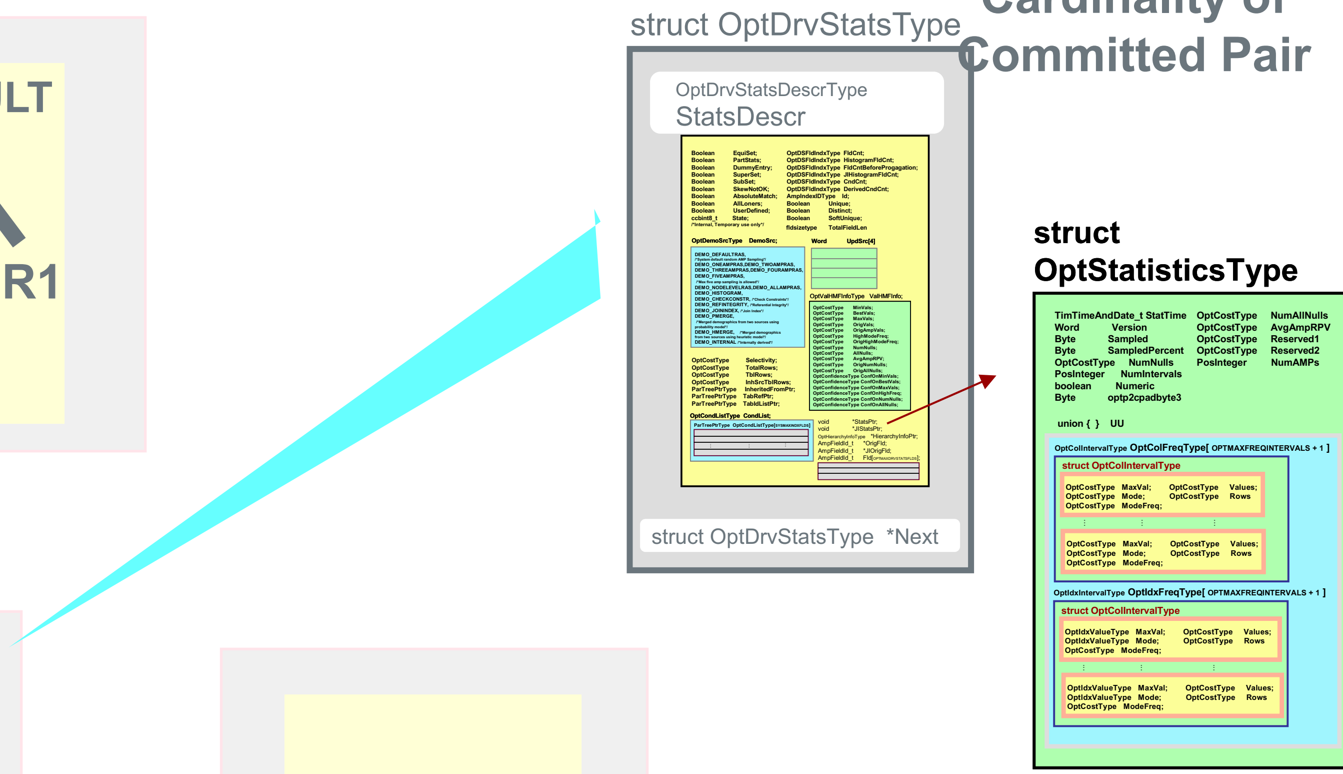




Cheapest Triplet

Committed Principal Join Pair





teradata.

Binary Join Generation

- Try all possible join types based on join conditions
  - Equality: nested join, merge join, hash join, product join
  - Other: product join
- For each join type, try all possible combinations of geography of tables
- Examples for Merge Join

direct left	direct right
hash left	hash right
hash left	local right
local left	hash right
local left	duplicate right
duplicate left	local right

teradata.

Views and Physical Database Design

Optimizer

High performance views:

- No penalty for using views
- Enables wide use for ease of use, security, privacy
- Recommend using views rather than users accessing the base tables
- Compiled early so optimizer only sees base tables and complete condition sets

Physical database design affects planning:

- Indexes, Join Indexes, partitions – provide access methods
- PI, Referential Integrity, data types – guide join planning
- Columnar, Compression, row definition – guide size and I/O estimates

Examples of Optimizer's Advanced Capabilities

- Aggregate Join Index (AJI)
- SATisfiability-Transitive Closure (SAT/TC)
- Query Re-Writes
  - Reorganize a query's structure for better optimization/performance
- Partial Redistribution Partial Duplication (PRPD)
  - Skew optimization
- Incremental Planning and Execution (IPE)
  - Plan in stages based on partial results
- In-Memory and Chipset Optimizations
- Temporal sequencing/normalization
- Geospatial indexing/bounding

Statistics

Optimizer

Three types of histograms are generated:

- Equal-height interval histograms:
  - Each interval has the same number of values
- High-biased interval histograms:
  - Each interval has no more than two values
- Compression histograms:
  - Up to 200 equal-height intervals, plus one or more high-biased intervals

```
COLLECT STATISTICS ON Employee
COLUMN Department_Number
```

26 Rows
8 Unique values,
0 Nulls

Min value = 100
Mode Value = 401
Mode Frequency = 7

4 Intervals (in High-biased interval histogram)

Max	Mode	Mode Frequency	Non-Mode Value	Non-Mode Rows
(1) **201**	**100**	**1**	**-2**	**2**
(2) **302**	**301**	**3**	**-2**	**1**
(3) **402**	**401**	**7**	**-2**	**2**
(4) **501**	**403**	**6**	**-2**	**4**

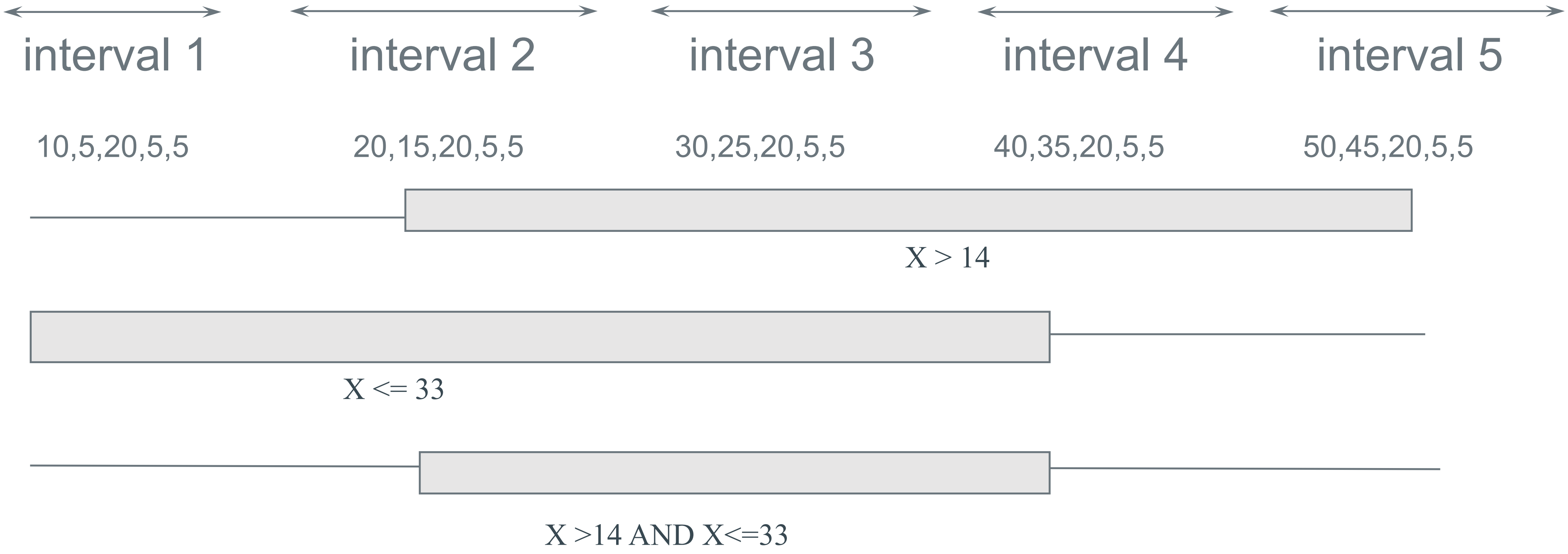
Statistics Collection

- High bias values separated to recognize skew
  - Optimizer will generate different plans for different values or ranges
- Statistics calculated across all rows on all AMPs with a fully scalable algorithm
- Statistics sampled when not available, dynamic choice of single AMP or all AMPs
- Statistics extrapolated when stale relative to data change and table size relative to when statistics were collected

Using Statistics to Estimate Number of Rows (cont.)

- Interval Information

  Max, Mode, Frequency, Number of values <>Mode, Number of rows <> Mode

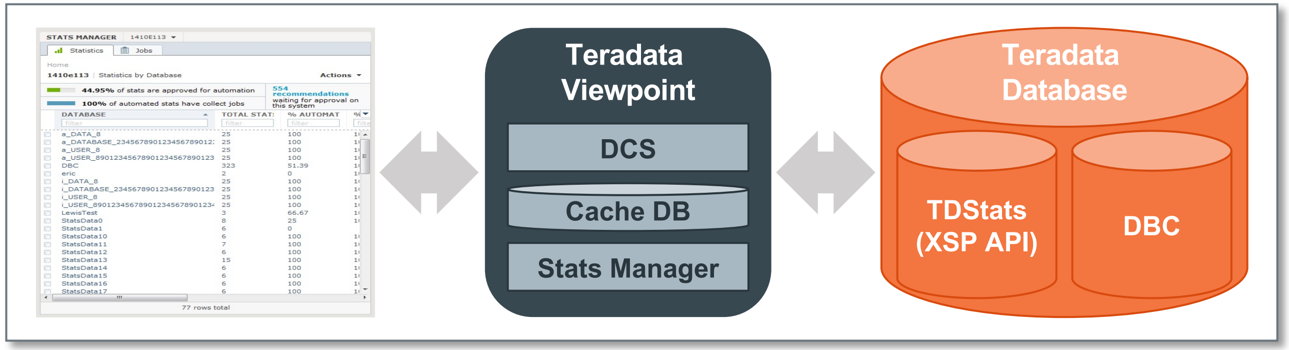


Auto-Stats Collection

Optimizer

Automates and provides intelligence to DBA tasks related to Optimizer Statistics Collections where such tasks include:

- Identifying and collecting missing statistics that are needed for query optimization
- Detecting stale statistics and promptly refreshing them
- Identifying and removing unused or unimportant statistics from routine maintenance
- Prioritizing the list of pending collections such that important and stale statistics are given precedence
- Executing needed collections in the background during scheduled time periods
- Dynamically issuing collections in response to key events, most notably the completion of bulk load operations
- Configuring the system to ensure that resource usage incurred by statistics collections is properly regulated and throttled through TASM or other mechanisms



teradata.

Explain

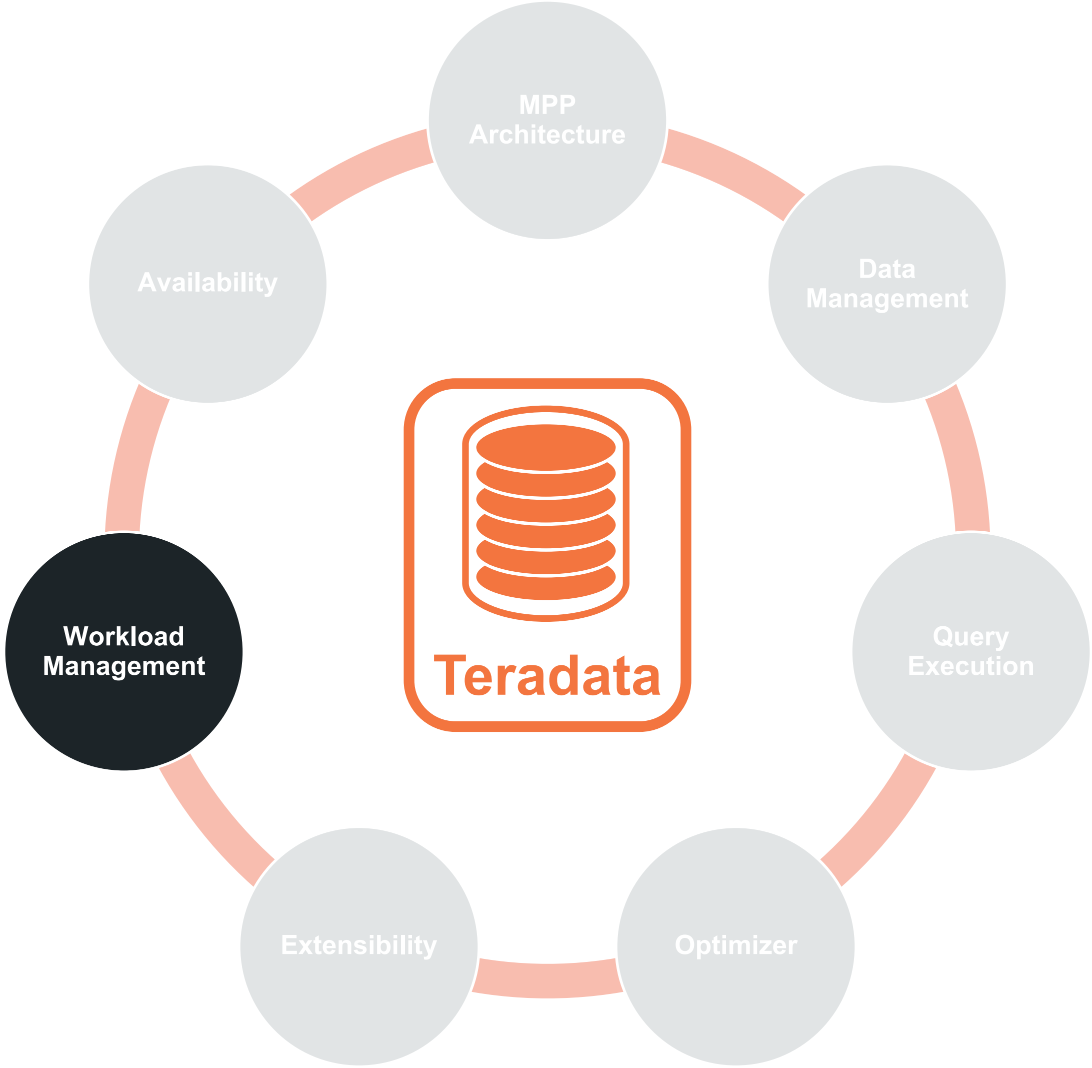
- Two table join with NUSI access in one table and FK to FK join. and aggregation and sort

BLOG

- https://www.teradata.com/Blogs/How-to-Repurpose-Successful-Database-Techniques-inside-Teradata-Vantage

What Makes Teradata Unique

Key Components/Concepts



teradata.

