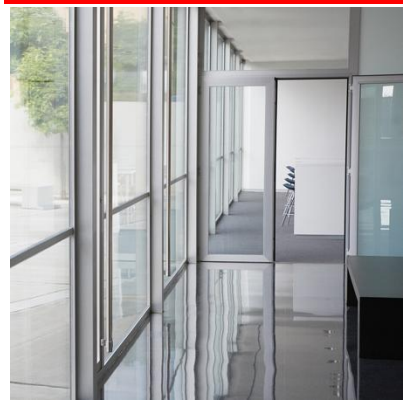

ORACLE®


ORACLE®

# Best Practices for Extreme Performance with Data Warehousing on Oracle Database

Rekha Balwada
Principal Product Manager

# Agenda



- Parallel Execution
- Workload Management on Data Warehouse
- Oracle Exadata Database Machine

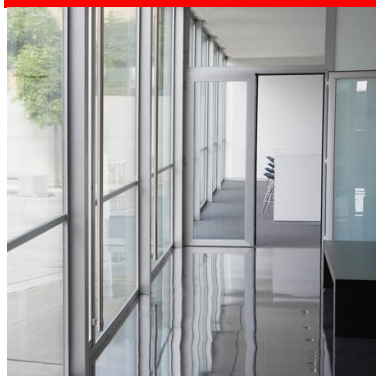
# Best Practices for Data Warehousing

## 3 Ps - Power, Partitioning, Parallelism

- **Power** – A Balanced Hardware Configuration
  - Weakest link defines the throughput
- **Partition** larger tables or fact tables
  - Facilitates data load, data elimination and join performance
  - Enables easier Information Lifecycle Management
- **Parallel** Execution should be used
  - Instead of one process doing all the work multiple processes working concurrently on smaller units
  - Parallel degree should be power of 2

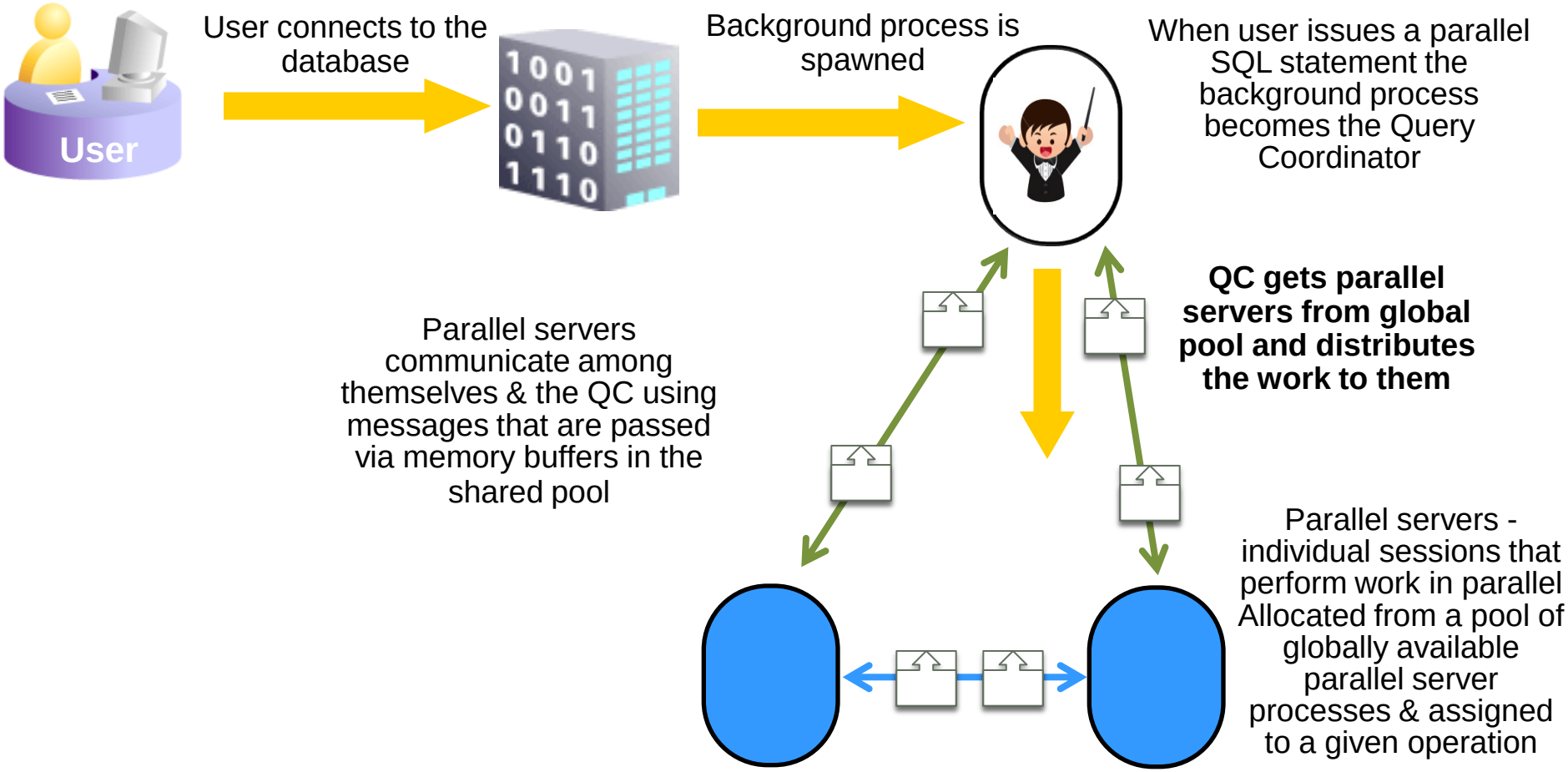
**Goal is to minimize the amount of data accessed and use the most efficient joins**

# Agenda



- Parallel Execution
- Workload Management on a Data Warehouse
- Oracle Exadata Database Machine

# How Parallel Execution works



# Monitoring Parallel Execution

```
SELECT  c.cust_last_name, s.time_id, s.amount_sold
FROM sales s, customers c
WHERE s.cust_id = c.cust_id;
```

Query Coordinator

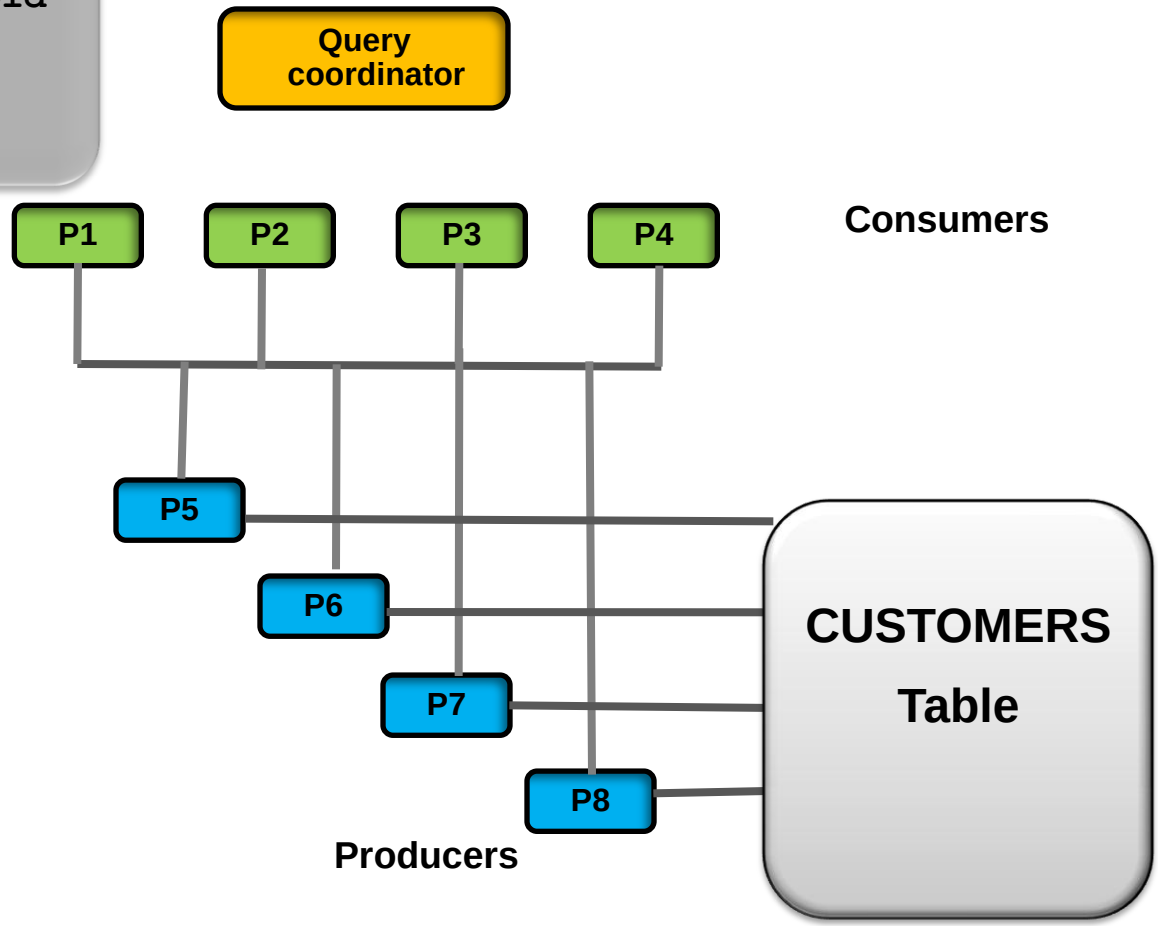
| Id | Operation | Name | Bytes | Cost (%CPU) | Time |
|---|---|---|---|---|---|
| 0 | SELECT STATEMENT | | | 311 (100) | |
| 1 | PX COORDINATOR | | | | |
| 2 | PX SEND QC (RANDOM) | :TQ10002 | 1049K | 31M 311 (2) | 00:00:04 |
| * 3 | HASH JOIN BUFFERED | | 1049K | 31M 311 (2) | 00:00:04 |
| 4 | PX RECEIVE | | 55500 | 704K 112 (0) | 00:00:02 |
| 5 | PX SEND HASH | :TQ10000 | 55500 | 704K 112 (0) | 00:00:02 |
| 6 | PX BLOCK ITERATOR | | 55500 | 704K 112 (0) | 00:00:02 |
| * 7 | TABLE ACCESS FULL | CUSTOMERS | 55500 | 704K 112 (0) | 00:00:02 |
| 8 | PX RECEIVE | | 1049K | 18M 196 (2) | 00:00:03 |
| 9 | PX SEND HASH | :TQ10001 | 1049K | 18M 196 (2) | 00:00:03 |
| 10 | PX BLOCK ITERATOR | | 1049K | 18M 196 (2) | 00:00:03 |
| * 11 | TABLE ACCESS FULL | SALES | | 18M 196 (2) | 00:00:03 |