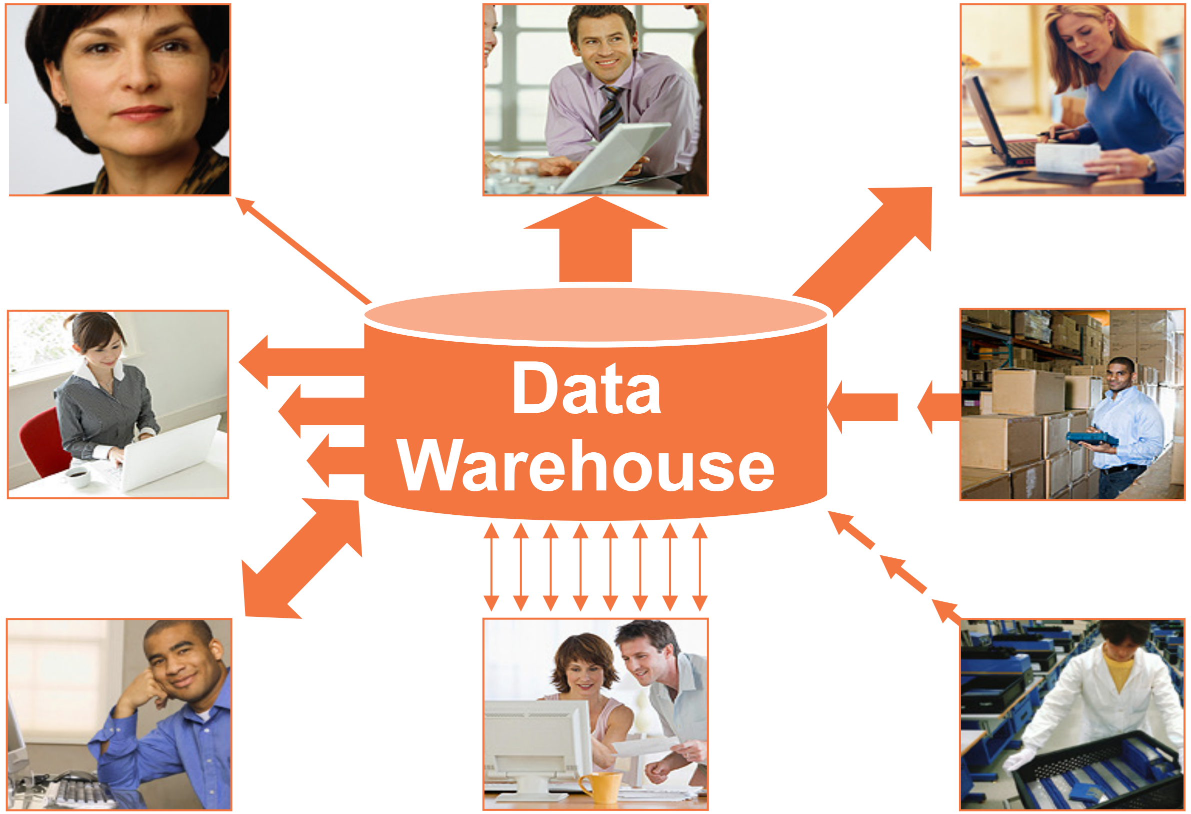
Many Demanding Users

Customer analytics
Needs 30% for long periods

CFO Dashboard
Needs 5% of the system in 10-second bursts

Power user reporting
Needs 30% for long periods

Query and Reporting
Needs 20% to 30% of the system daily



Inventory delivery data load
Needs 10% of the system in bursts

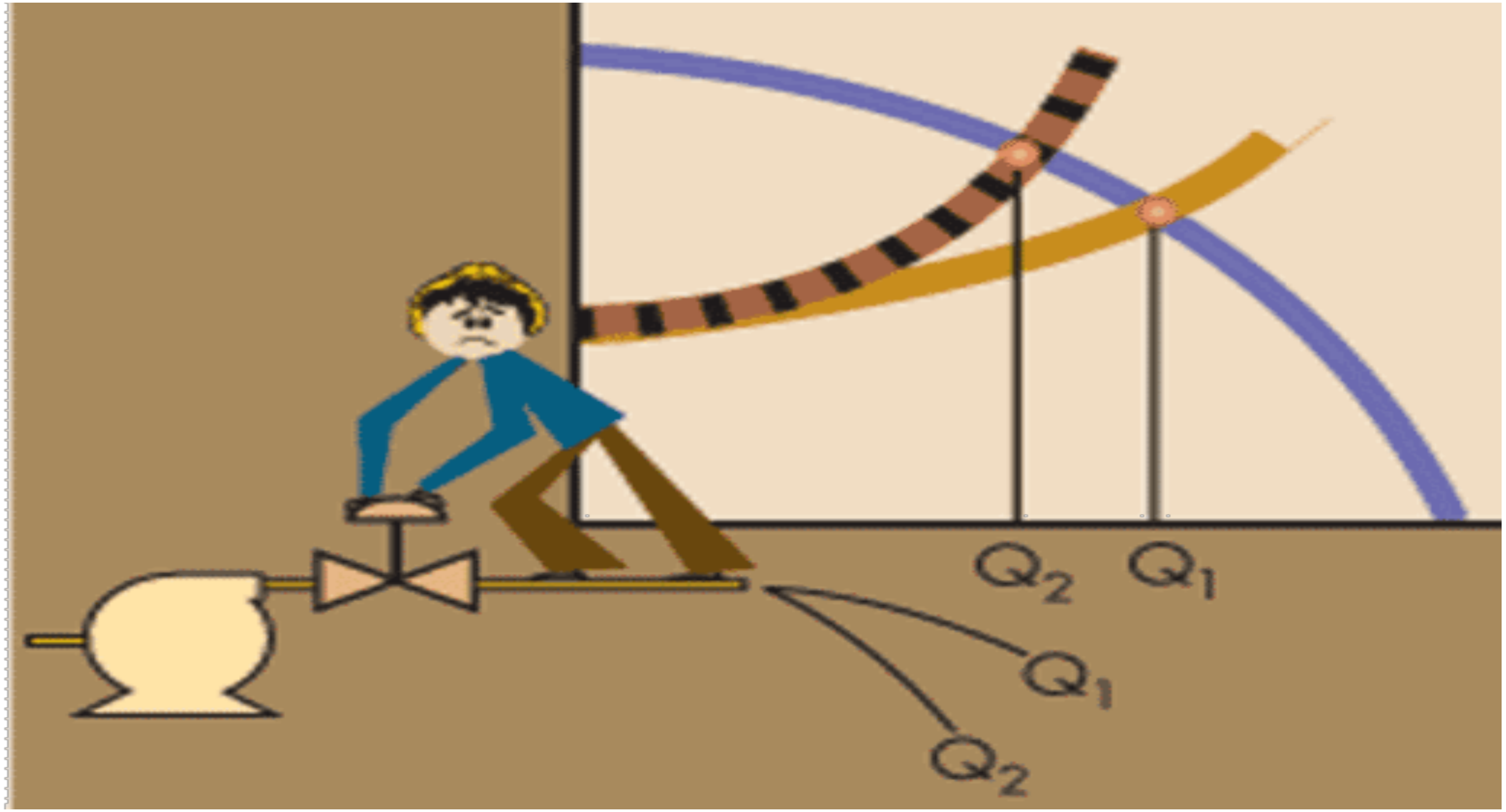
Operations Batch jobs
Needs 30% of the system from 8AM to 10:30 AM and nightly

Production line data/load
Needs 10% of the system in bursts

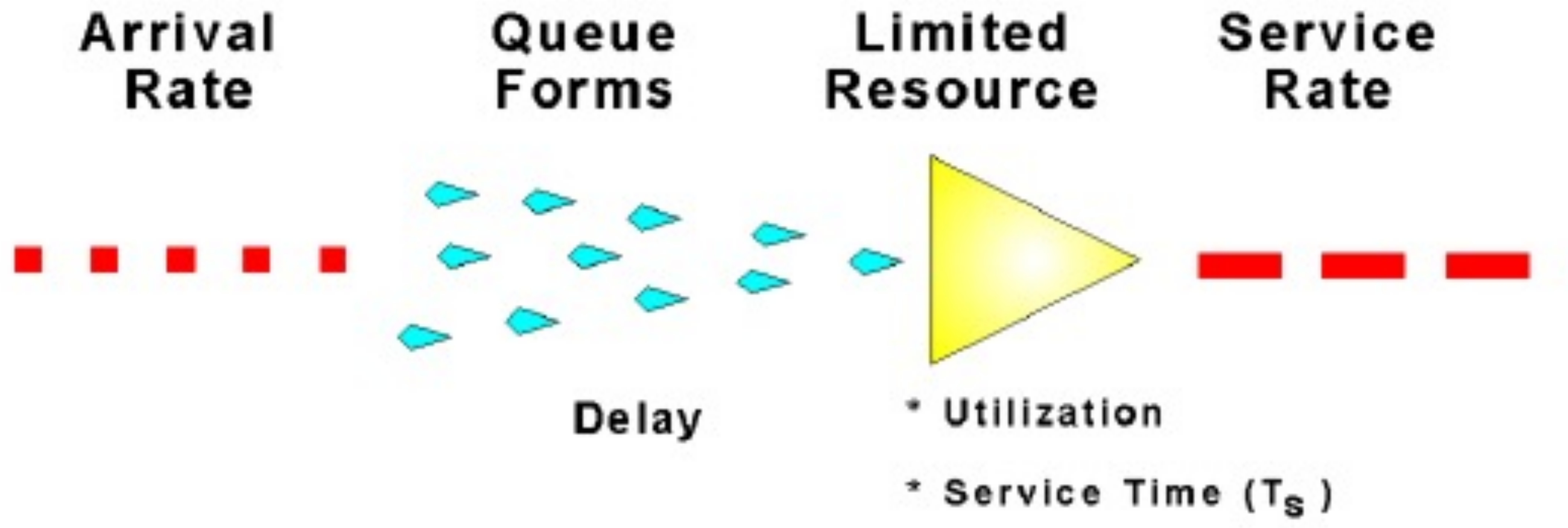
Customer access
Thousands of one-second inquiry and purchase requests

teradata.

Basic Workload Management Controls



Workload Management Queuing

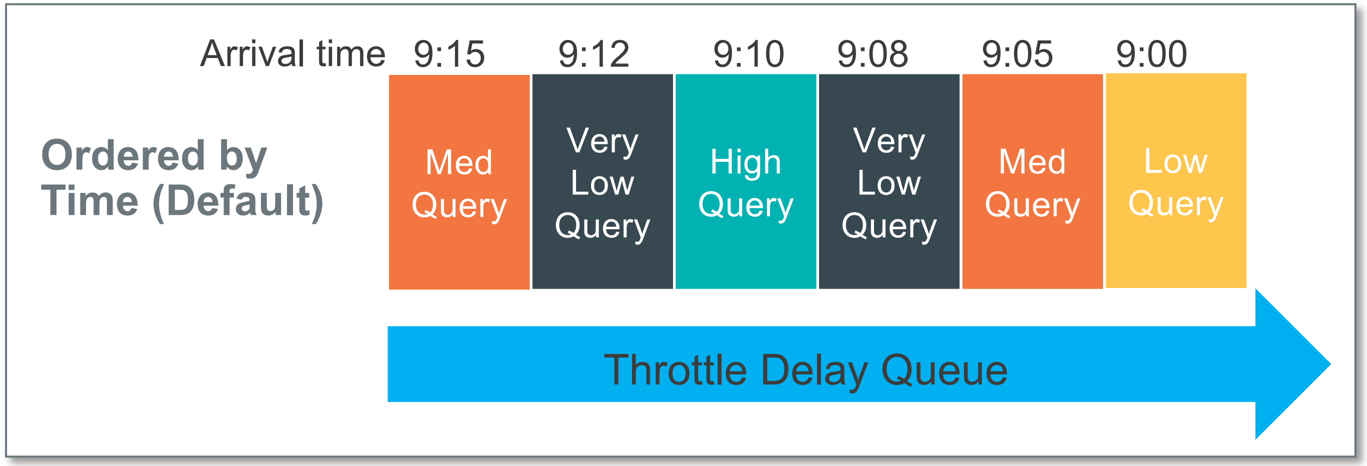


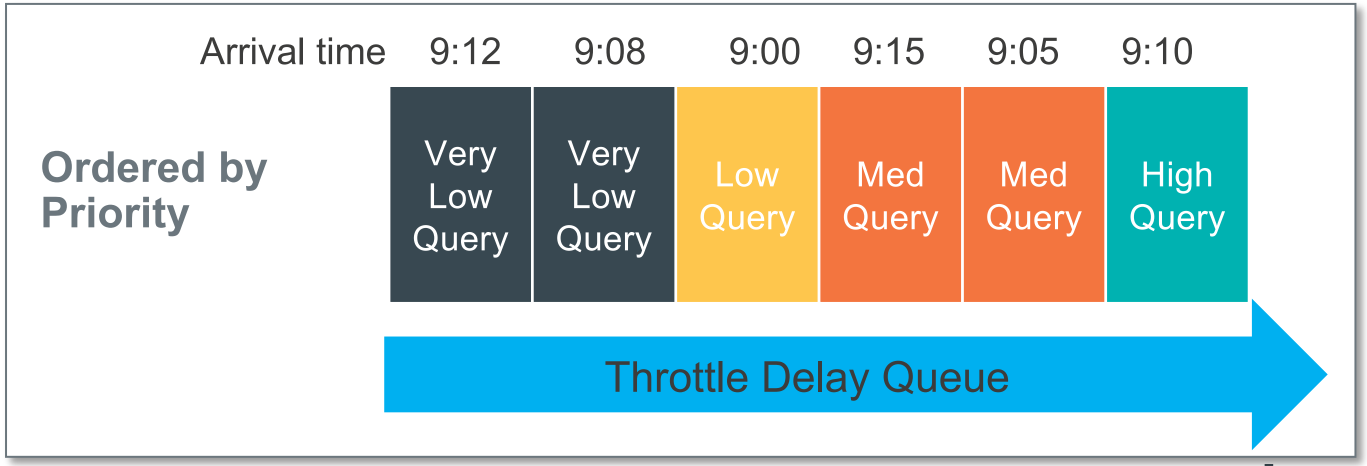
teradata.

Prioritized Throttle Delay Queue Option

Workload Management

- Default delay queue ordering is first-in-first-out (FIFO)
- New option to reorder the delay queue by priority
- Can be used to make sure the highest priority requests are released first

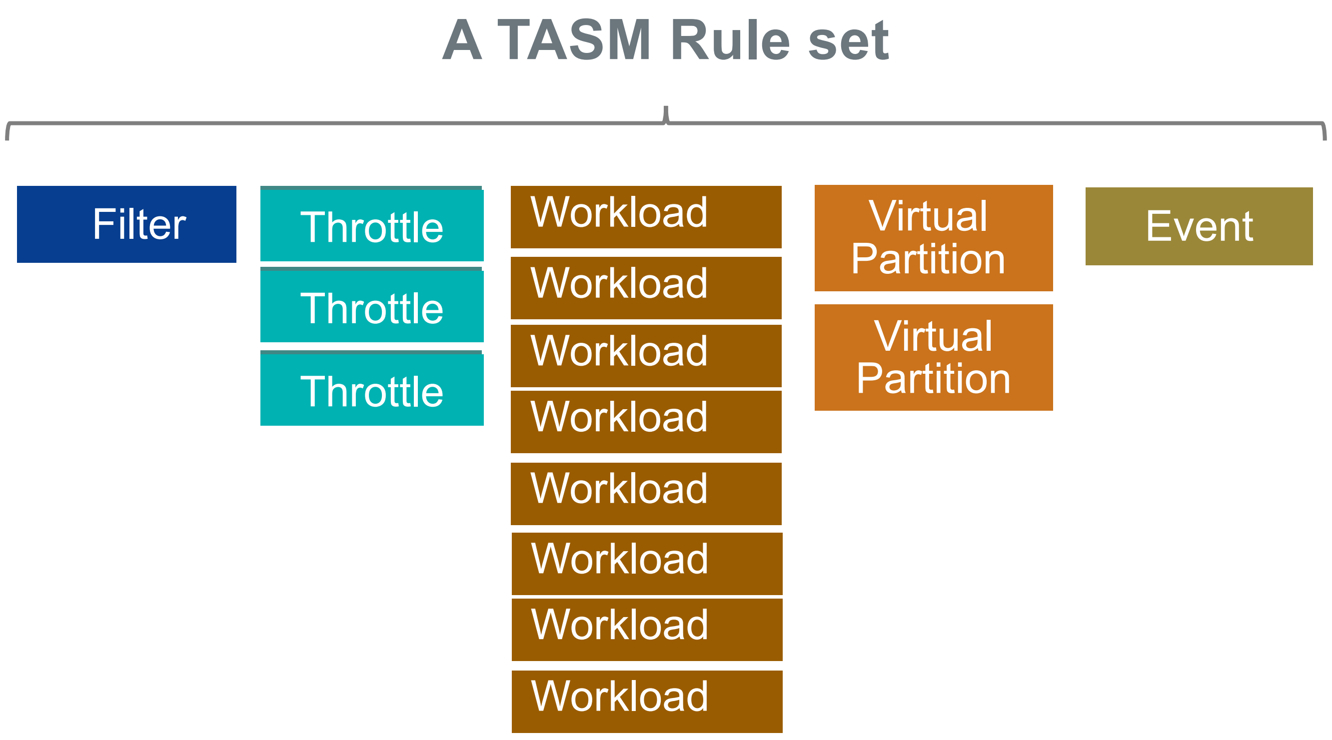




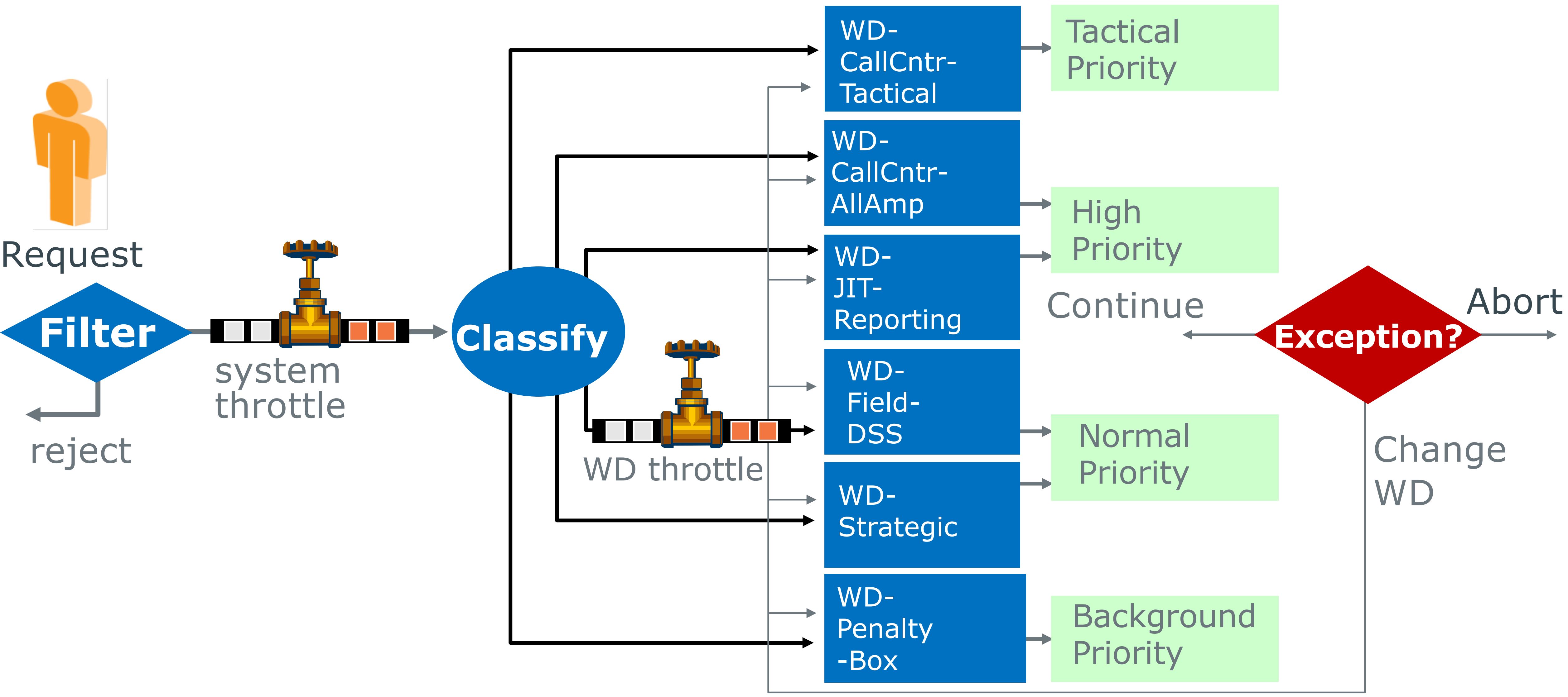
teradata.

How Workload Management Goals are Defined

- A rule set is a bundle of settings and definitions
- Created within Viewpoint Workload Designer
- Only one rule set may be active at a time



TASM: Up to 6 Methods of Management



A Wide Variety of Workload Classification Criteria

Workload Management

Requests are mapped to workloads, system throttles, or system filters if the query's characteristics match the object's classification criteria

WHO Criteria

- User
- Account
- ClientID
- ClientAddr
- Profile
- Application
- QueryBand

"Source" criteria

WHERE Criteria

- Tables
- Databases
- Views
- Stored procedure
- Functions/methods
- QueryGrid server

"Target" criteria

WHAT Criteria

- Estimated time
- Type of join
- Estimated rows
- AMP limits
- Load utility type
- Statement type
- Final row count
- Percent of table accessed

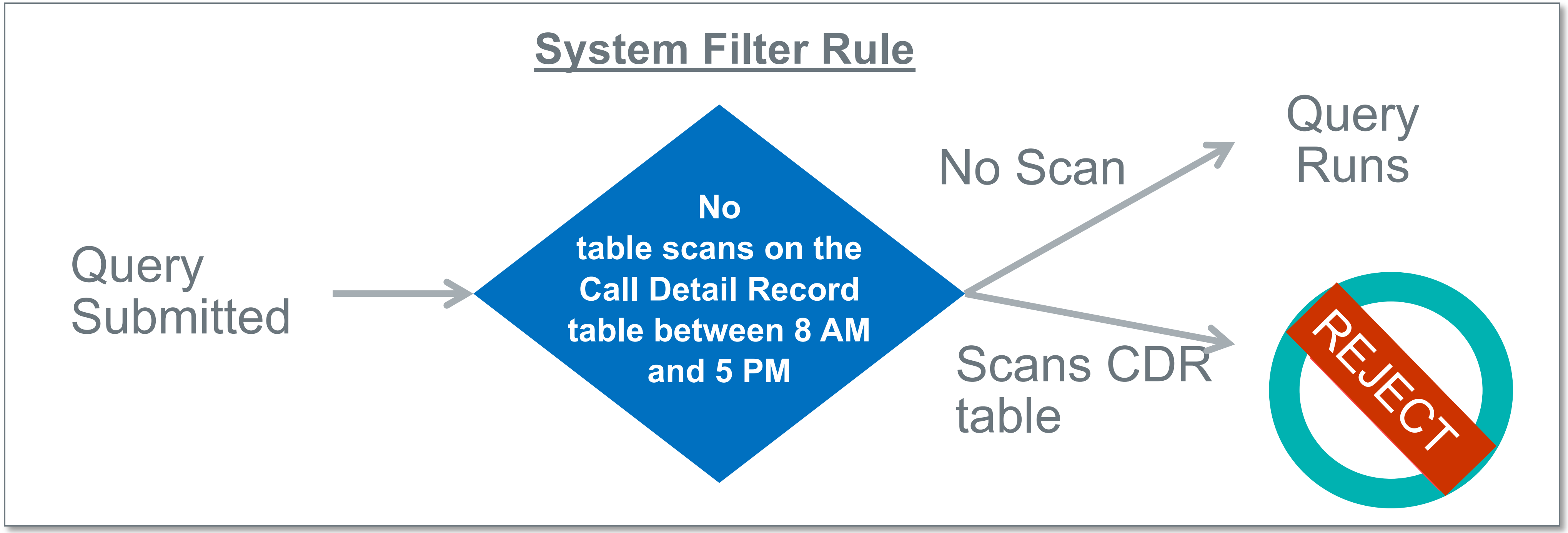
"Query characteristics"

teradata.

System Filter Rules

Reject Queries with Specific characteristics, Before They Begin Running

Workload Management



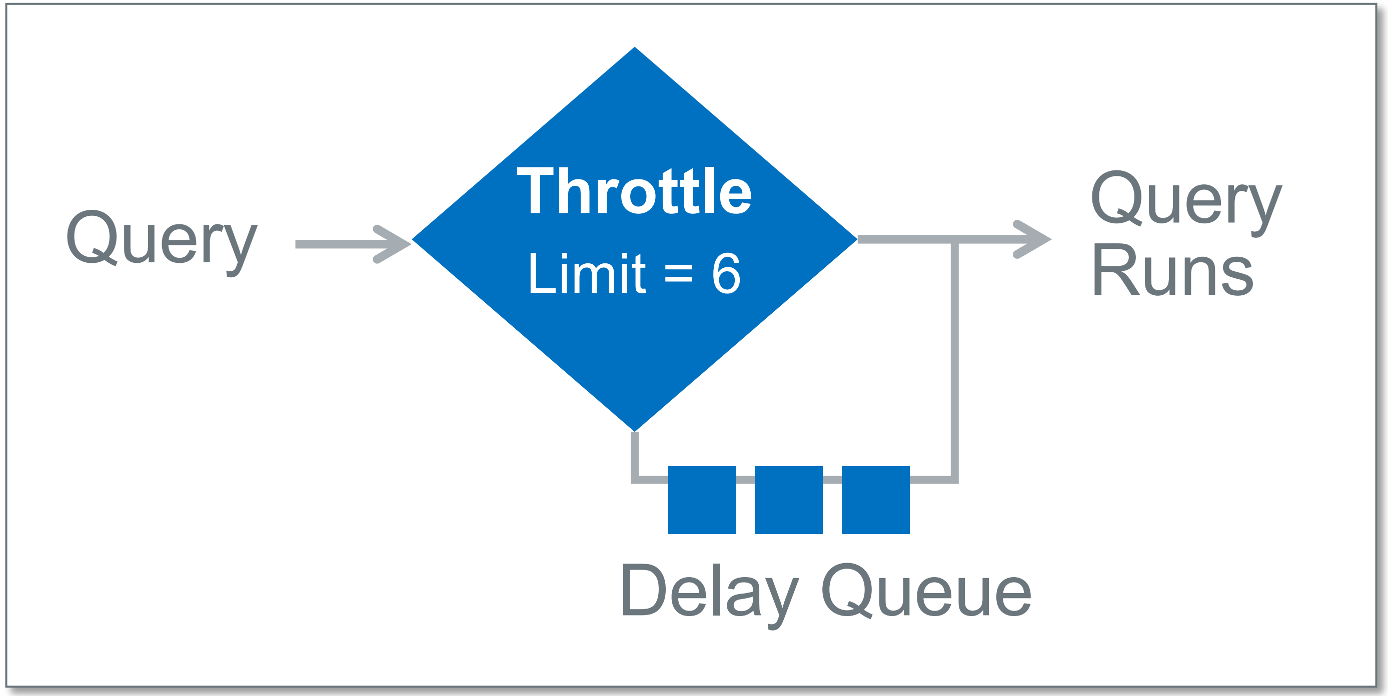
- Prevents inappropriate queries from executing
- Can contribute to user education
- “Warning mode” will report, but not reject the query

teradata.

Throttle Rules

Soften Processing Peaks by Delaying Some Queries at Busy Times

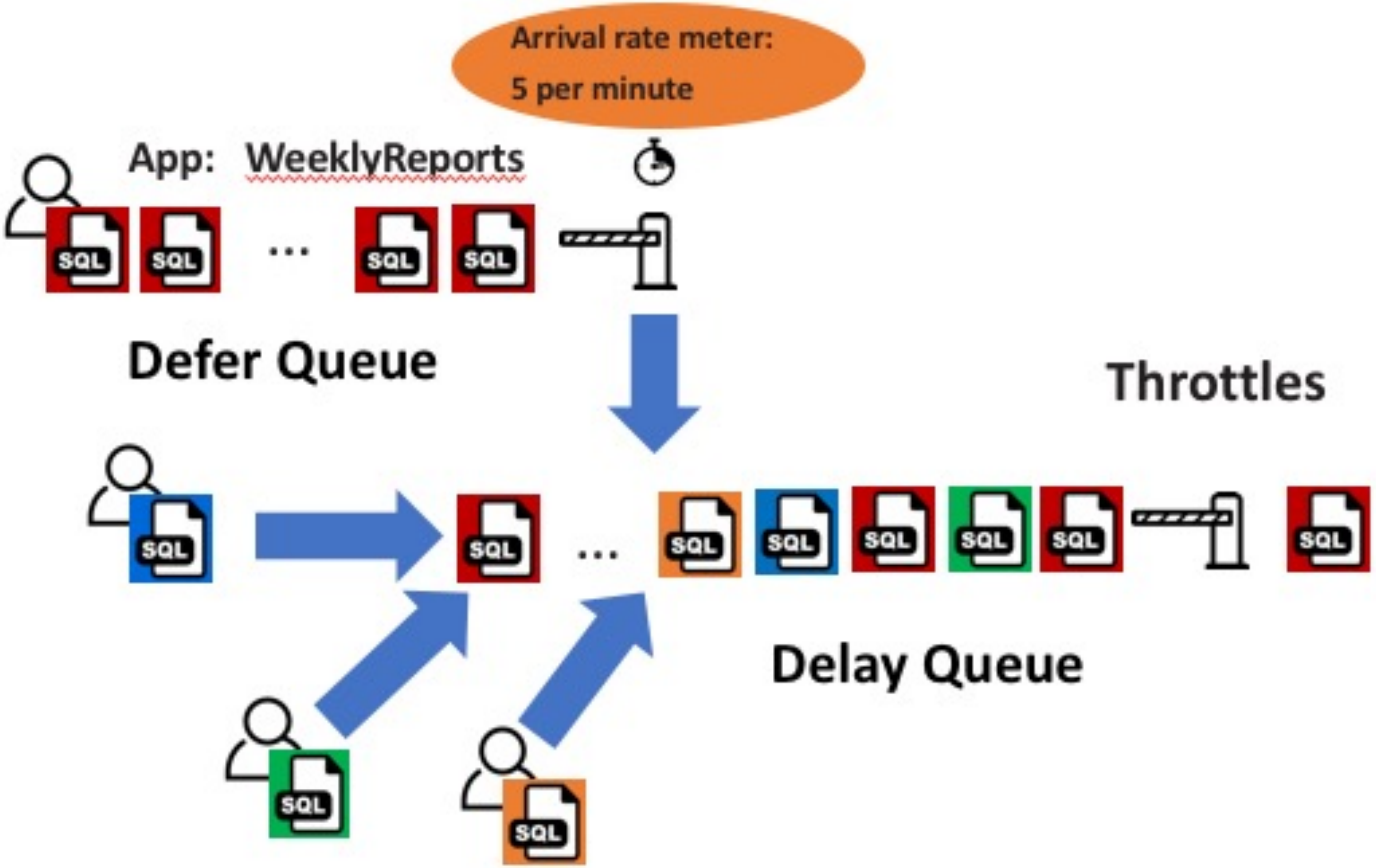
Workload Management



- You select a query limit for each throttle rule
- A counter is kept to ensure the query limit is not exceeded
- Queries that would exceed the throttle limit are placed in a delay queue

Arrival Rate Meter – New in TD17.10

Create an Arrival Rate Meter for this application so that other requests can get in between each time unit



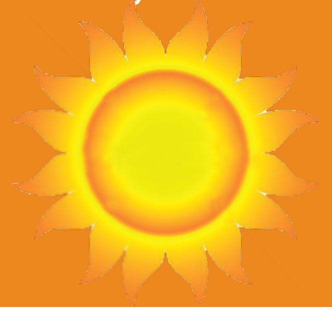
Greater balance is established in the throttle delay queue

teradata.