Parallel Servers

do majority of the work

# How Parallel Execution works

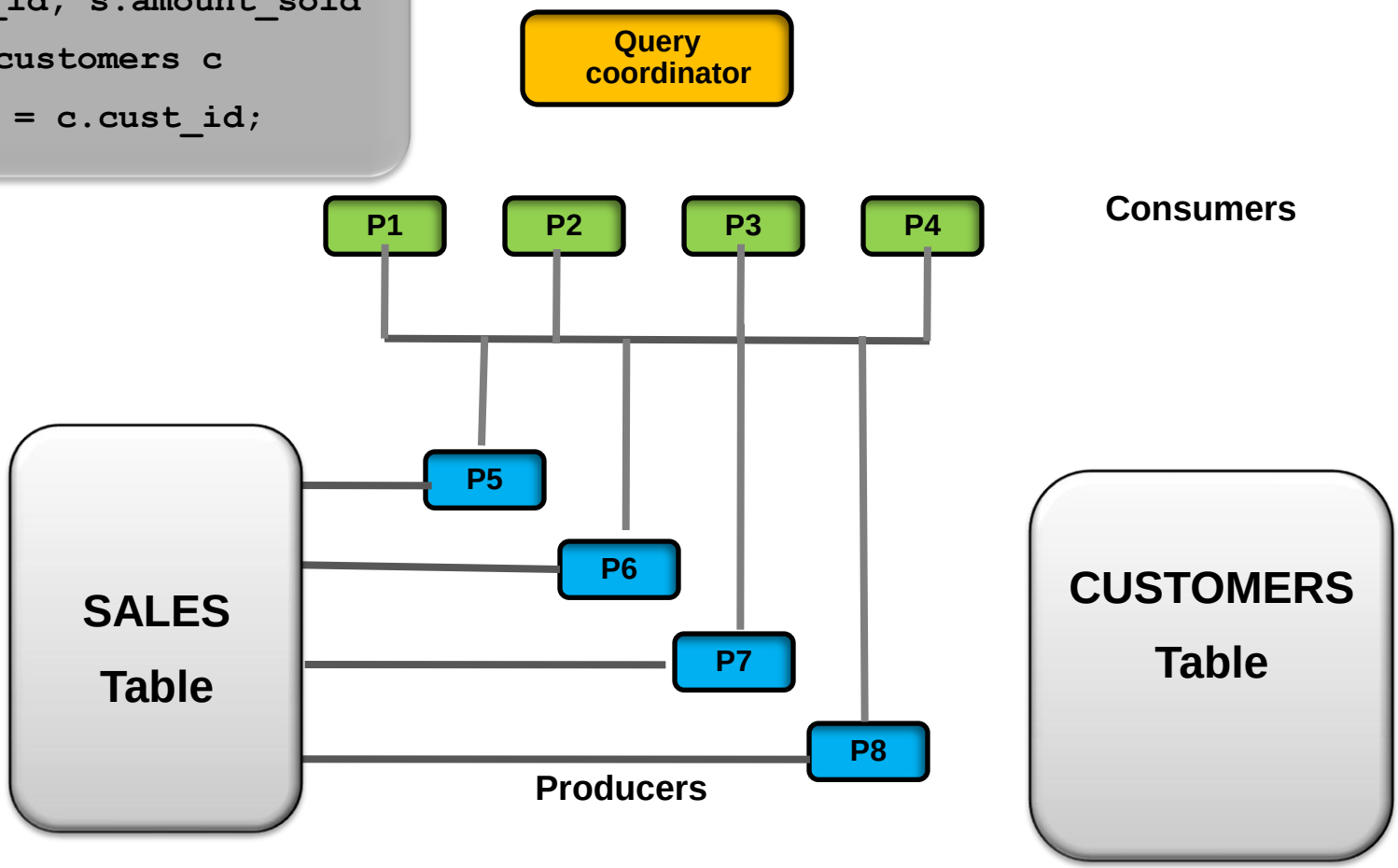
```
SELECT c.cust_last_name,
       s.time_id, s.amount_sold
FROM  sales s, customers c
WHERE s.cust_id = c.cust_id;
```




*Hash join always begins with a scan of the smaller table. In this case that's is the customer table. The 4 producers scan the customer table and send the resulting rows to the consumers*

# How Parallel Execution works

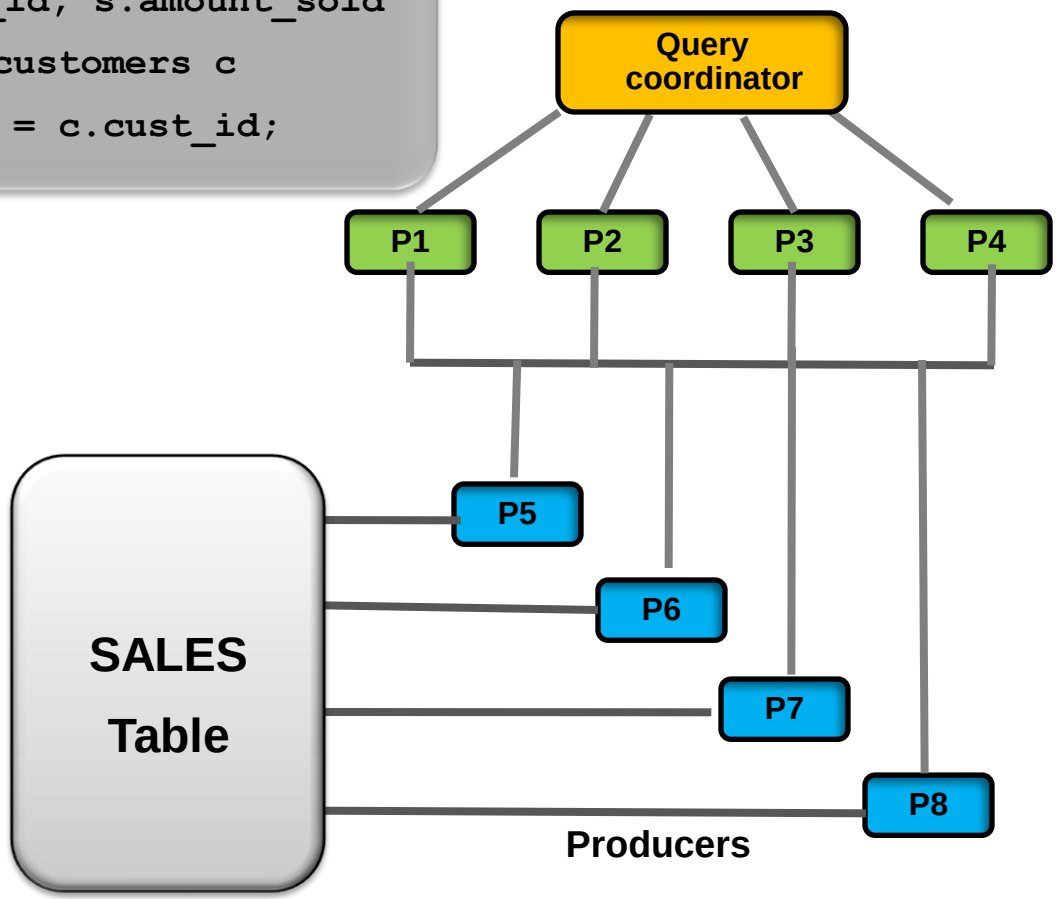
```
SELECT c.cust_last_name,
       s.time_id, s.amount_sold
FROM  sales s, customers c
WHERE s.cust_id = c.cust_id;
```




*Once the 4 producers finish scanning the customer table, they start to scan the Sales table and send the resulting rows to the consumers*

# How Parallel Execution works

```
SELECT c.cust_last_name,
       s.time_id, s.amount_sold
FROM  sales s, customers c
WHERE s.cust_id = c.cust_id;
```




**Consumers**

***Once the consumers receive the rows from the sales table they begin to do the join. Once completed they return the results to the QC***

# Monitoring Parallel Execution

```
SELECT  c.cust_last_name, s.time_id, s.amount_sold
FROM sales s, customers c
WHERE s.cust_id = c.cust_id;
```

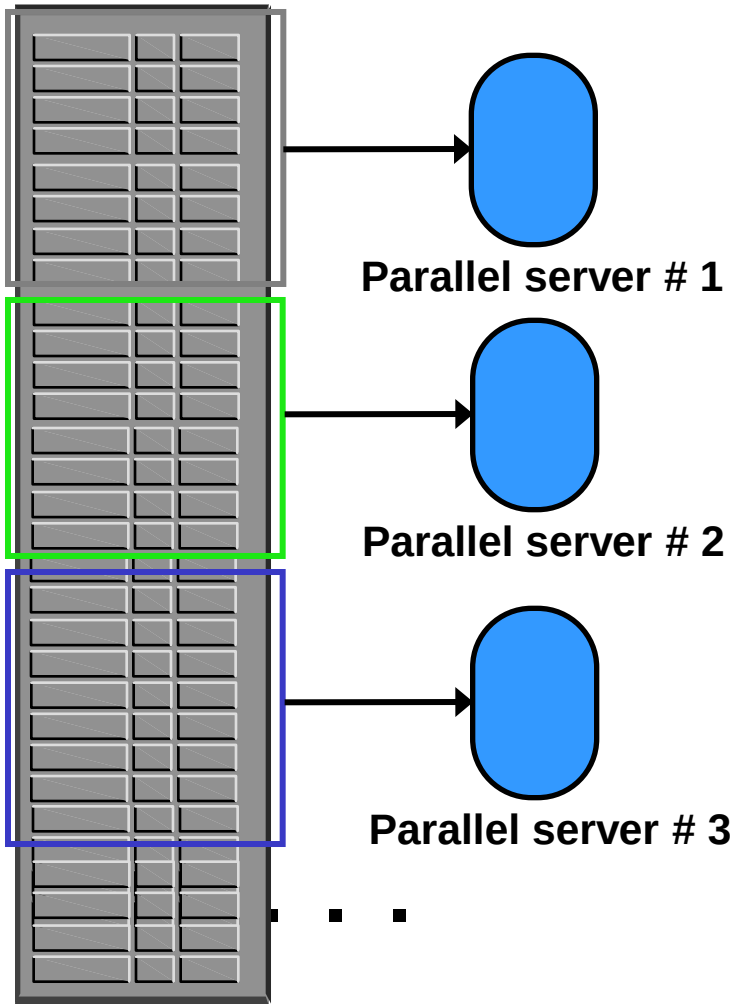
Consumers

Query Coordinator

| | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
|---|---|---|---|---|---|---|
| 0 | SELECT STATEMENT | | | | 311 (100) | |
| 1 | PX COORDINATOR | | | | | |
| 2 | PX SEND QC (RANDOM) | :TQ10002 | 1049K | 31M | 311 (2) | 00:00:04 |
| * 3 | HASH JOIN BUFFERED | | 1049K | 31M | 311 (2) | 00:00:04 |
| 4 | PX RECEIVE | | 55500 | 704K | 112 (0) | 00:00:02 |
| 5 | PX SEND HASH | :TQ10000 | 55500 | 704K | 112 (0) | 00:00:02 |
| 6 | PX BLOCK ITERATOR | | 55500 | 704K | 112 (0) | 00:00:02 |
| * 7 | TABLE ACCESS FULL | CUSTOMERS | 55500 | 704K | 112 (0) | 00:00:02 |
| 8 | PX RECEIVE | | 1049K | 18M | 196 (2) | 00:00:03 |
| 9 | PX SEND HASH | :TQ10001 | 1049K | 18M | 196 (2) | 00:00:03 |
| 10 | PX BLOCK ITERATOR | | 1049K | 18M | 196 (2) | 00:00:03 |
| * 11 | TABLE ACCESS FULL | SALES | 1049K | 18M | 196 (2) | 00:00:03 |