System Events Trigger State Changes

- A "State" encapsulates a set of "working values"
- Different TASM "states" can be defined based on:
  - Time periods or day-of-the-week
  - Level of system busy-ness (CPU utilization levels, for example)
  - When a particular type of processing completes (such as nightly loads)
- **A move from one state to another triggers setup definition changes**

Day State



Online apps get high priority

Batch work gets low priority

Night State

Online apps get low priority

Batch work gets high priority

What is a System Event?

- Triggers a change in state
- A wide variety of events are available
- Most highly-used system events:
  - AMP worker task availability
  - System CPU usage levels
  - New: I/O usage event

Settings that can be changed:

Workload priorities
Throttle query limits
Filter rules on/off
Workload Management Capacity on Demand

Sample of system-wide events:

Excessive use of CPU
Start/End of load window
Hardware component disabled
Workload arrival rates
Service level goals being missed
Delay queue depth or time in the delay queue

Resource Usage Data

ResUsageSpma
- Contains data for system-wide information by node

ResUsageSvpr
- Contains data for system-wide virtual processor information

ResUsageScpu
- Contains data specific to the CPUs within the Nodes

ResUsageShst
- Contains data specific to the host channels and TCP/IP networks communicating with the Teradata database

ResUsageSps
- Contains data regarding workload behavior statistics for utilities and SQL operations

ResUsageSawt
- Contains data regarding Amp Worker Task (AWT) behavior statistics for utilities and SQL operations

ResUsageIpma
- Contains data for system-wide information by node
- Generally used by Teradata engineers

ResUsageIvpr
- Contains data for system-wide virtual processor information
- Generally used by Teradata engineers

ResUsageSldv
- Contains data for system-wide logical device data collected from storage devices

ResUsageSpdsk
- Contains data specific to physical disk storage statistics

ResUsageSvdsk
- Contains data specific to amp-level disk storage statistics

Collection and Logging Standard

✓ 60 - 600 second collection interval
✓ 60 - 600 second logging interval
✓ Use Active Filter Enable for AWT and SPS
✓ Use Summary mode for vdsk if enabled

DBQL Data

DBQLLogTbl
- Contains detail query data (CPU, IO, Spool, etc.)
- One row / query

DBQLObjTbl
- One row per object referenced in query
- Identify unused / under-used objects.

DBQLSqlTbl
- Contains full SQL text
- May record more than one row

DBQLStepTbl
- Contains Plan and Actual Step level data
- One row per query step

DBQLExplainTbl
- Contains Full EXPLAIN text
- May record more than one row

DBQLXMLTbl
- Contains optimizer query plans, statistics recommendation as XML format

DBQLXMLLockTbl
- Contains query lock contention as XML format

DBQLParamTbl
- Contains values of the parameters for a parameterized query

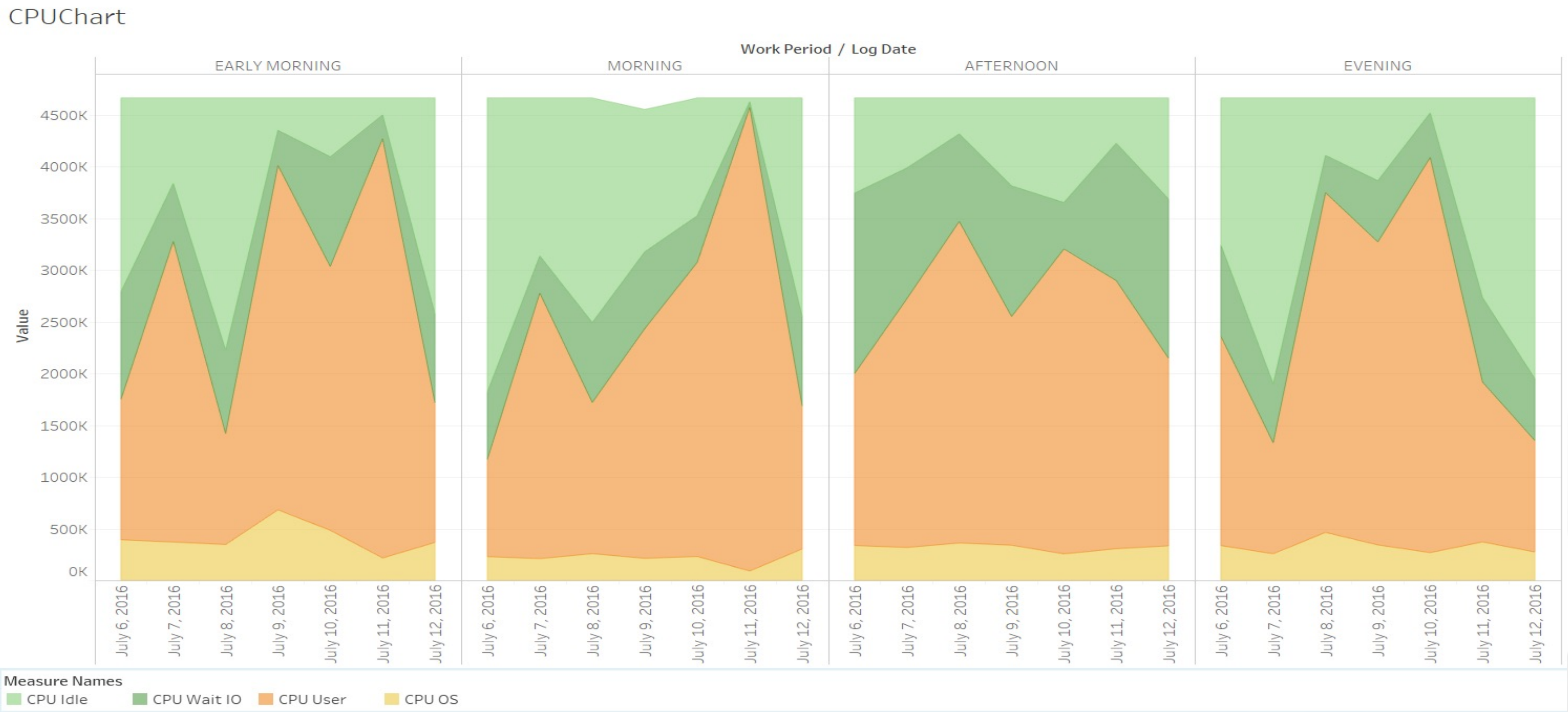
Increasing level of detail

DBQLSummaryTBL*
- Populated by Summary or Threshold Logging
- Aggregated data for 10 minute time period

DBQLUtilityTbl *
- Contains for each individual utility job: jobname, utility name, user, queryband, LSN, timing by phases, rows/block count, CPU, IO , max AWT usage
- **How does this join???**

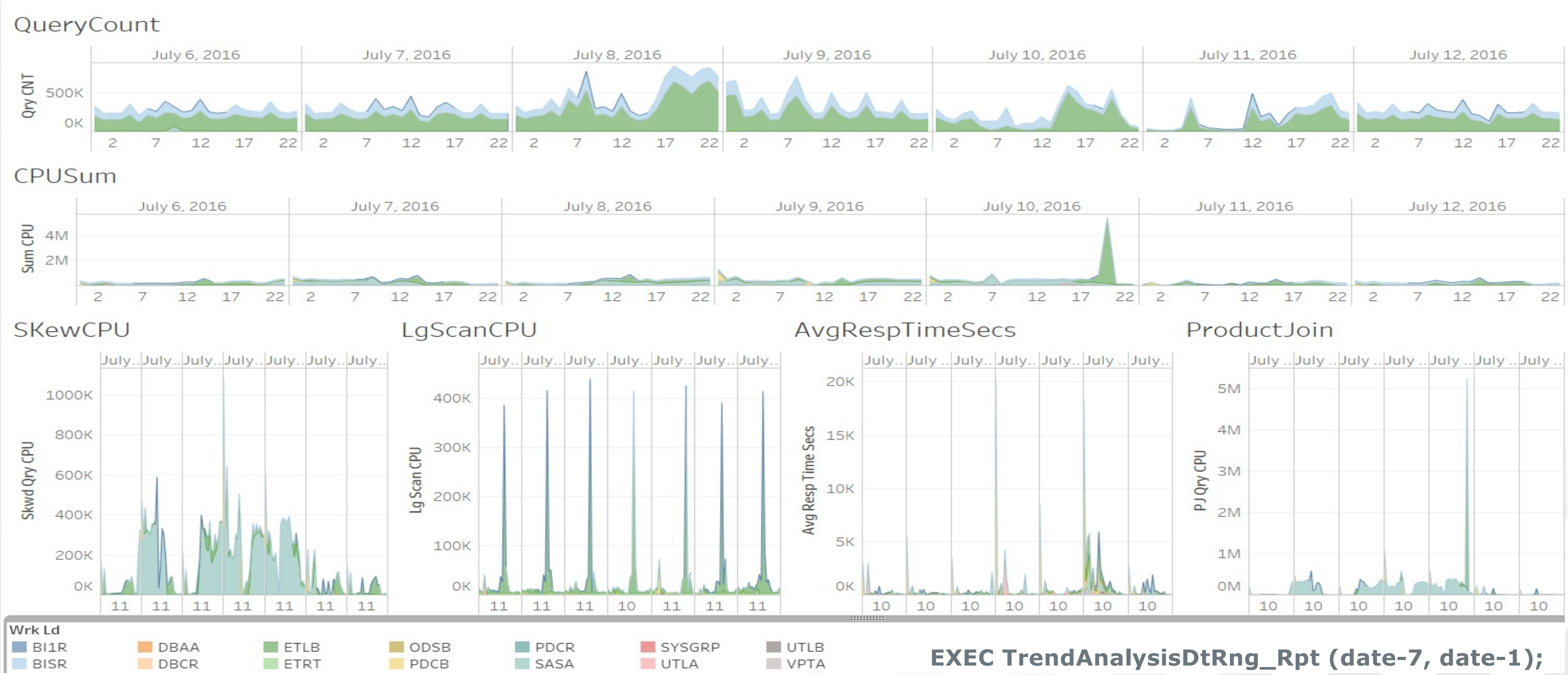
All tables except * joined on queryid and procid.

System Performance Sample



System Performance Sample

Application Performance Sample



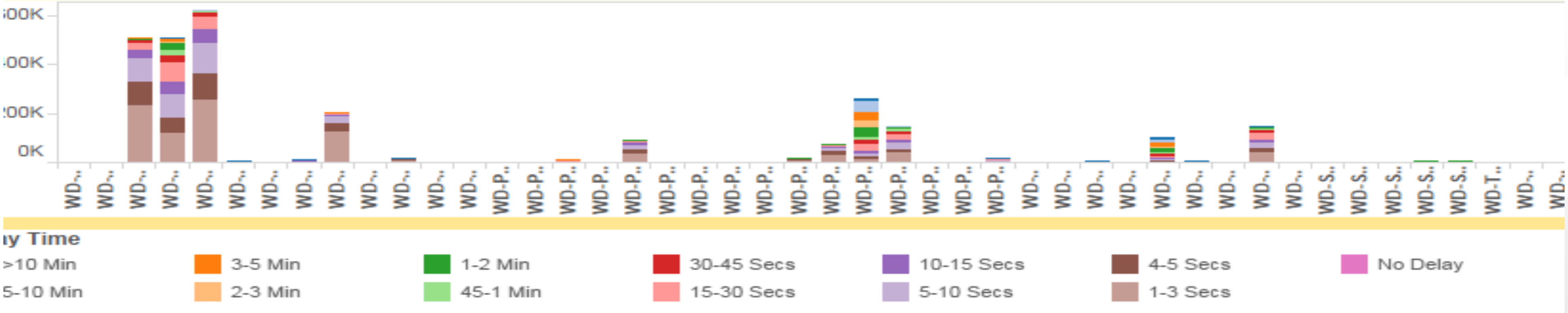
EXEC TrendAnalysisDtRng_Rpt (date-7, date-1);
(Macro in Bonus Section)

Workload Management Sample

Percent of Overall delays by Day / Hour

nth, Da..	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
y 16, 2017	1%																							
y 15, 2017	2%	4%	2%	1%	0%	2%	5%	10%	11%	6%	1%	3%	3%	3%	2%	1%	0%	0%	0%	2%	0%	1%	8%	3%
y 14, 2017	0%	1%	0%	0%	3%	0%	0%	2%	5%	3%	4%	3%	8%	2%	4%	3%	2%	4%	4%	10%	2%	3%	6%	0%
y 13, 2017	0%	1%	3%	0%	1%	1%	2%	2%	0%	0%	0%	0%	1%	0%	0%	0%	0%	0%	14%	11%	0%	0%	0%	0%
y 12, 2017	0%	2%	3%	0%	8%	12%	3%	4%	3%	4%	5%	7%	2%	0%	0%	0%	0%	2%	2%	5%	0%	0%	0%	0%
y 11, 2017	0%	2%	3%	1%	3%	14%	22%	15%	10%	3%	1%	2%	0%	1%	0%	0%	0%	0%	1%	1%	0%	0%	0%	0%
y 10, 2017	0%	3%	3%	2%	2%	12%	11%	17%	16%	13%	15%	12%	4%	3%	2%	4%	2%	1%	1%	3%	0%	0%	2%	1%
y 9, 2017	0%	2%	2%	0%	3%	4%	7%	6%	4%	7%	2%	2%	6%	16%	14%	7%	2%	2%	0%	1%	0%	0%	2%	0%
y 8, 2017	1%	1%	3%	0%	3%	2%	11%	24%	6%	11%	8%	6%	7%	11%	3%	3%	2%	1%	0%	3%	1%	0%	2%	0%

Delays by Workload Type



Priority Scheduler

SLES11 – Virtual Partitions

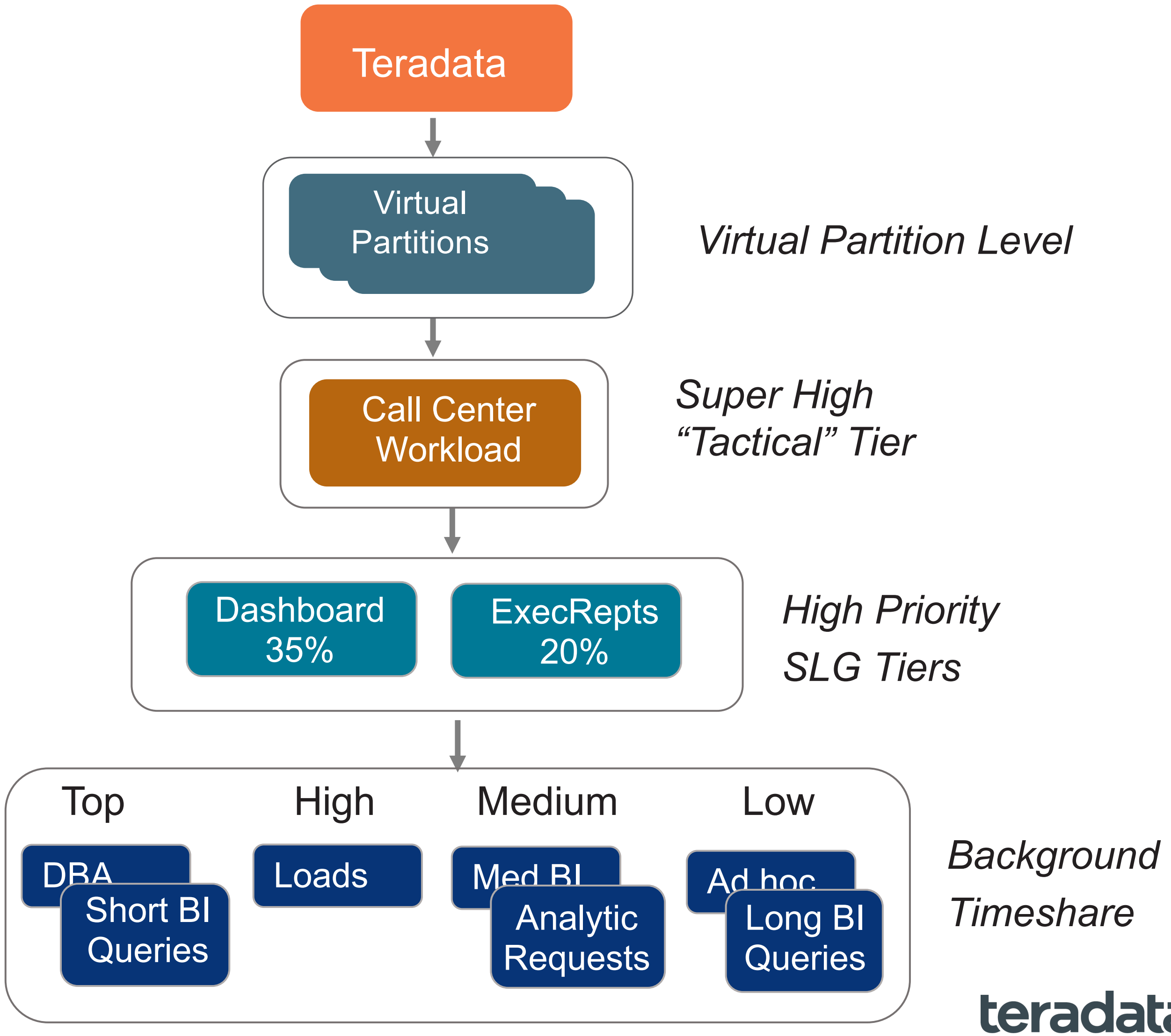
Finance 60%	Marketing 30%	Sales 10%
Finance Workload Management Model	Marketing Workload Management Model	Sales Workload Mgt Model

- Values indicate the target % of CPU or I/O for that partition
- Up to 10, default of 1
- TASM only feature

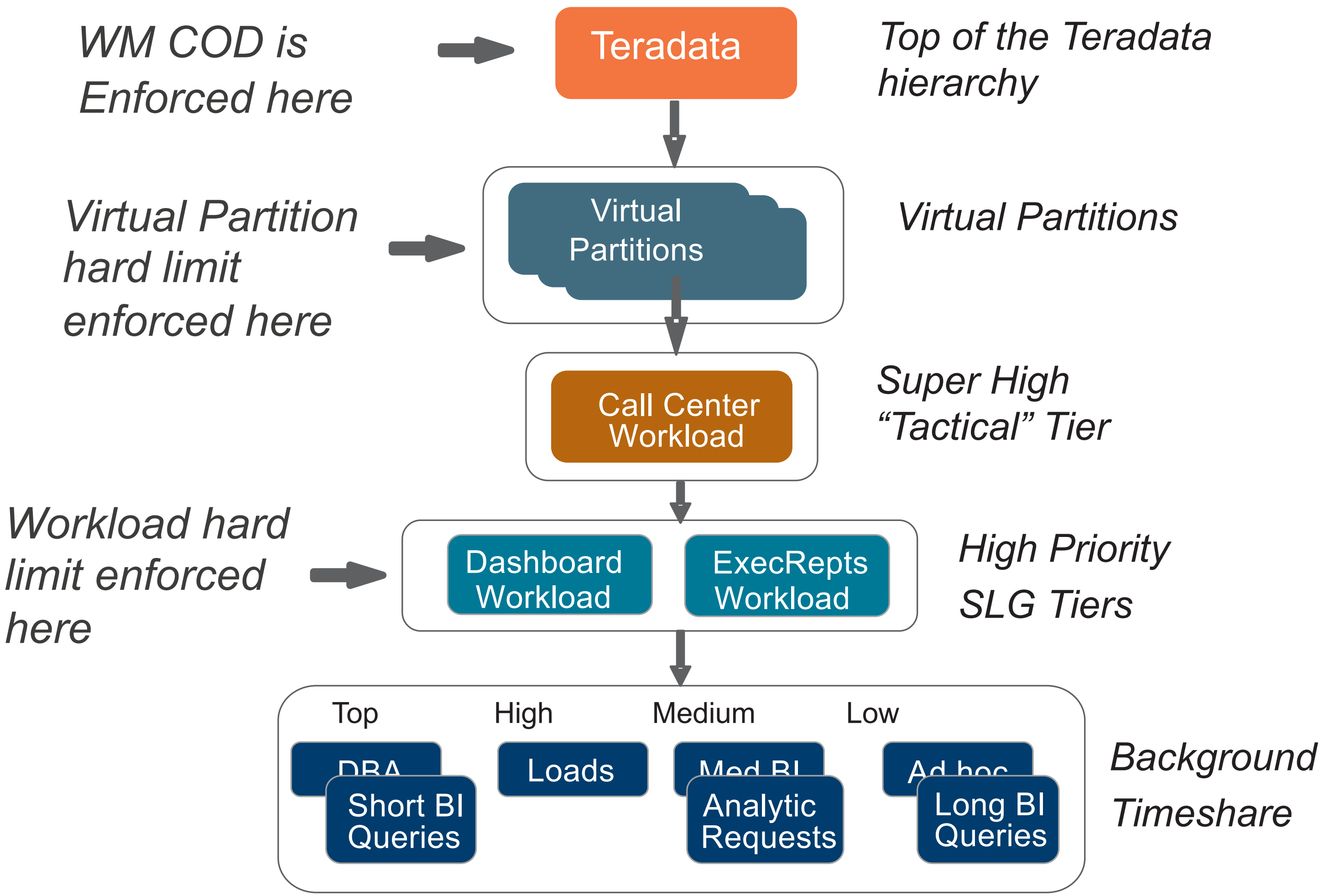
teradata.

TASM Workloads are Placed in the Priority Hierarchy

- The "workload" is the priority component
- Hierarchical-based priority definitions
- Resources flow from the top tiers downward
- A special very high priority "Tactical Tier"
- Simple-to-use GUI tool
- Includes both CPU and I/O prioritization

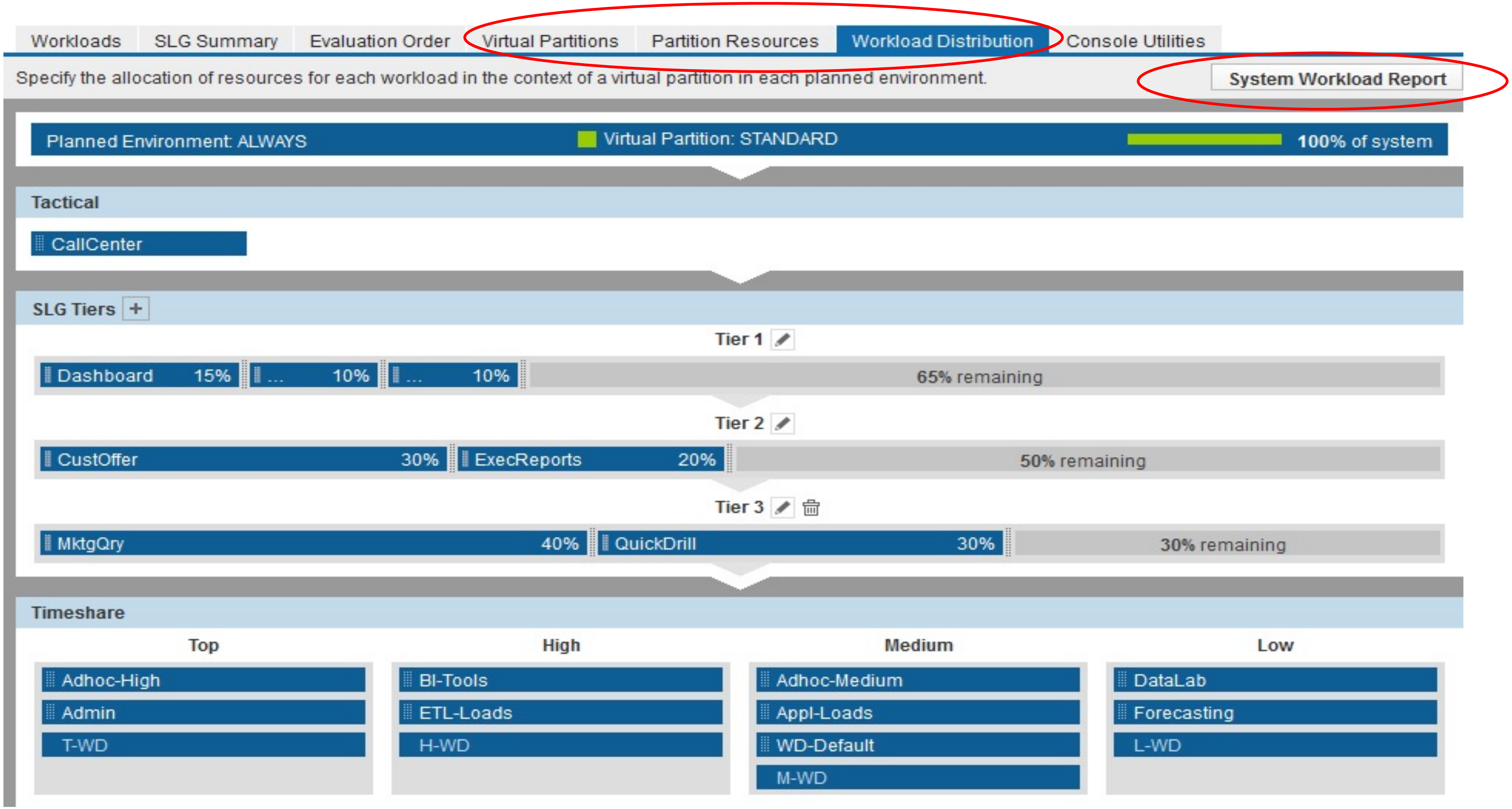


Hard Limits at Multiple Levels



- WM COD puts a hard limit at the top of the Teradata priority hierarchy
- Defining a "fixed" VP enforces a hard limit at the VP's defined allocation %
- SLG Tier workloads can have hard limits of any percent defined
  - Timeshare and tactical workload do not support hard limits

Viewpoint Workload Designer with SLES11



Teradata 14.0 SLES 11 Priority Scheduler

System Workload Report

View workload resource allocations across all virtual partitions. Translation of workload allocations to system resource percentages does not consider workload hard limits. To sort on a second column, hold the Control/Command key.

Planned Environment: **Always**

Virtual Partitions

Standard	90%	Ne...	10%

Workloads

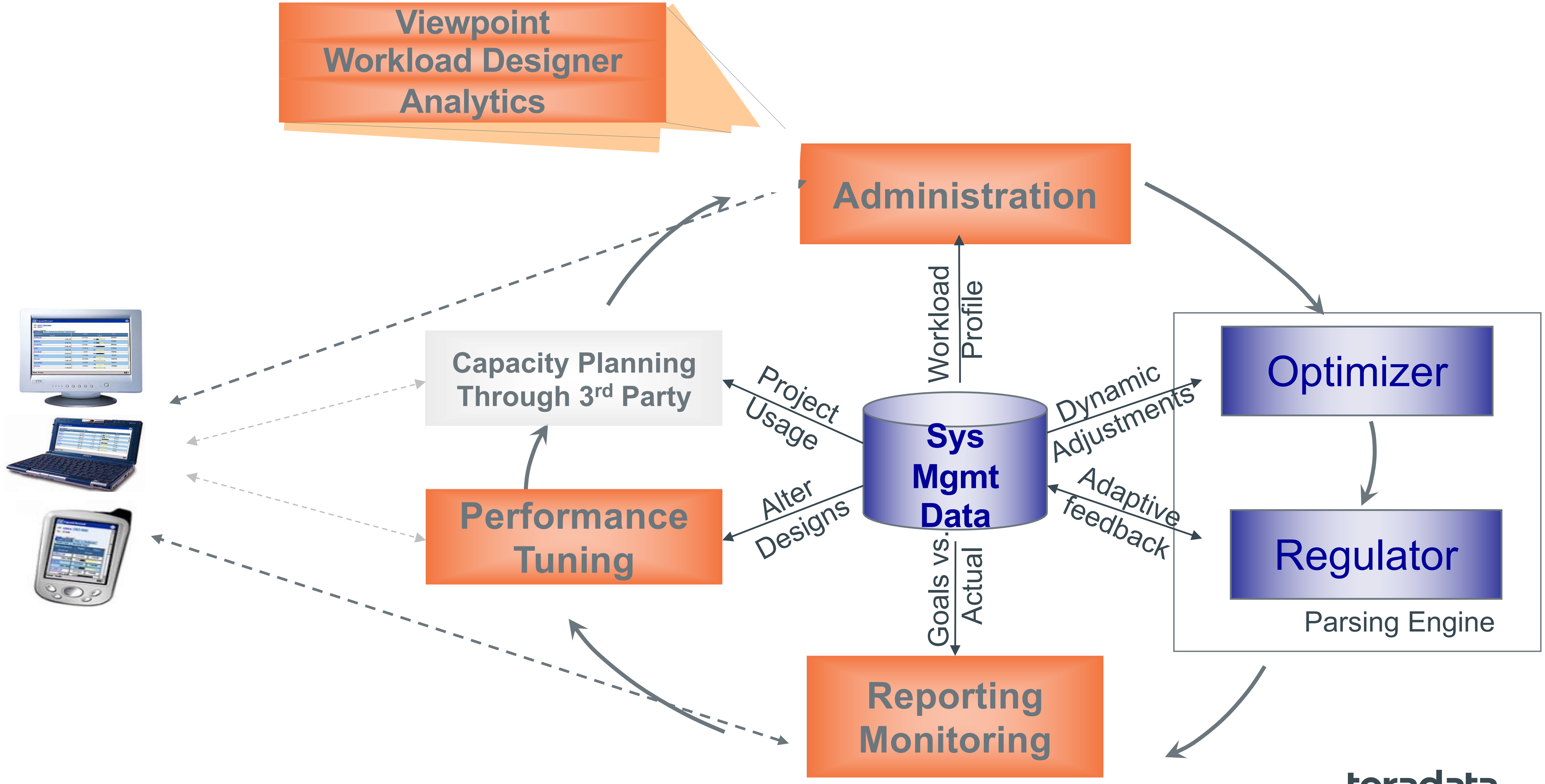
WORKLOAD	VIRTUAL PARTI.	TIER	% OF TIER	% OF SYSTEM
slg1	NewVP	Tier 1	13.6	1.4
slg2	NewVP	Tier 2	26	2.2
ts-top	NewVP	TS1 Timeshare Top		2.2
ts-top2	NewVP	TS1 Timeshare Top		2.2
ts-high	NewVP	TS2 Timeshare High		1.1
ts-med	NewVP	TS3 Timeshare Medium		0.6
ts-low	NewVP	TS4 Timeshare Low		0.3
webApp1	Standard	Tier 1	10	9
webApp2	Standard	Tier 1	10	9
dashboard	Standard	Tier 1	15	13.5
CustOffer	Standard	Tier 2	30	17.6
ExecRpts	Standard	Tier 2	20	11.7
MktQry	Standard	Tier 3	40	11.7
QuickDrill	Standard	Tier 3	30	8.8
SLGapp1	Standard	Tier 4	5	0.4
SLG5app1	Standard	Tier 5	5	0.4
T-WD	Standard	TS1 Timeshare Top		2.9
wd1	Standard	TS2 Timeshare High		1.4