Producers

# Oracle Parallel Query: Scanning a Table




- Data is divided into Granules
  - Block range or partition
- Each Parallel Server is assigned one or more Granules
- No two Parallel Servers ever contend for the same Granule
- Granules are assigned so that the load is balanced across all Parallel Servers
- Dynamic Granules chosen by the optimizer
  - Granule decision is visible in execution plan

# Identifying Granules of Parallelism during Scans in the Plan

```
 Id  | Operation                | Name     | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |    TQ  |IN-OUT| PQ Distrib
-----------------------------------------------------------------------------------------------------------------------
   0 | SELECT STATEMENT         |          |    17 |   153 |   565 (100)| 00:00:07 |       |       |        |      |
   1 |  PX COORDINATOR          |          |       |       |            |          |       |       |        |      |
   2 |   PX SEND QC (RANDOM)    | :TQ10001 |    17 |   153 |   565 (100)| 00:00:07 |       |       |  Q1,01 | P->S | QC (RAND)
   3 |    HASH GROUP BY         |          |    17 |   153 |   565 (100)| 00:00:07 |       |       |  Q1,01 | PCWP |
   4 |     PX RECEIVE           |          |    17 |   153 |   565 (100)| 00:00:07 |       |       |  Q1,01 | PCWP |
   5 |      PX SEND HASH        | :TQ10000 |    17 |   153 |   565 (100)| 00:00:07 |       |       |  Q1,00 | P->P | HASH
   6 |       HASH GROUP BY      |          |    17 |   153 |   565 (100)| 00:00:07 |       |       |  Q1,00 | PCWP |
   7 |        PX BLOCK ITERATOR |          |   10M |    85M|    60  (97)| 00:00:01 |     1 |    16 |  Q1,00 | PCWC |
*  8 |         TABLE ACCESS FULL| SALES    |   10M |    85M|    60  (97)| 00:00:01 |     1 |    16 |  Q1,00 | PCWP |
-----------------------------------------------------------------------------------------------------------------------

redicate Information (identified by operation id):
---------------------------------------------------

  8 - filter("CUST_ID"<=22810 AND "CUST_ID">=22300)
```

```
 Id  | Operation                          | Name       | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |    TQ  |IN-OUT| PQ Distrib
-----------------------------------------------------------------------------------------------------------------------------------
   0 | SELECT STATEMENT                   |            |    17 |   153 |     2  (50)| 00:00:01 |       |       |        |      |
   1 |  PX COORDINATOR                    |            |       |       |            |          |       |       |        |      |
   2 |   PX SEND QC (RANDOM)              | :TQ10001   |    17 |   153 |     2  (50)| 00:00:01 |       |       |  Q1,01 | P->S | QC (RAND)
   3 |    HASH GROUP BY                   |            |    17 |   153 |     2  (50)| 00:00:01 |       |       |  Q1,01 | PCWP |
   4 |     PX RECEIVE                     |            |    26 |   234 |     1   (0)| 00:00:01 |       |       |  Q1,01 | PCWP |
   5 |      PX SEND HASH                  | :TQ10000   |    26 |   234 |     1   (0)| 00:00:01 |       |       |  Q1,00 | P->P | HASH
   6 |       PX PARTITION RANGE ALL       |            |    26 |   234 |     1   (0)| 00:00:01 |     1 |    16 |  Q1,00 | PCWC |
   7 |        TABLE ACCESS BY LOCAL INDEX ROWID| SALES      |    26 |   234 |     1   (0)| 00:00:01 |     1 |    16 |  Q1,00 | PCWP |
*  8 |         INDEX RANGE SCAN           | SALES_CUST |    26 |       |     0   (0)| 00:00:01 |     1 |    16 |  Q1,00 | PCWP |
-----------------------------------------------------------------------------------------------------------------------------------

redicate Information (identified by operation id):
---------------------------------------------------

  8 - access("CUST_ID">=22300 AND "CUST_ID"<=22810)
```

# Best Practices for using Parallel Execution

## Current Issues

- Difficult to determine ideal DOP for each table without manual tuning
- One DOP does not fit all queries touching an object
- Not enough PX server processes can result in statement running serial
- Too many PX server processes can thrash the system
- Only uses IO resources

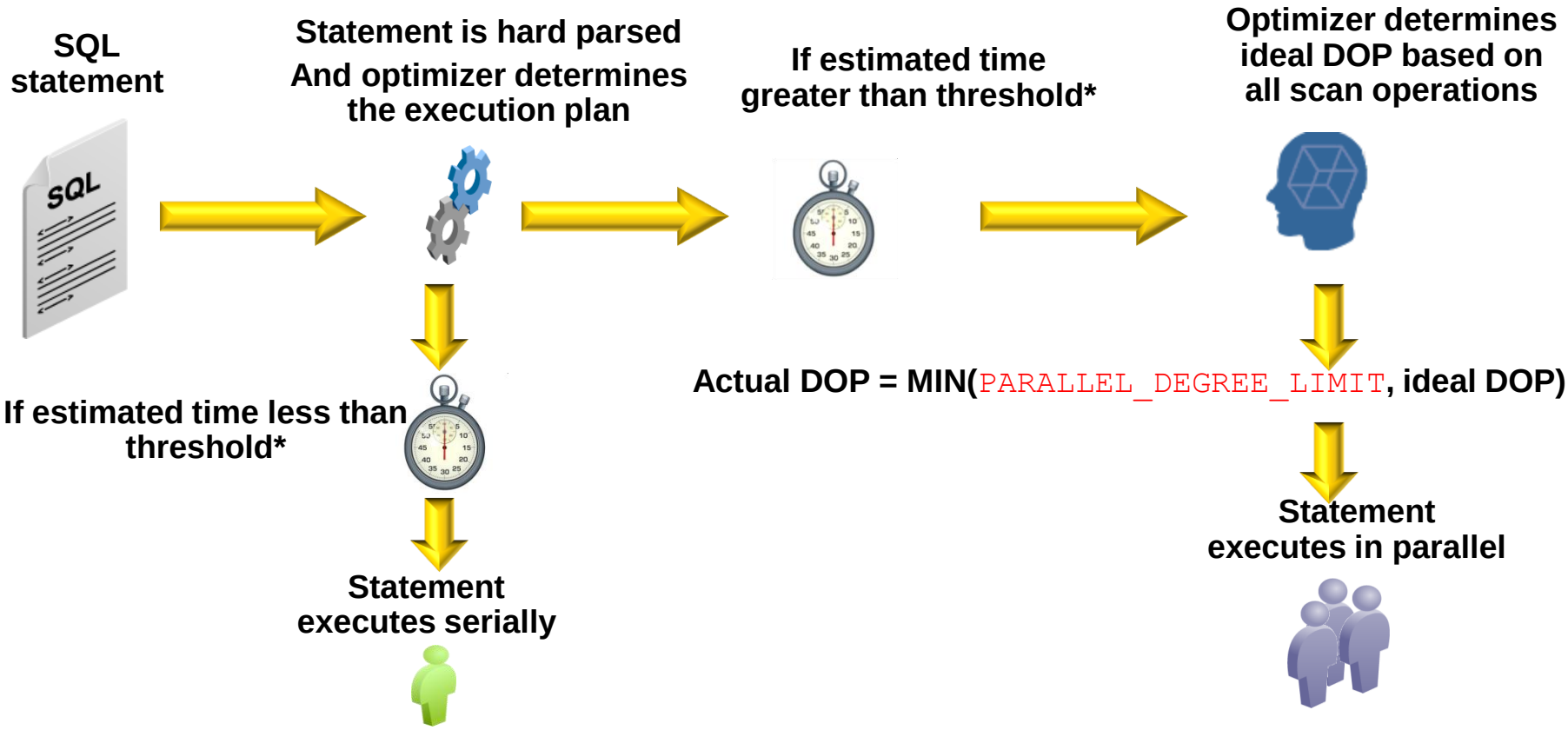
## Solution

- Oracle automatically decides if a statement
  - Executes in parallel or not and what DOP it will use
  - Can execute immediately or will be queued
  - Will take advantage of aggregated cluster memory or not

# Auto Degree of Parallelism

**Enhancement addressing:**

- Difficult to determine ideal DOP for each table without manual tuning
- One DOP does not fit all queries touching an object




* Threshold set in parallel_min_time_threshold (default = 10s)

# Controlling Auto DOP (not queuing!)

- Controlled by three init.ora parameters:
  - PARALLEL_DEGREE_POLICY
    - Controls whether or not auto DOP will be used
    - Default is MANUAL which means no Auto DOP
    - Set to AUTO  or LIMITED to enable auto DOP
  - PARALLEL_MIN_TIME_THRESHOLD
    - Controls which statements are candidate for parallelism
    - Default is 10 seconds
  - PARALLEL_DEGREE_LIMIT
    - Controls maximum DOP per statement
    - Default setting is the literal value “CPU” meaning DEFAULT DOP