23 rows total

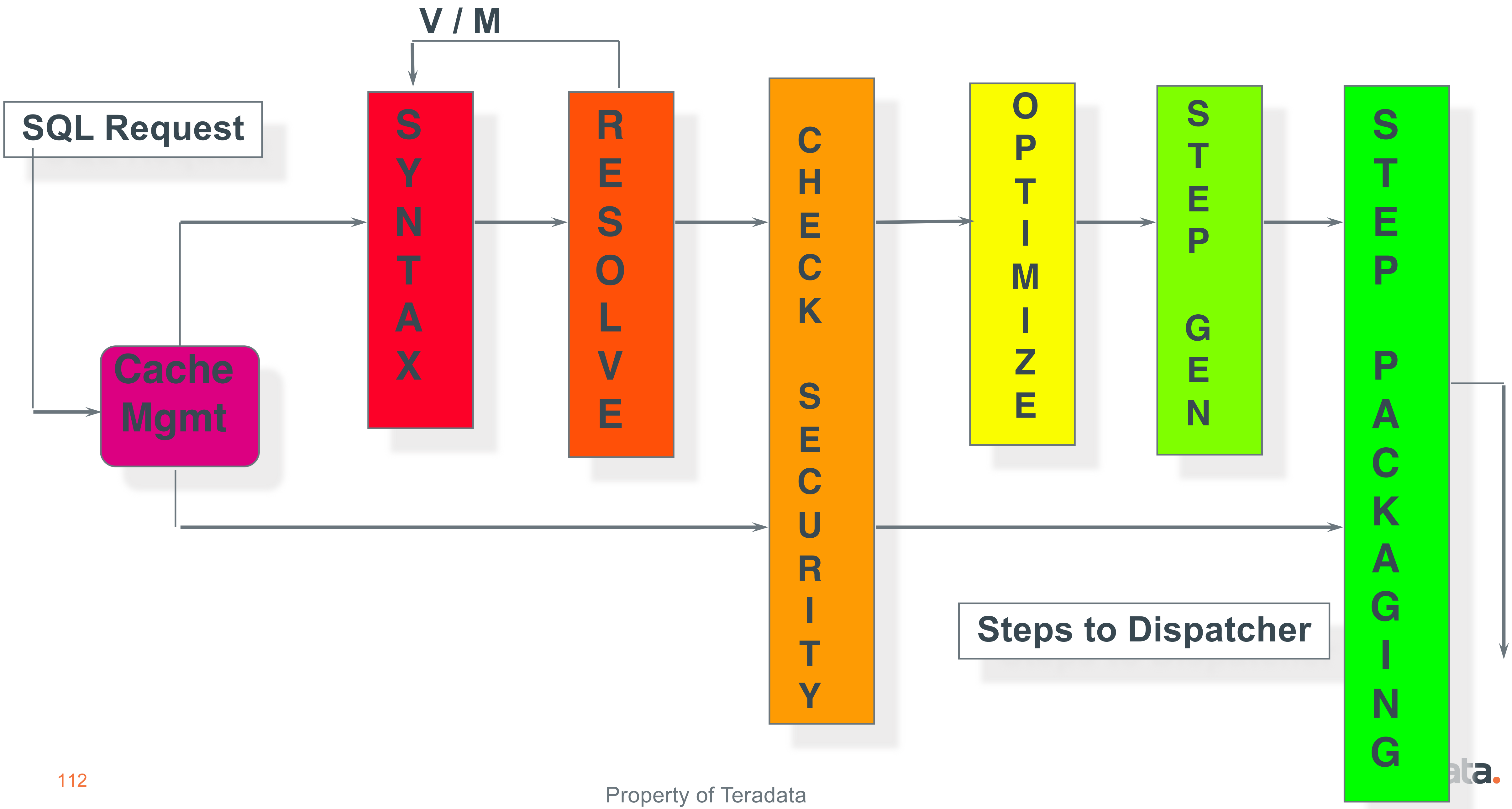
Close

TASM Architecture

TASM – High level Architecture flow

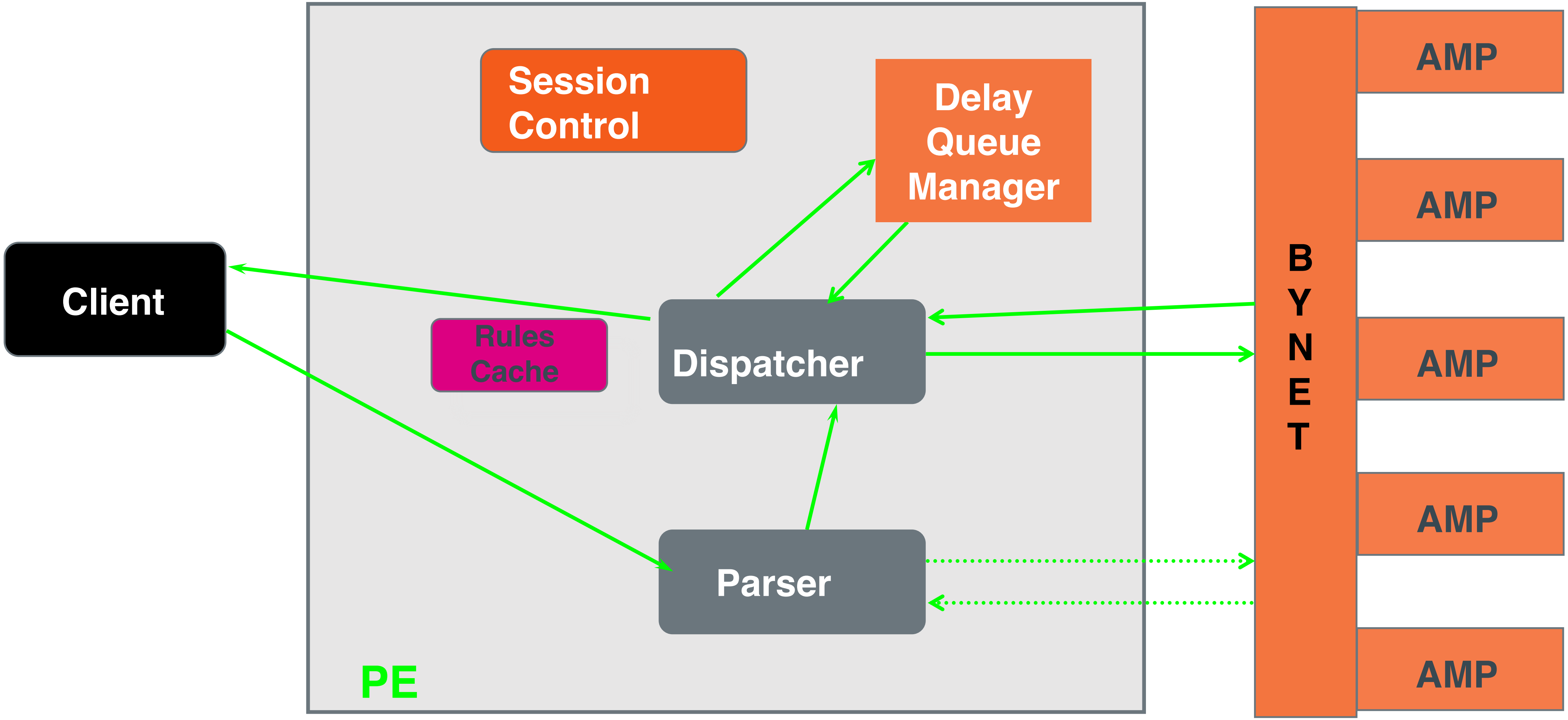


Parser Flow



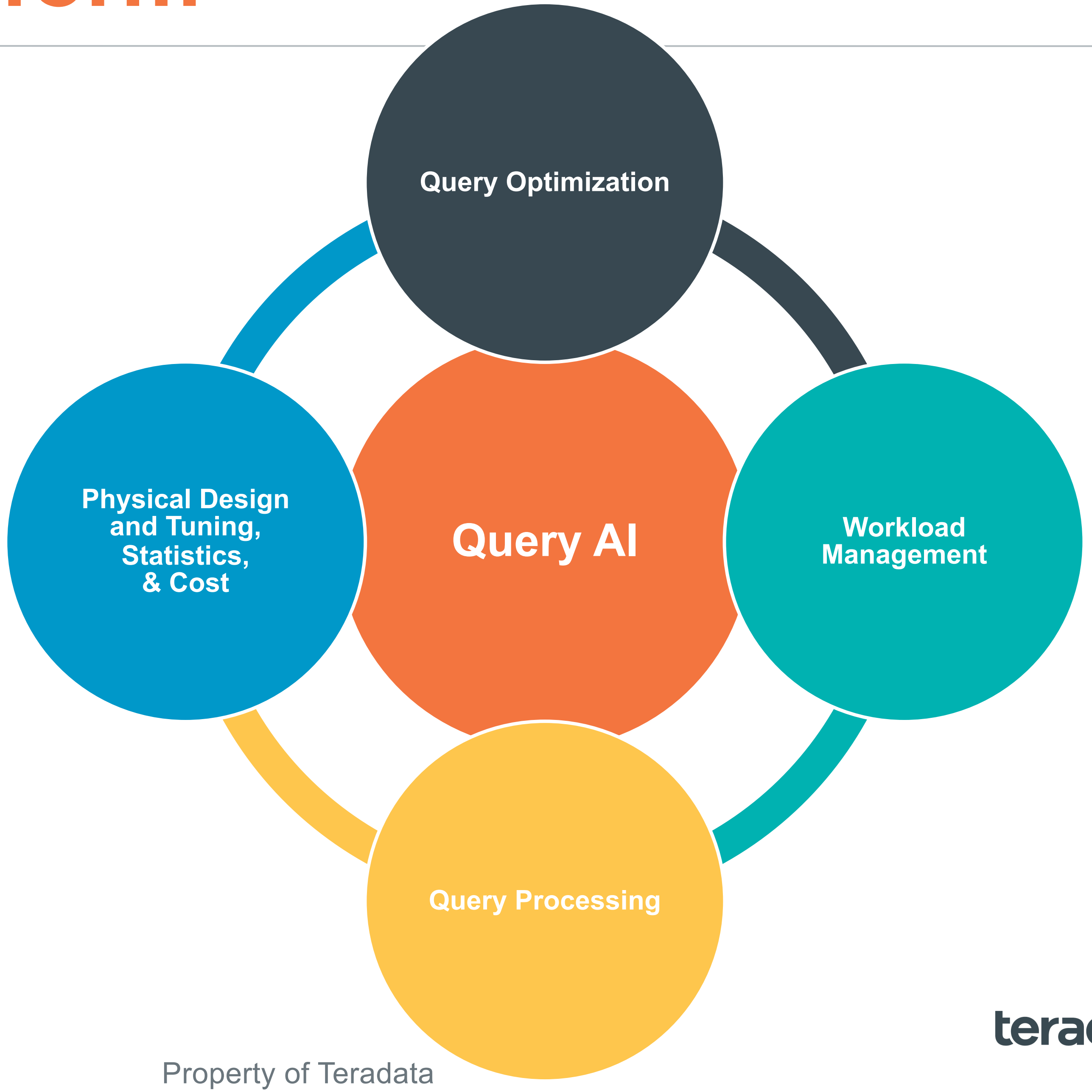
V / M
SQL Request
Cache Mgmt
SYNTAX
RESOLVE
CHECK SECURITY
OPTIMIZE
STEP GEN
STEP PACKAGING
Steps to Dispatcher