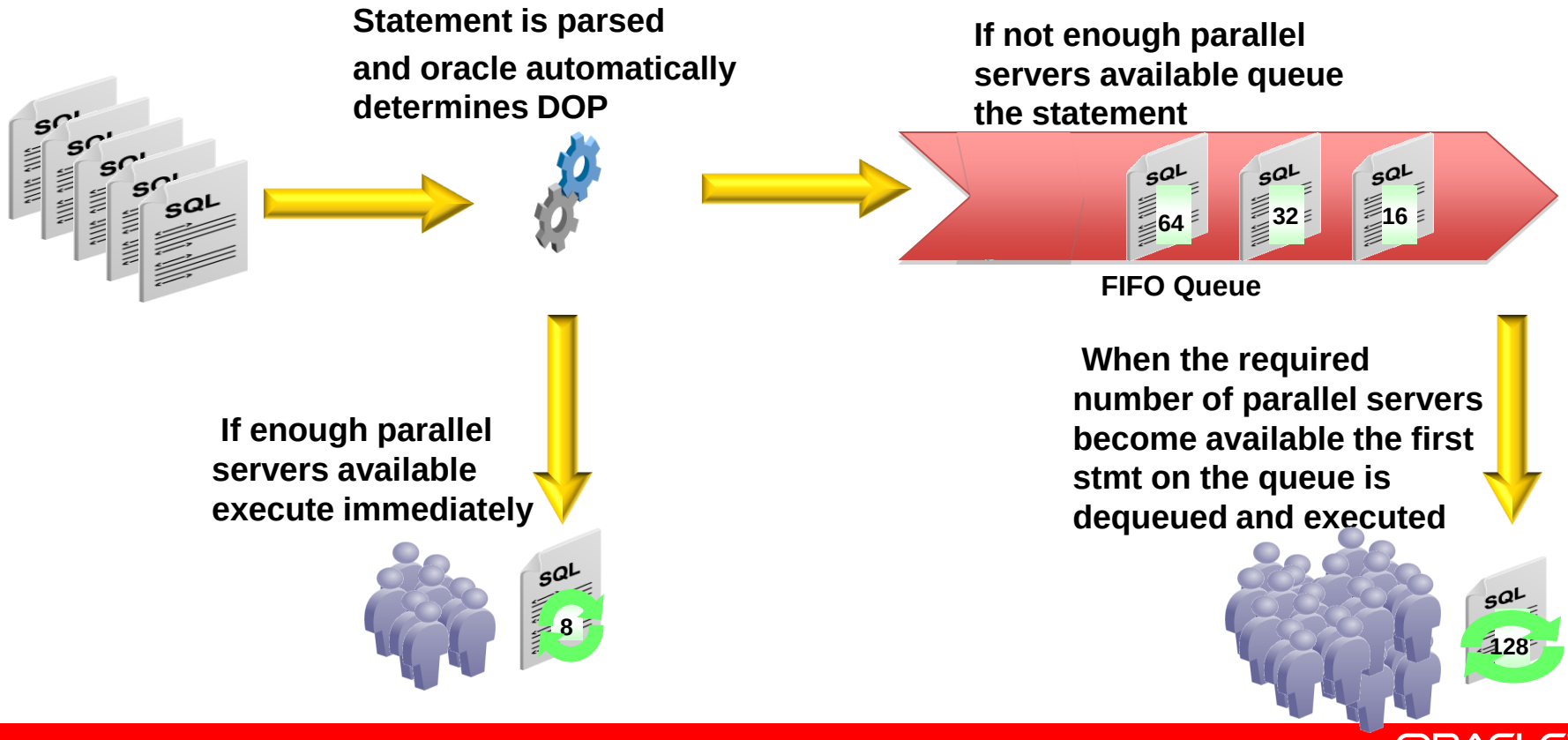
# What does this really mean?

## Working with Statement Queuing

# Parallel Statement Queuing

**Enhancement addressing:**

- Not enough PX server processes can result in statement running serial
- Too many PX server processes can thrash the system




**NOTE:** Parallel_Servers_Target new parameter controls number of active PX processes before statement queuing kicks in

# Parallel Statement Queuing

- Benefits:
  - Allows for higher DOPs per statement without thrashing the system
  - Allows a set of queries to run at roughly the same aggregate time by allowing the optimal DOP to be used all the time
- Potential Costs:
  - Adds delay to your execution time if your statement is queued making elapse times more unpredictable
- Goal:
  - Find the optimal queuing point based on desired concurrency

# Parallel Statement Queuing

## Calculating Minimal Concurrency based on processes

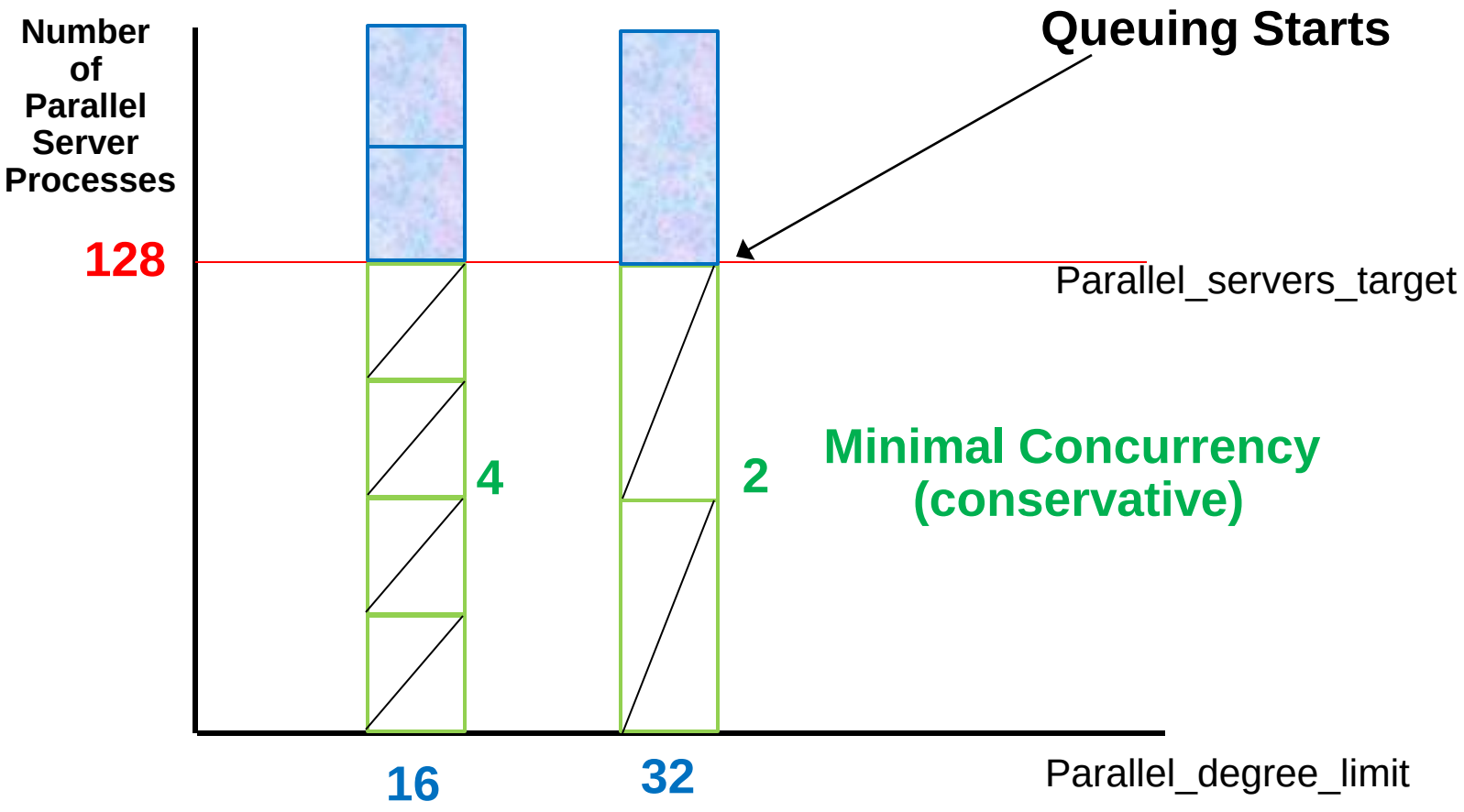
- Minimal concurrency is the minimal number of parallel statements than can run before queuing kicks in:

$$\text{minimal concurrency} = \frac{\text{Parallel\_servers\_target}}{\text{Parallel\_degree\_limit}} \times 0.5$$

- The conservative assumption is that you always have producers and consumers (and not PWJ all the time)

# Parallel Statement Queuing

## Real Minimal Concurrency



# Crucial for “all” 11g R2 PX features

| Parameter | Default Value | Description |
|---|---|---|
| PARALLEL_DEGREE_LIMIT | “CPU” | Max DOP that can be granted with Auto DOP |
| PARALLEL_DEGREE_POLICY | “MANUAL” | Specifies if Auto DOP, Queuing, & In-memory PE are enabled |
| PARALLEL_MIN_TIME_THRESHOLD | “AUTO” | Specifies min execution time a statement should have before AUTO DOP will kick in |
| PARALLEL_SERVERS_TARGET | 4*CPU_COUNT* PARALLEL_THREADS_PER_CPU * ACTIVE_INSTANCES | Specifies # of parallel processes allowed to run parallel stmts before queuing will be use |

# Enabling the 11g Features

INIT.ORA parameter PARALLEL_DEGREE_POLICY

Three possible modes:

- Manual
  - As before, DBA must manually specify all aspects of parallelism
  - No new features enabled
- Limited
  - Restricted AUTO DOP for queries with tables decorated with default PARALLEL
  - No Statement Queuing, No In-Memory Parallel Execution
- Auto
  - All qualifying statements subject to executing in parallel
  - DOP set on tables are ignored
  - Statements can be queued
  - In-memory PQ available