Parser Request Flow



Autonomous Platform

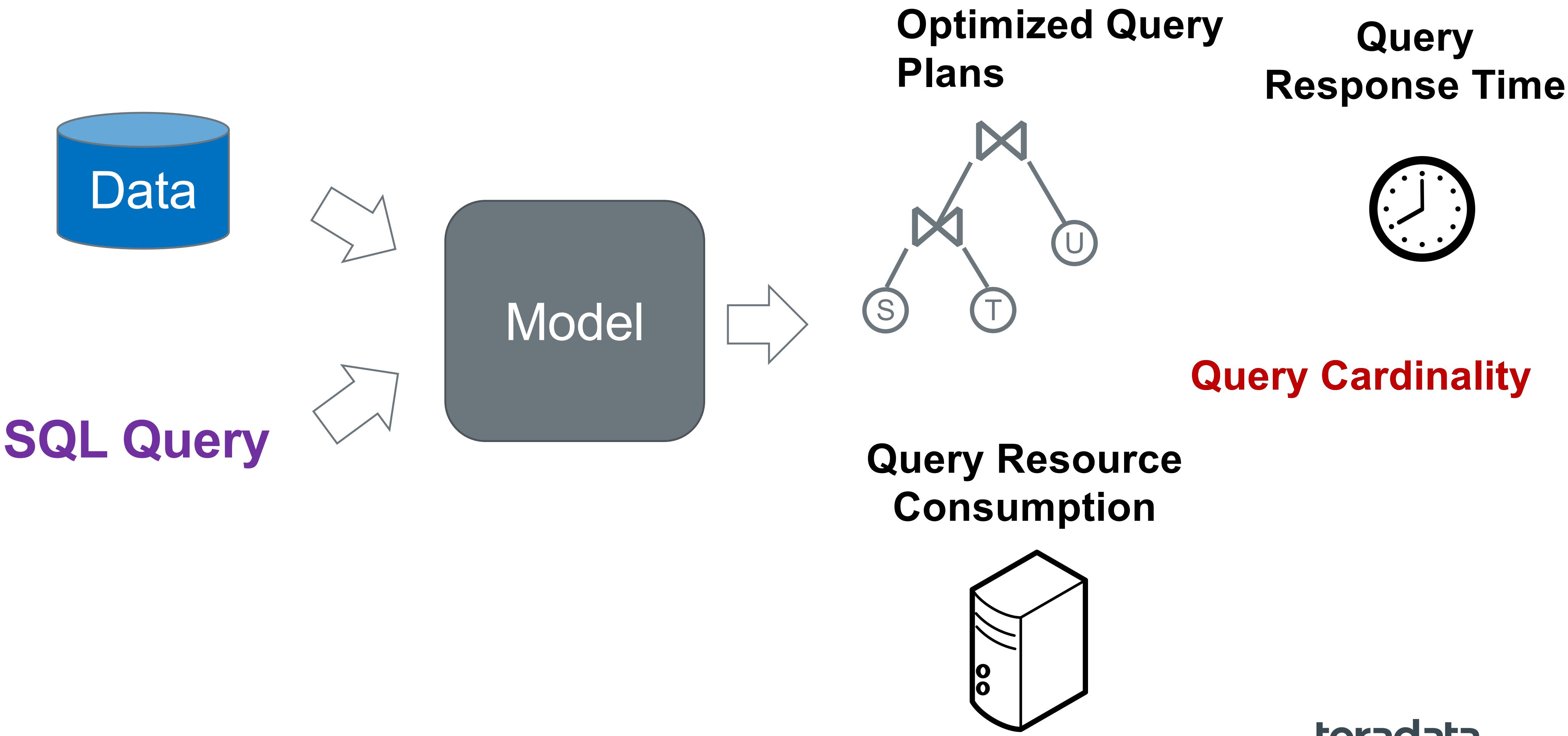
- Artificial Intelligence for Queries
  - Physical design and tuning
  - Statistics
  - Costing
  - Optimization
  - Workload Management
  - Query Processing
- Learning at the Edge
- Focused ML Models
- Automation
- Closed loop



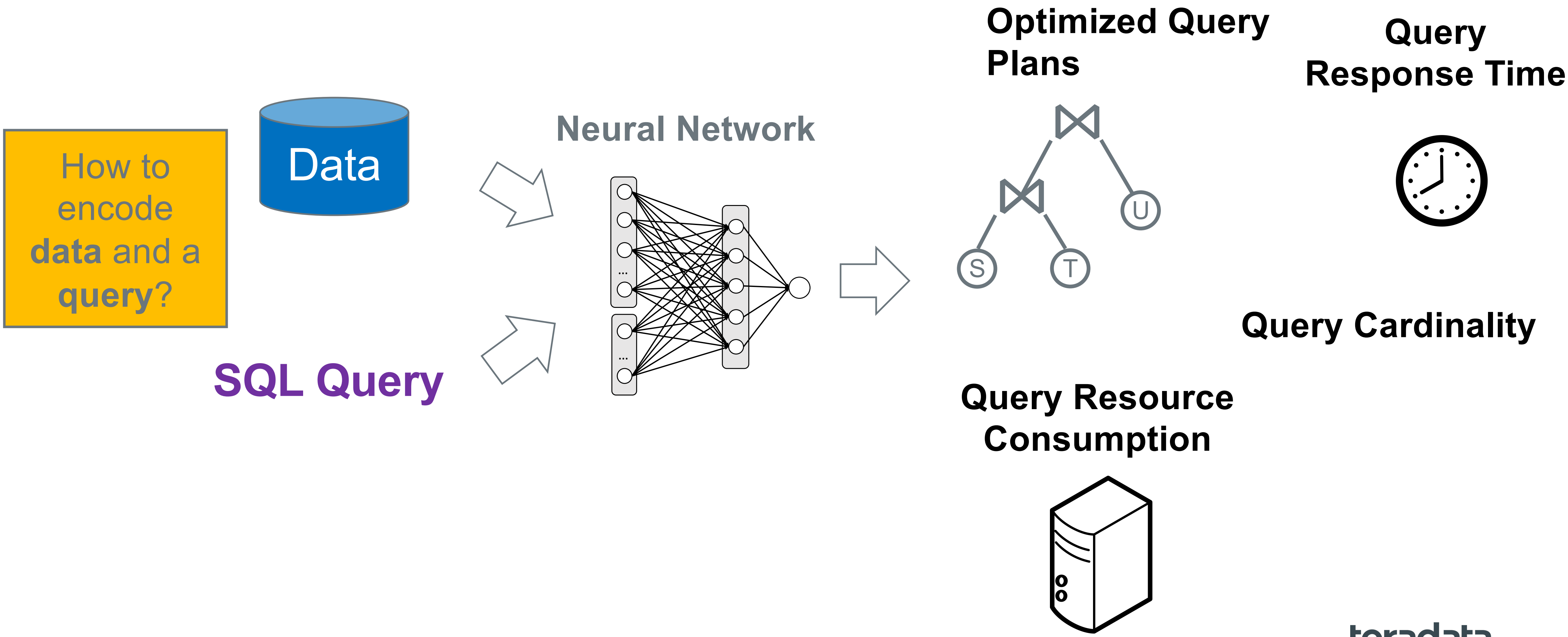
teradata.

Our Vision

Can we rethink **Query Optimization and WLM** in the context of **Deep Learning**?

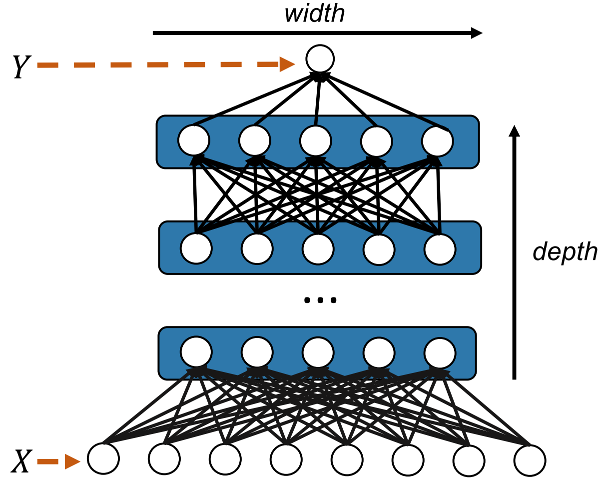


Deep Learning with Query Optimization

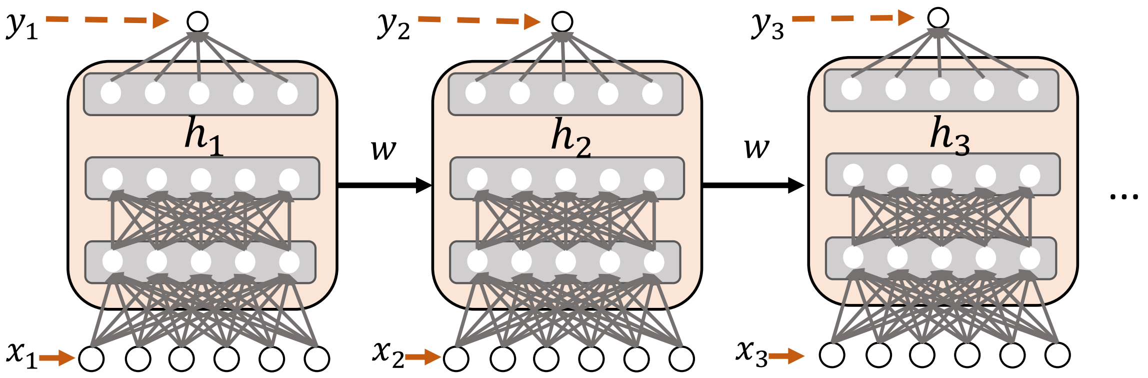


Models

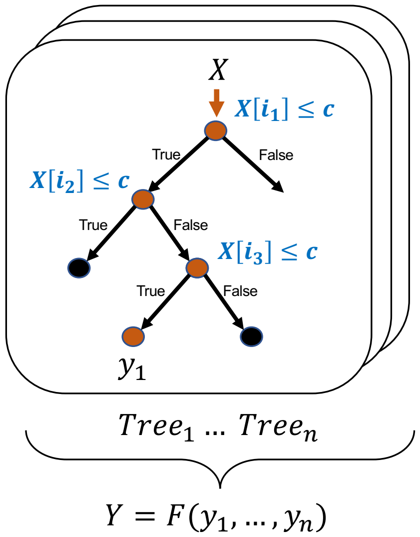
Neural Network



Recurrent Neural Network



Random Forest



teradata.

Encoding the Input

Example:

Orders ⋈ Customers

Data

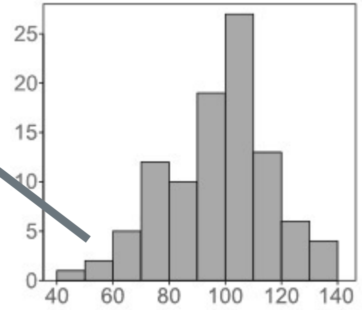
Basic Statistics:
1D-Histograms

Input Array

3

5

histogram of
Orders column



Need ability to encode all possible queries!

…on customers and orders

…on customers and region

SQL Query

Encode Join Pred…
Encode Selection…

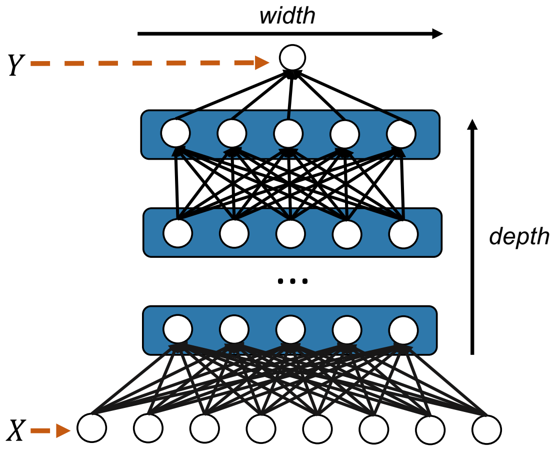
teradata.

Encoding a Query

Example dataset:

- **Relations :** A, B, C
- Each has two columns:
  - A: a1 and a2
  - B: b1 and b2
  - C: c1 and c2