# Parameter Hierarchy

1. Parallel_degree_policy = Manual
   a) None of the parameters have any impact

PX Features:
- NONE

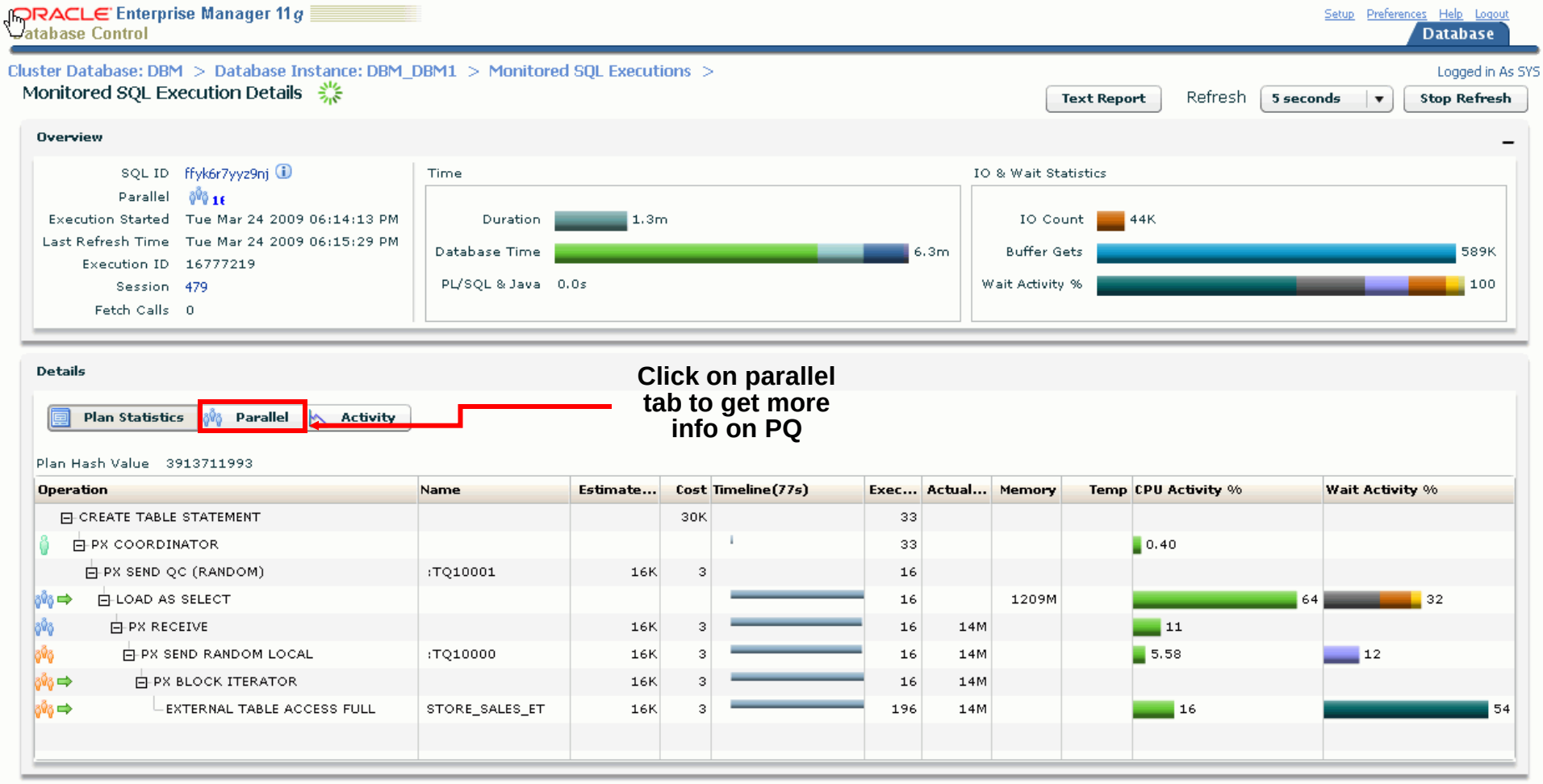
2. Parallel_degree_policy = Limited
   a) Parallel_min_time_threshold = 10s
   b) Parallel_degree_limit = CPU

PX Features:
- Auto DOP

3. Parallel_degree_policy = Auto
   a) Parallel_min_time_threshold = 10s
   b) Parallel_degree_limit = CPU
   c) Parallel_servers_target = Set to Default DOP on Exadata

PX Features:
- Auto DOP
- Queuing
- In-Memory

# SQL Monitoring screens




**Click on parallel tab to get more info on PQ**

**The green arrow indicates which line in the execution plan is currently being worked on**

# SQL Monitoring Screens

Details

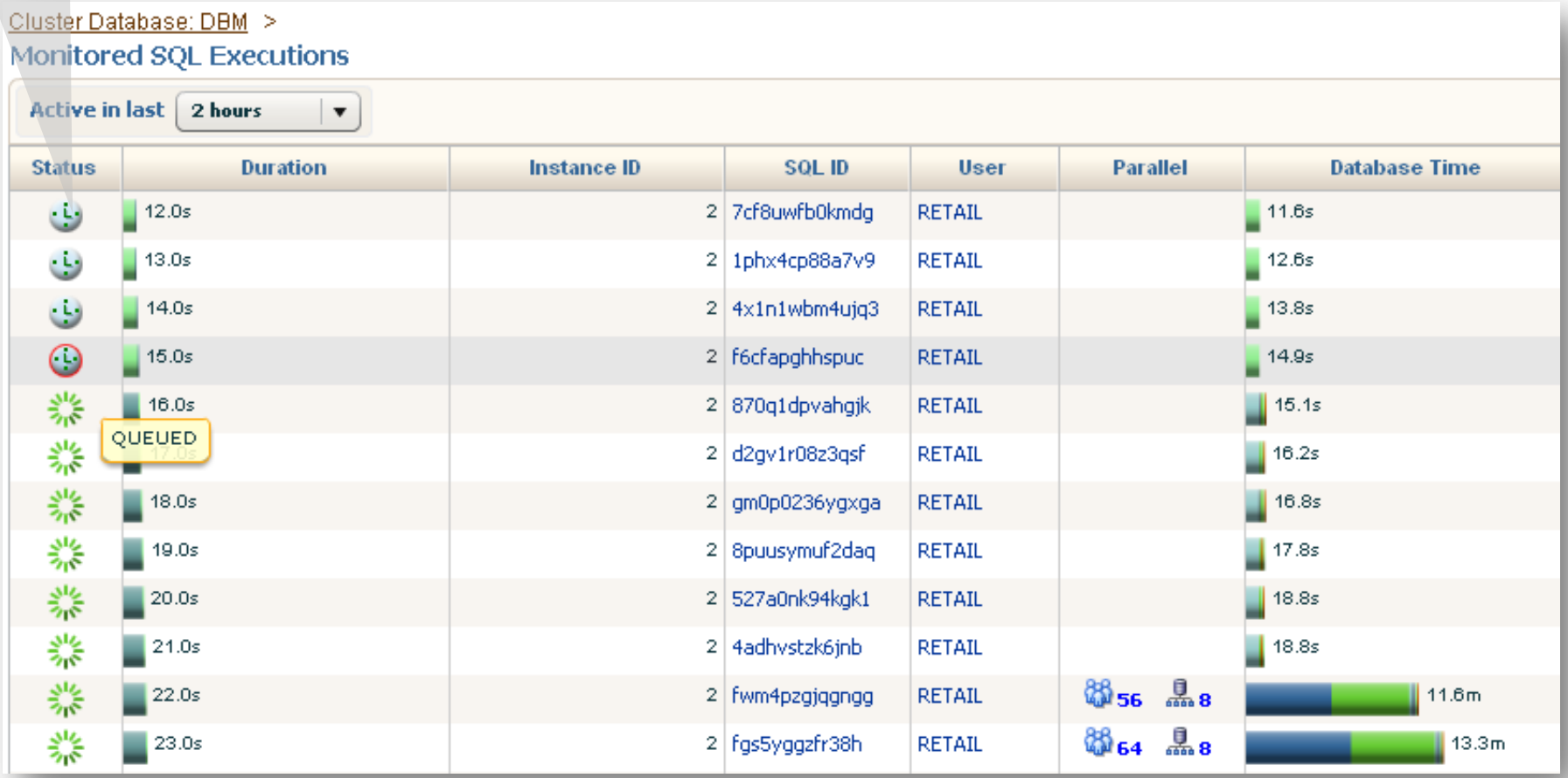
Plan Statistics | Parallel | Activity

**By clicking on the + tab you can get more detail about what each individual parallel server is doing. You want to check each slave is doing an equal amount of work**

| Parallel Server | Database Time | Wait Activity % | IO Count | Buffer Gets |
|---|---|---|---|---|
| All Parallel Servers | | | | |
| Parallel Coordinator | 8.4s | 2.04 | | 264K |
| Parallel Set 1 | | | | |
| Parallel Server 1 (p000) | 3.4s | | 36 | 8413 |
| Parallel Server 2 (p001) | 29.0s | 2.04 | 4768 | 19K |
| Parallel Server 3 (p002) | 3.4s | 4.08 | | 8069 |
| Parallel Server 4 (p003) | 6.6s | 4.08 | 1107 | 11K |
| Parallel Server 5 (p004) | 3.7s | 2.04 | 108 | 9342 |
| Parallel Server 6 (p005) | 3.5s | 4.08 | | 8016 |
| Parallel Server 7 (p006) | 6.4s | | 1062 | 11K |
| Parallel Server 8 (p007) | 5.0s | | 90 | 9359 |
| Parallel Server 9 (p008) | 13.3s | 2.04 | 3985 | 18K |
| Parallel Server 10 (p009) | 3.4s | 2.04 | 36 | 8296 |
| Parallel Server 11 (p010) | 16.6s | | 4793 | 19K |
| Parallel Server 12 (p011) | 3.5s | 2.04 | | 8069 |
| Parallel Server 13 (p012) | 6.5s | 4.08 | 1107 | 11K |
| Parallel Server 14 (p013) | 3.6s | | 108 | 9823 |
| Parallel Server 15 (p014) | 3.3s | 2.04 | | 8016 |
| Parallel Server 16 (p015) | 6.4s | 2.04 | 1062 | 10K |
| Parallel Set 2 | 1.2m | 67 | | 90K |

# Simple Example of Queuing

Queued stmts are indicated by the clock




| Status | Duration | Instance ID | SQL ID | User | Parallel | Database Time |
|---|---|---|---|---|---|---|
| | 12.0s | 2 | 7cf8uwfb0kmdg | RETAIL | | 11.6s |
| | 13.0s | 2 | 1phx4cp88a7v9 | RETAIL | | 12.6s |
| | 14.0s | 2 | 4x1n1wbm4ujq3 | RETAIL | | 13.8s |
| | 15.0s | 2 | f6cfapghhspuc | RETAIL | | 14.9s |
| | 16.0s | 2 | 870q1dpvahgjk | RETAIL | | 15.1s |
| | 17.0s | 2 | d2gv1r08z3qsf | RETAIL | | 16.2s |
| | 18.0s | 2 | gm0p0236ygxga | RETAIL | | 16.8s |
| | 19.0s | 2 | 8puusymuf2daq | RETAIL | | 17.8s |
| | 20.0s | 2 | 527a0nk94kgk1 | RETAIL | | 18.8s |
| | 21.0s | 2 | 4adhvstzk6jnb | RETAIL | | 18.8s |
| | 22.0s | 2 | fwm4pzgjqgngg | RETAIL | 56 8 | 11.6m |
| | 23.0s | 2 | fgs5yggzfr38h | RETAIL | 64 8 | 13.3m |

8 Statements run before queuing kicks in

# Preventing Extreme DOPs

## Setting a system wide parameter

- By setting parallel_degree_limit you CAP the maximum degree ANY statement can run with on the system
- Default setting is Default DOP which means no statement ever runs at a higher DOP than Default DOP
- Think of this as your safety net for when the magic fails and Auto DOP is reaching extreme levels of DOP

Note: EM will not show a downgrade for capped DOPs!

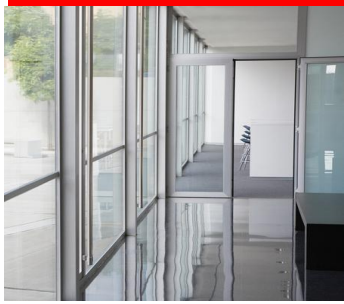
# In-Memory Parallel Execution

Efficient use of memory on clustered servers

**In-Memory Parallel Query in Database Tier**

- **Compress more data into available memory on cluster**
- **Intelligent algorithm**
  - Places table fragments in memory on different nodes
- **Reduces disk IO and speeds query execution**

# Agenda



- Parallel Execution
- Workload Management on a Data Warehouse
- Oracle Exadata Database machine

# MIXED WORKLOAD: what does it mean?

Diverse workload running on a Data Warehouse system concurrently.

Examples:

- Continuous data loads while end users are querying data
- OLTP like activities (both queries and trickle data loads) mixed in with more classic ad-hoc data intensive query patterns.

# Step 1: Understand the Workload

- Review the customer workload to find out:
  - Who is doing the work?
  - What types of work are done on the system?
  - When are certain types being done?
  - Where are performance problem areas?
  - What are the priorities, and do they change during a time window?
  - Are there priority conflicts?

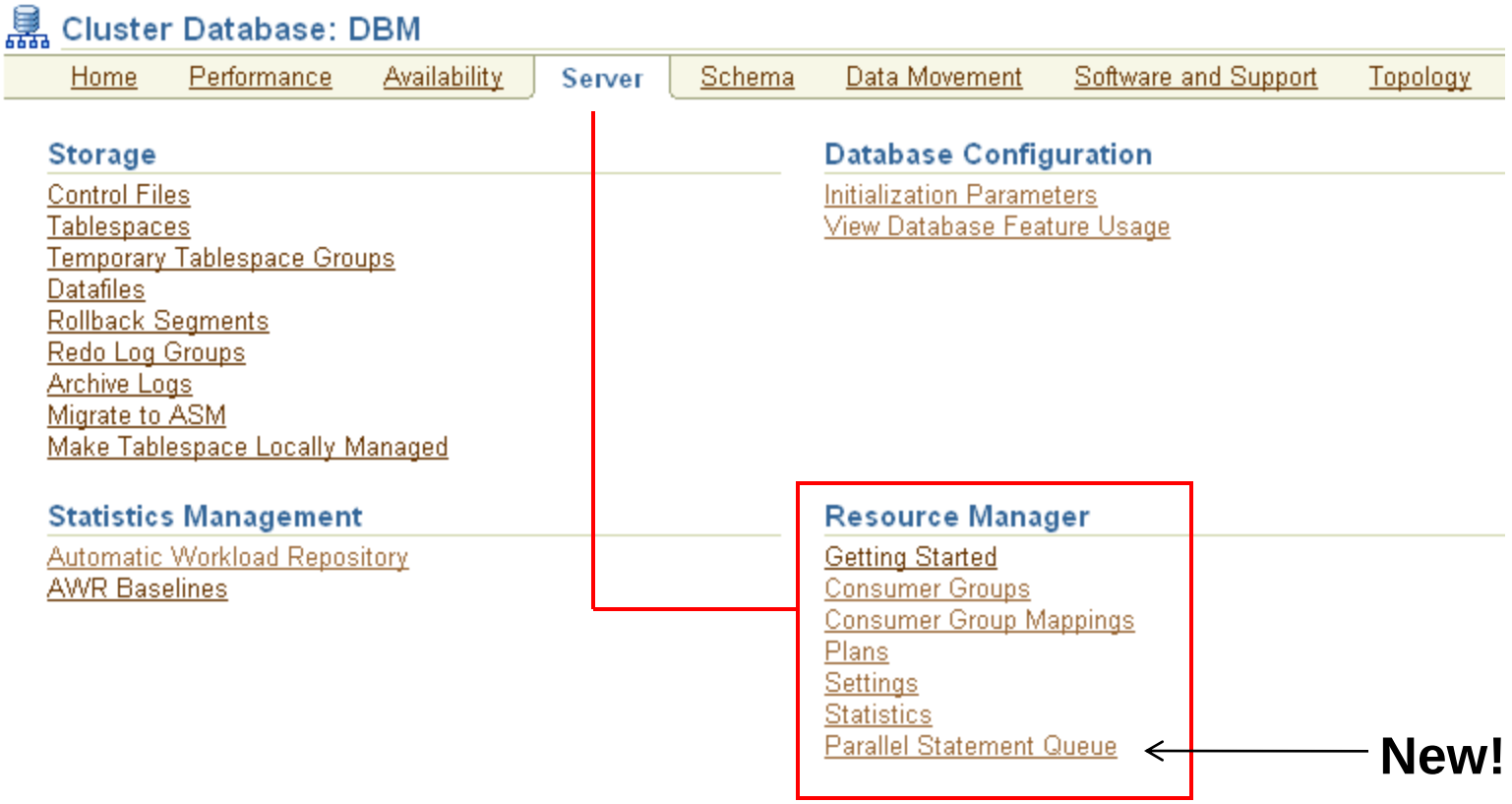
# Step 2: Map the Workload to the System

- Create the resource groups:
  - Map to users
  - Map to estimated execution time
  - Etc
- Create the required Resource Plans:
  - For example: Nighttime vs. daytime, online vs. offline
- Set the overall priorities
  - Which resource group gets most resources
  - Cap max utilizations
- Drill down into parallelism, queuing and session throttles

# Why use Resource Manager?

- Manage workloads contending for CPU
- Prevent excessive CPU load, destabilizing server
- Manage parallel query processes, queuing and concurrency
- Prevent runaway queries
- Manage workloads contending for I/O

# Resource Manager User Interface



# Working example of Workload management using DBRM

Step 1: Understand your workload

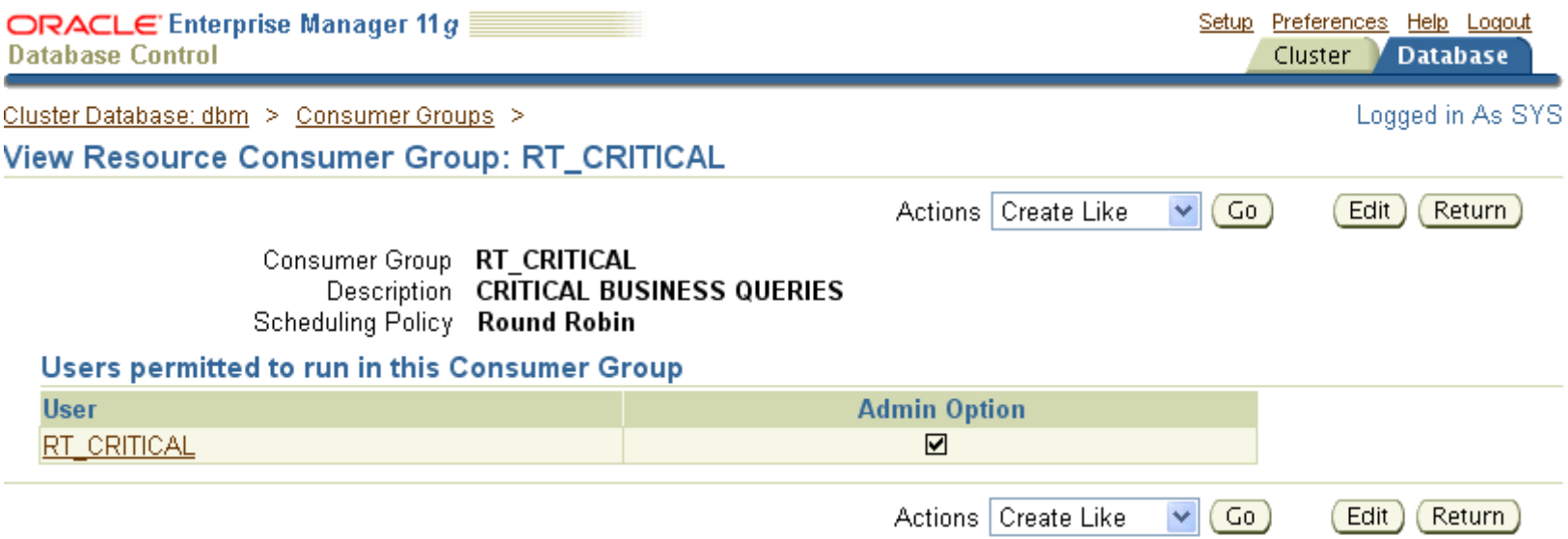
Working Example:

- User 1: RTL runs long running analytical queries.
- User 2: RT_CRITICAL is a “SUPER” business user run short critical queries various times of the day.

**GOAL:**

**Ensure the Critical queries will run in a consistent and timely manner even when the system is loaded with batch analytical queries.**