Neural Network

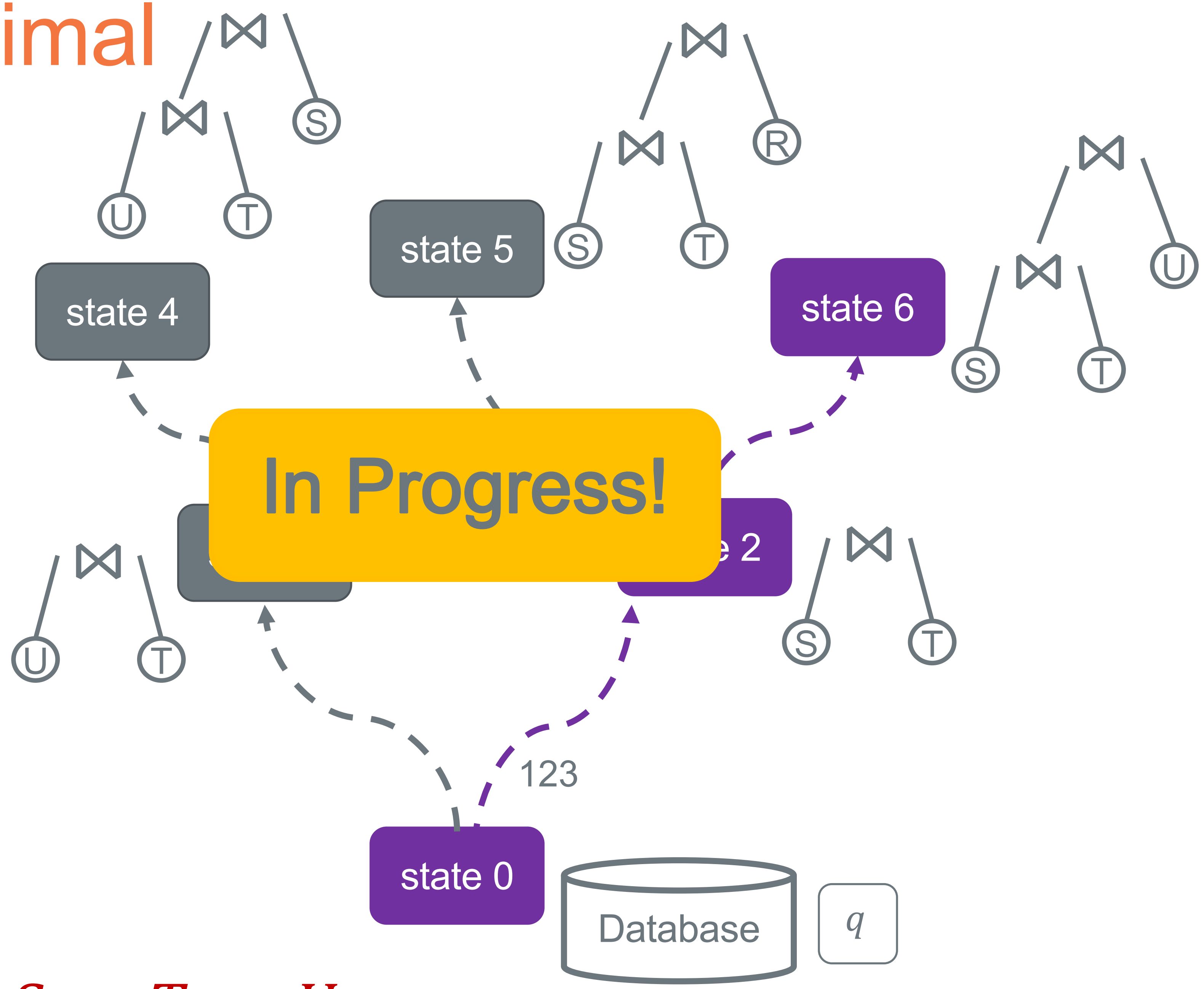


q: SELECT * FROM **A,B** WHERE **a_1 <= 10** and **a_2=b_1**

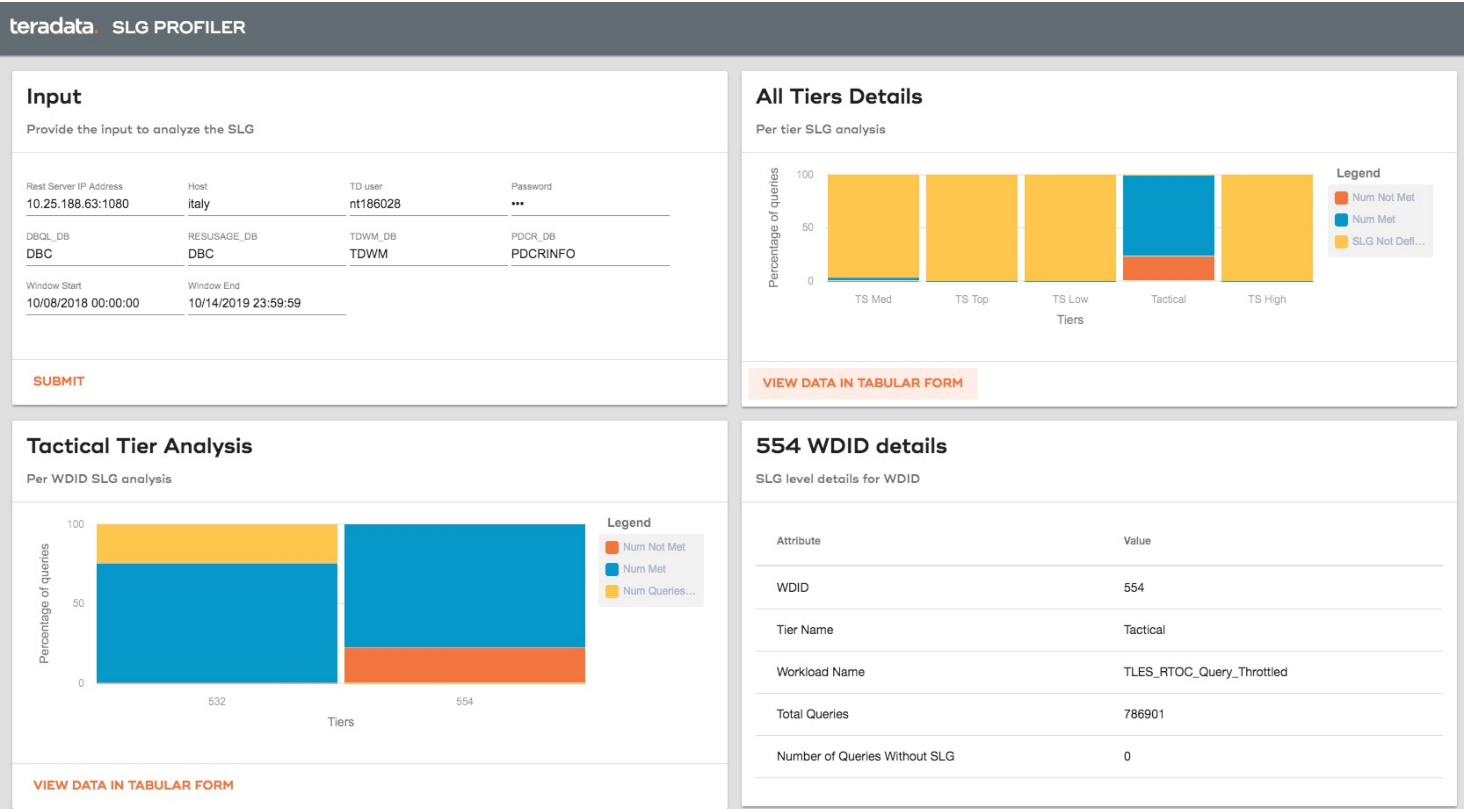
1	1	0	.1	1	1	1	0	0	1	0
A	B	C	a_1	a_2	b_1	b_2	c_1	c_2	a_2=b_1	b_2=c_1

$l_{relation}$ $l_{selpred}$ $l_{joinpred}$

Learning Optimal Paths



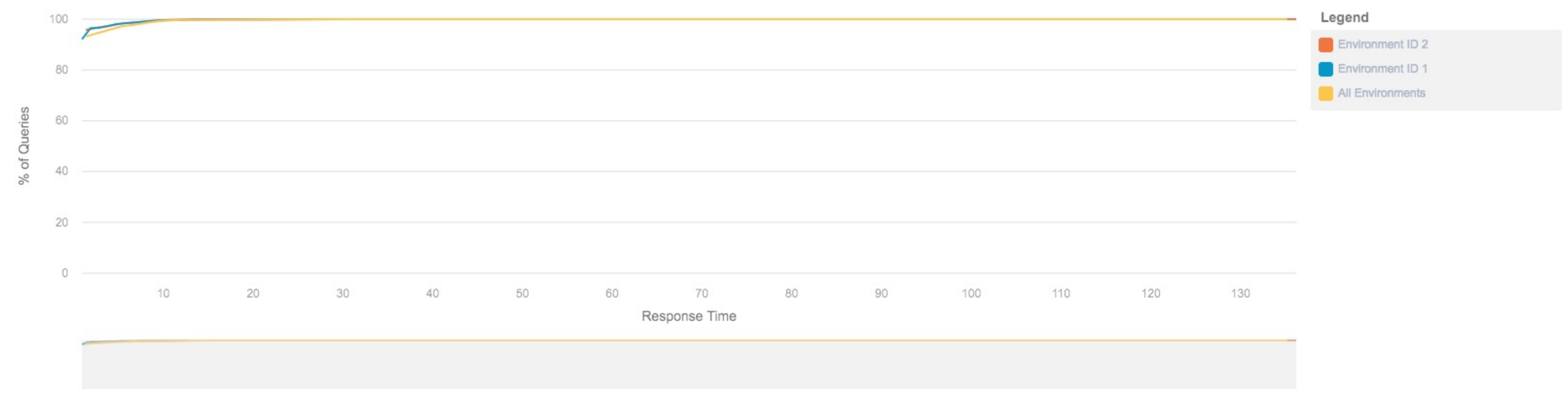
Example: $S \bowtie T \bowtie U$


teradata. SLG PROFILER
Input
Provide the input to analyze the SLG
Rest Server IP Address
10.25.188.63:1080
Host
italy
TD user
nt186028
Password
•••
DBQL_DB
DBC
RESUSAGE_DB
DBC
TDWM_DB
TDWM
PDCR_DB
PDCRINFO
Window Start
10/08/2018 00:00:00
Window End
10/14/2019 23:59:59
SUBMIT
All Tiers Details
Per tier SLG analysis
Percentage of queries
100
50
0
TS Med
TS Top
TS Low
Tactical
TS High
Tiers
Legend
Num Not Met
Num Met
SLG Not Defi...
VIEW DATA IN TABULAR FORM
Tactical Tier Analysis
Per WDID SLG analysis
Percentage of queries
100
50
0
532
554
Tiers
Legend
Num Not Met
Num Met
Num Queries...
VIEW DATA IN TABULAR FORM
554 WDID details
SLG level details for WDID
Attribute
Value
WDID
554
Tier Name
Tactical
Workload Name
TLES_RTOC_Query_Throttled
Total Queries
786901
Number of Queries Without SLG
0

teradata. SLG PROFILER

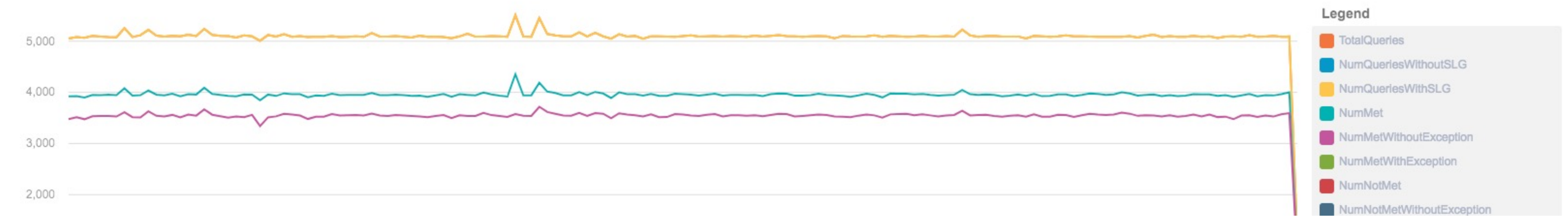
Response time distribution for WDID 554

Response time distribution



Query Statistics for WDID 554

SLG related statistics



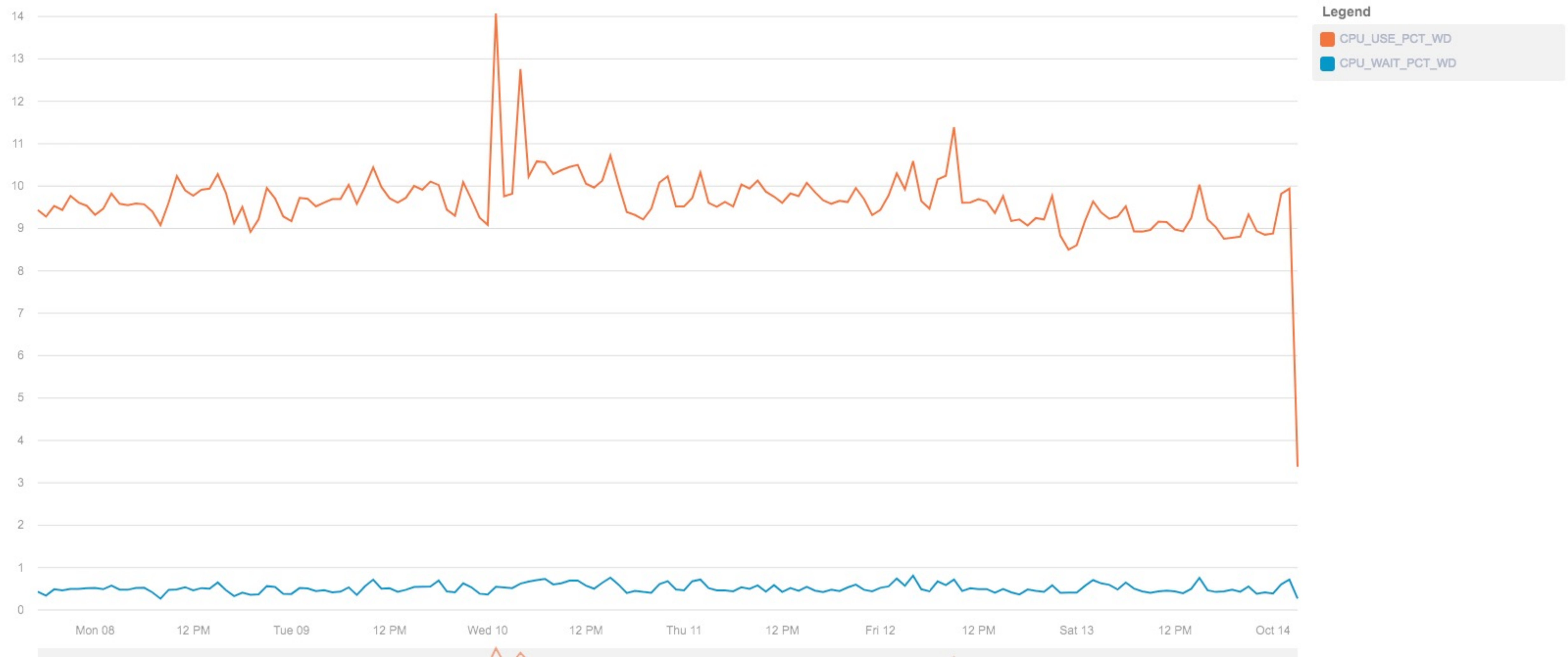
teradata. SLG PROFILER

Resource utilization for WDID 554

CPU, IO, AWT and other resource details

CPU Utilization | AWT | I/O | Request Count | Flow Control | Lock Wait | Other Wait

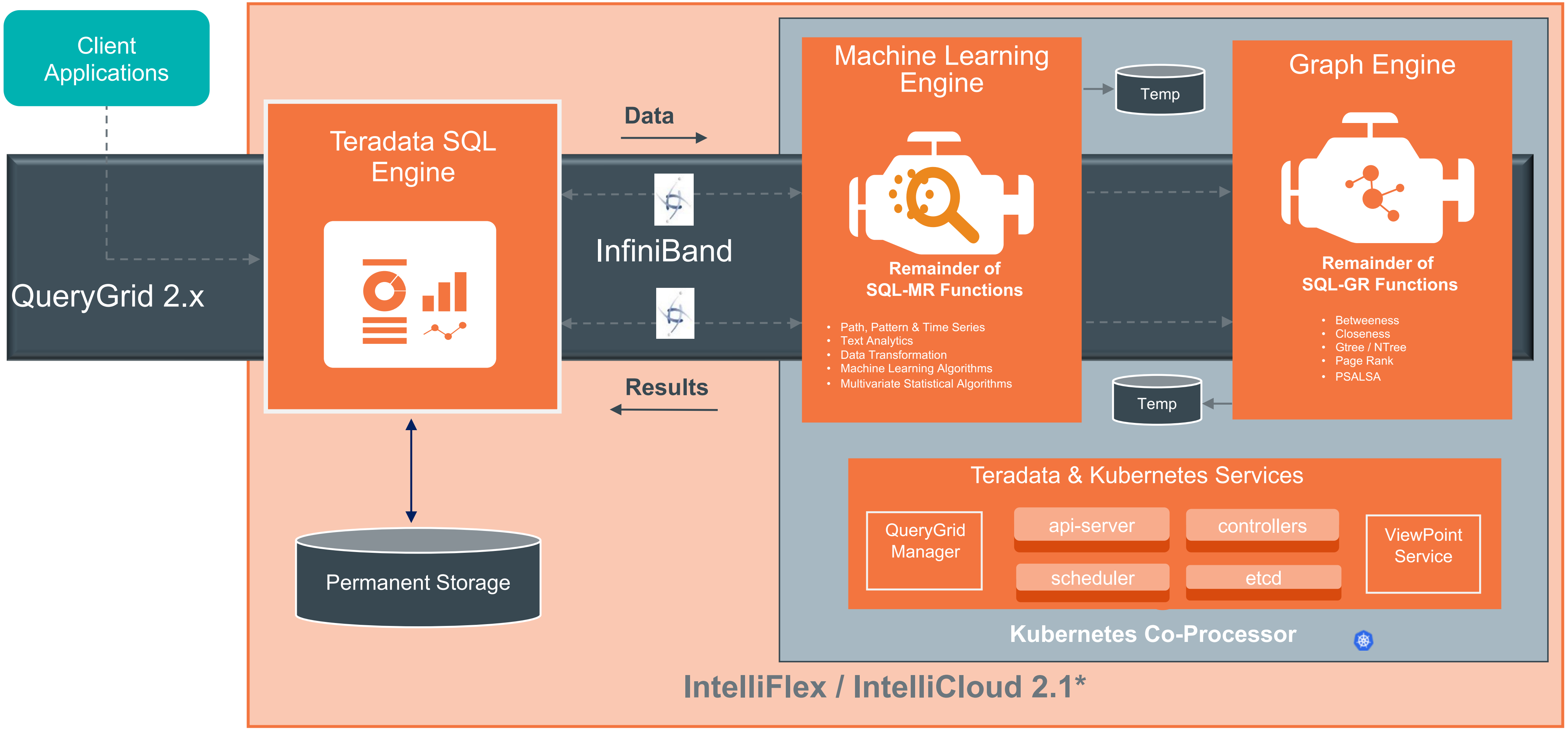
WD: 554



Teradata Vantage – Information Resources

- YouTube Teradata Database 101
- www.teradata.com
- https://www.teradatauniversitynetwork.com
- https://www.teradata.com/Resources/TEN

Teradata Vantage



teradata.

Thank you.

teradata.