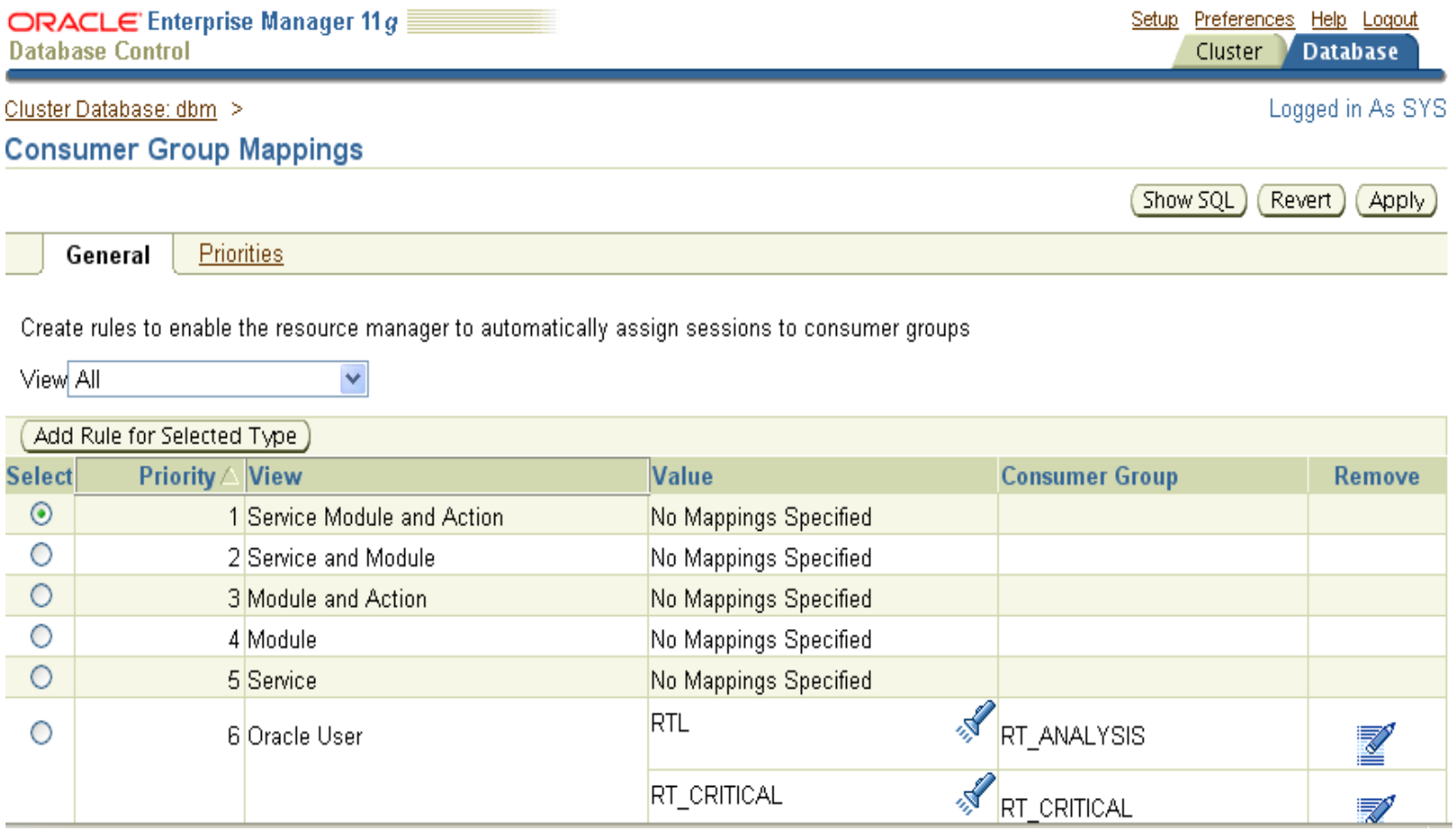
# DBRM – RESOURCE GROUP CRITICAL SHORT QUERIES STEP 1




**TWO CONSUMER GROUPS :**

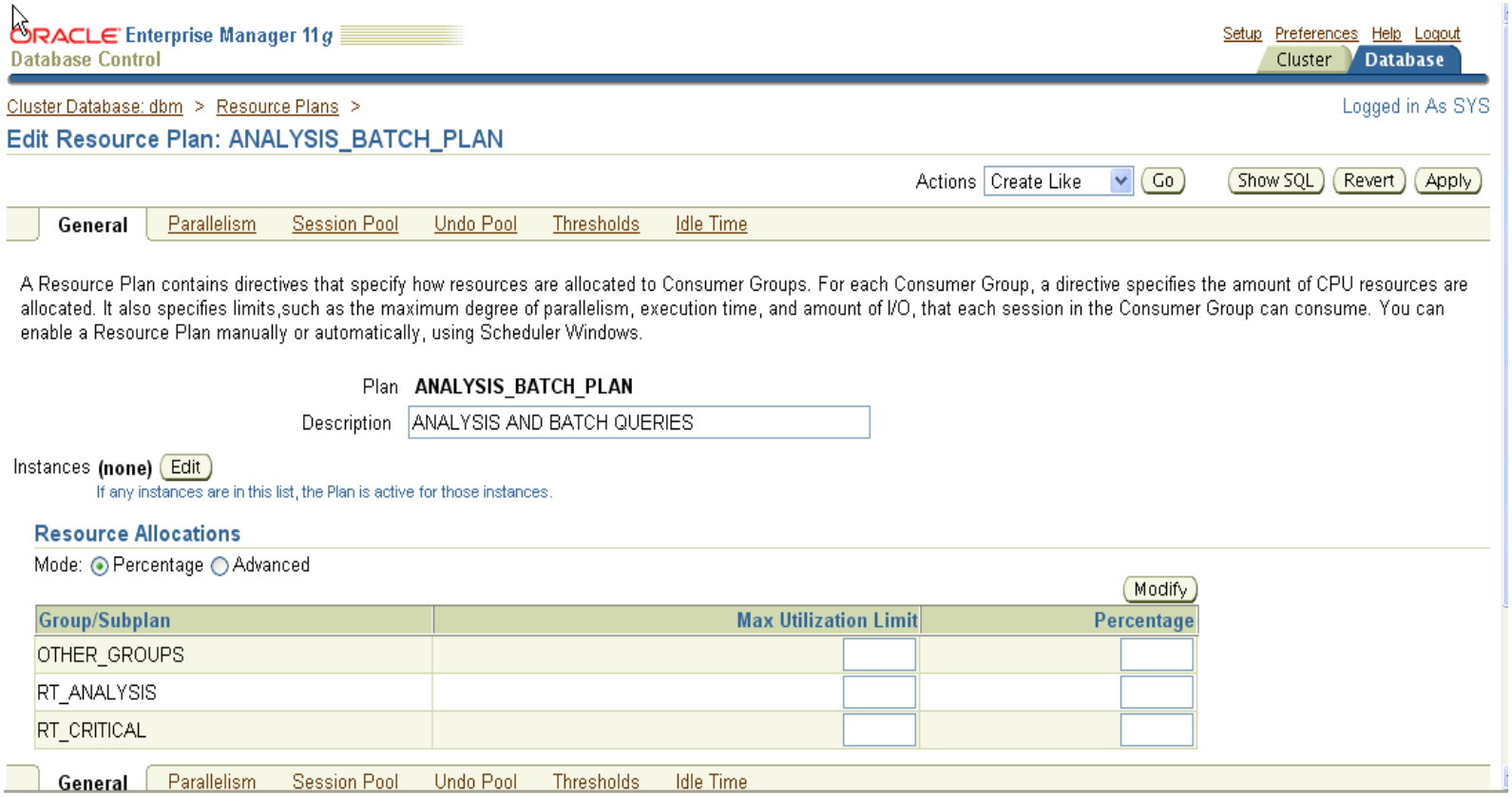
- **RT_CRITICAL**
- **RT_ANALYSIS**

# Consumer Group Mappings




- **Map RTL USER to consumer group RT_ANALYSIS**
- **Map RT_CRITICAL to consumer group RT_CRITICAL**

# Step 2 CREATE RESOURCE PLAN

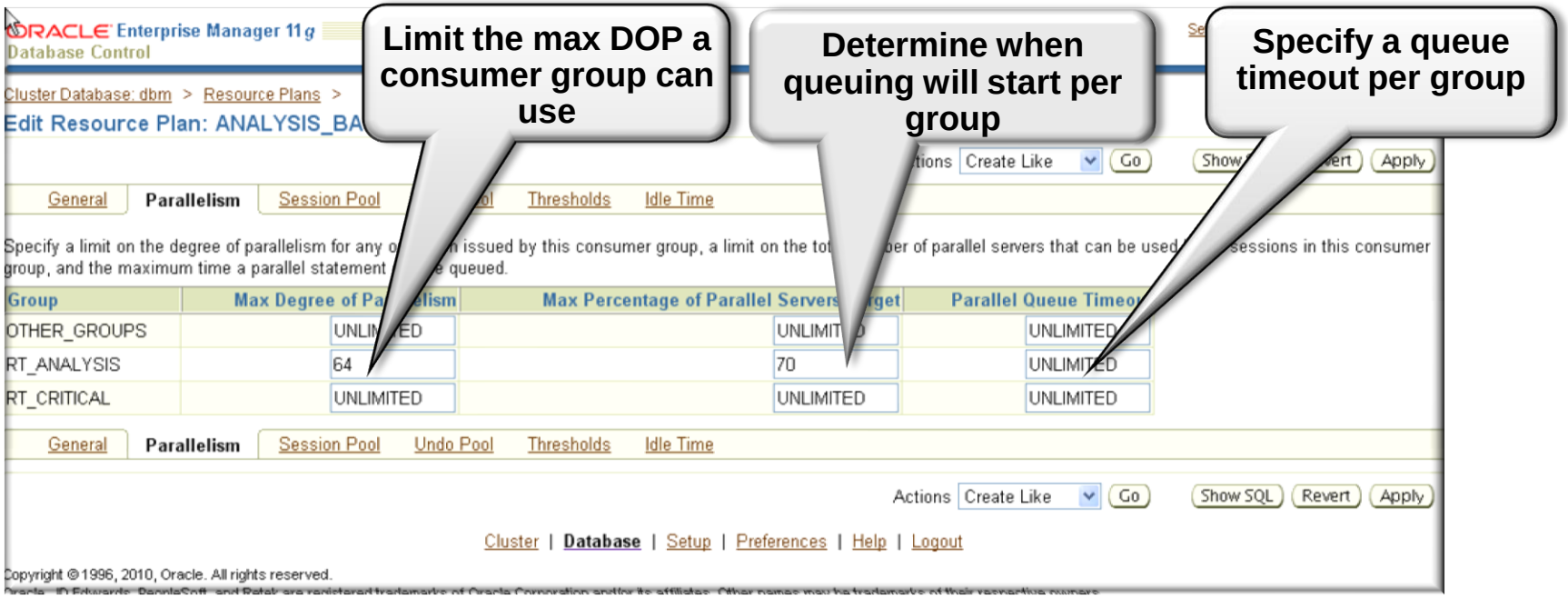



**Created two plans:**

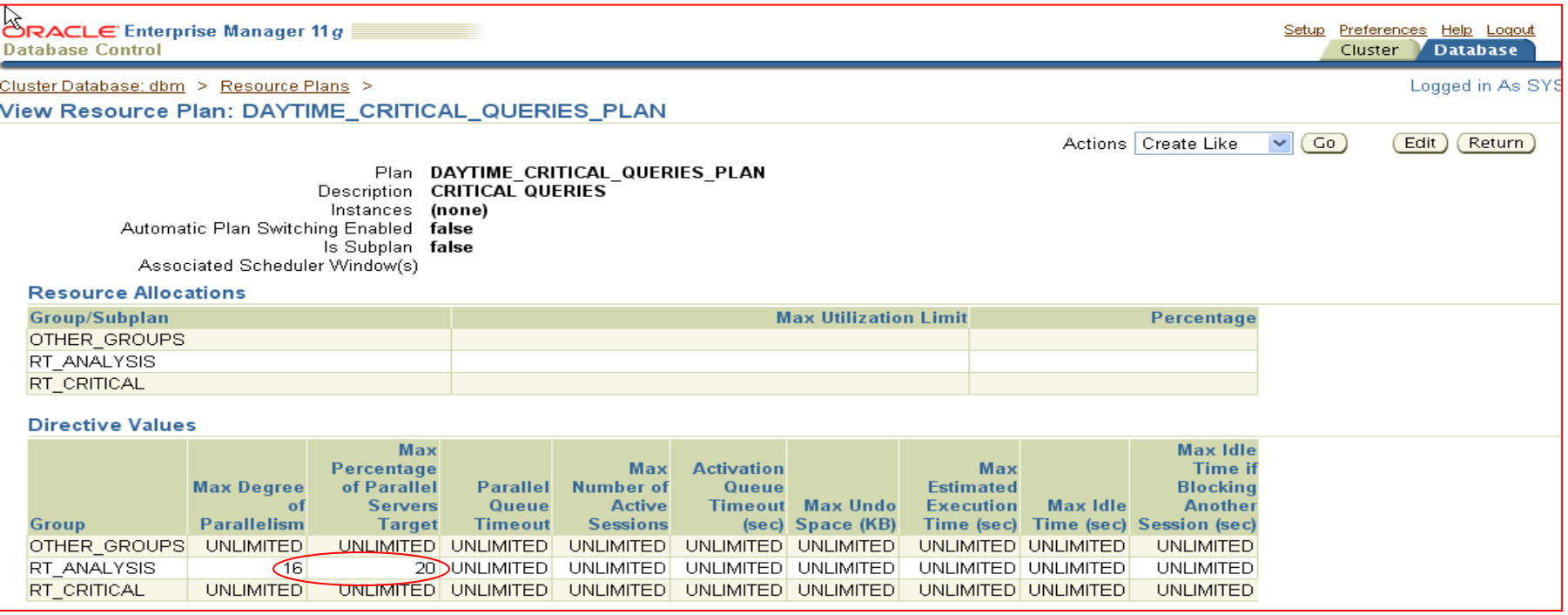
- **ANALYSIS_BATCH_PLAN**
- **Critical_BATCH_PLAN**

# Configure Parallel Execution

- **Manage parallel execution resources and priorities by consumer groups**



# Allocate resources DAYTIME_CRITICAL_QUERIES_PLAN



ORACLE Enterprise Manager 11g
Database Control

Setup Preferences Help Logout
Cluster Database

Cluster Database: dbm > Resource Plans >

Logged in As SYS

View Resource Plan: DAYTIME_CRITICAL_QUERIES_PLAN

Actions Create Like Go Edit Return

Plan DAYTIME_CRITICAL_QUERIES_PLAN
Description CRITICAL QUERIES
Instances (none)
Automatic Plan Switching Enabled false
Is Subplan false
Associated Scheduler Window(s)

Resource Allocations

| Group/Subplan | Max Utilization Limit | Percentage |
|---|---|---|
| OTHER_GROUPS | | |
| RT_ANALYSIS | | |
| RT_CRITICAL | | |

Directive Values

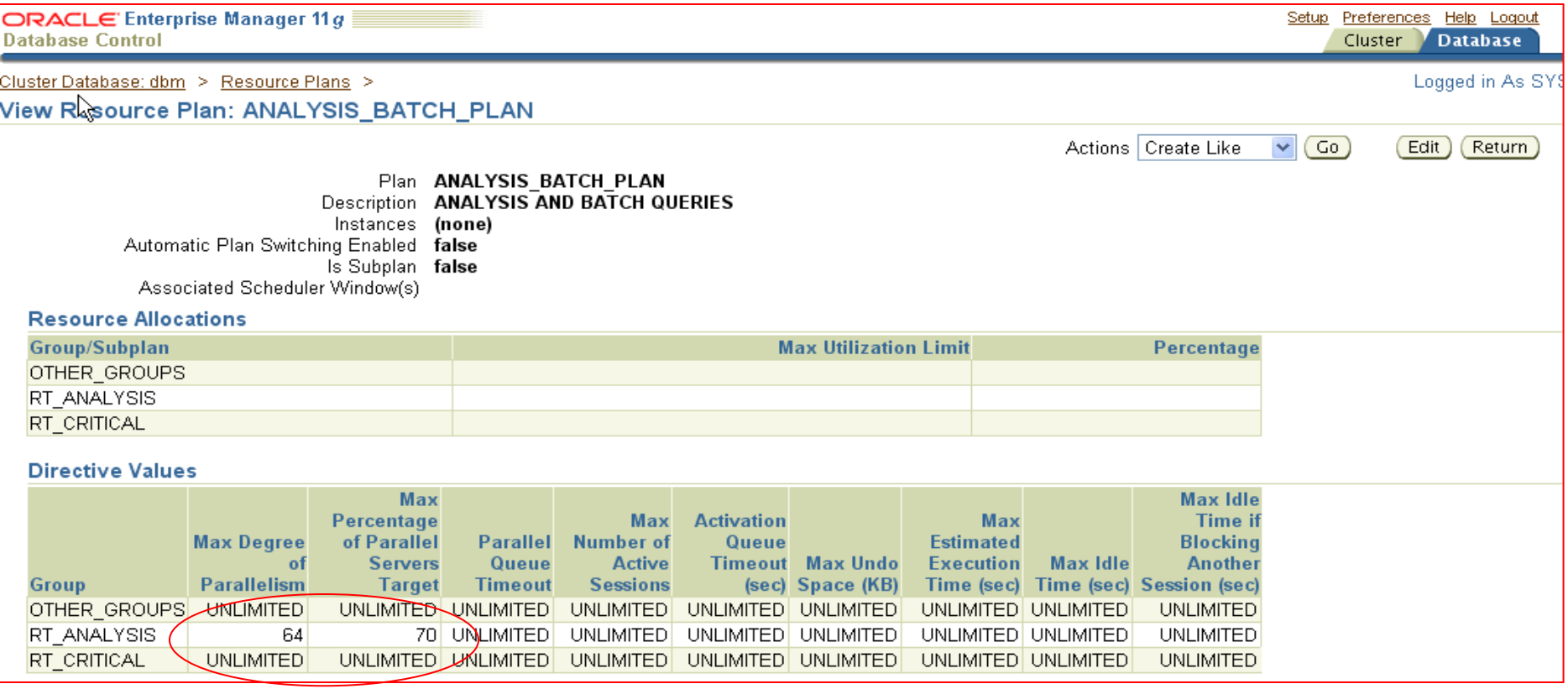
| Group | Max Degree of Parallelism | Max Percentage of Parallel Servers Target | Parallel Queue Timeout | Max Number of Active Sessions | Activation Queue Timeout (sec) | Max Undo Space (KB) | Max Estimated Execution Time (sec) | Max Idle Time (sec) | Max Idle Time if Blocking Another Session (sec) |
|---|---|---|---|---|---|---|---|---|---|
| OTHER_GROUPS | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED |
| RT_ANALYSIS | 16 | 20 | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED |
| RT_CRITICAL | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED | UNLIMITED |

**Statements issued by RTL_ANALYSIS:**

**DOP is capped at 16**

**Statements will queue once the number of Max parallel processes 204= 20/100* 1024( 1024).**

# Allocate resources




**RTL _ANALYSIS GROUP : MAX DOP capped at 64 and queuing will start once processes reach 70% of Parallel_server_Target**

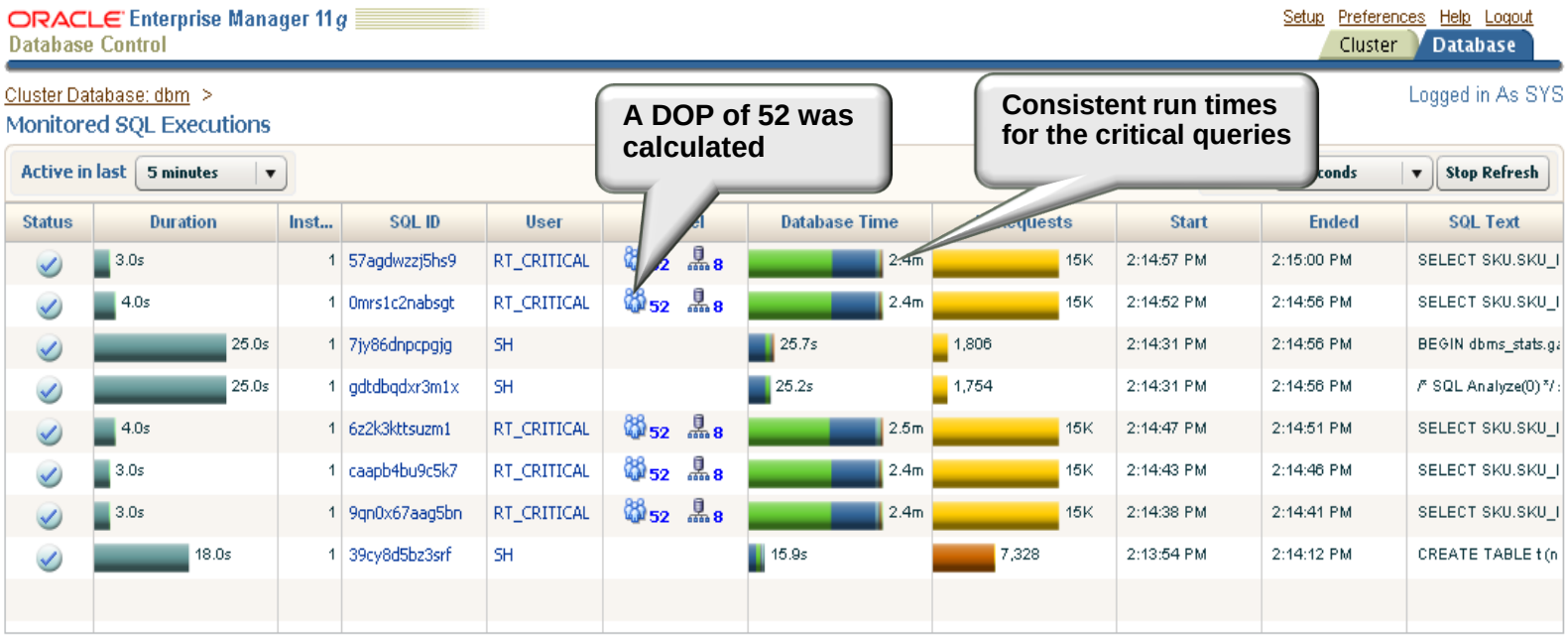
**UNLIMITED: MAX DOP is capped by Parallel_Degree_Limit**

# Example long-running query

## No load on the system and no resource mgmt activated

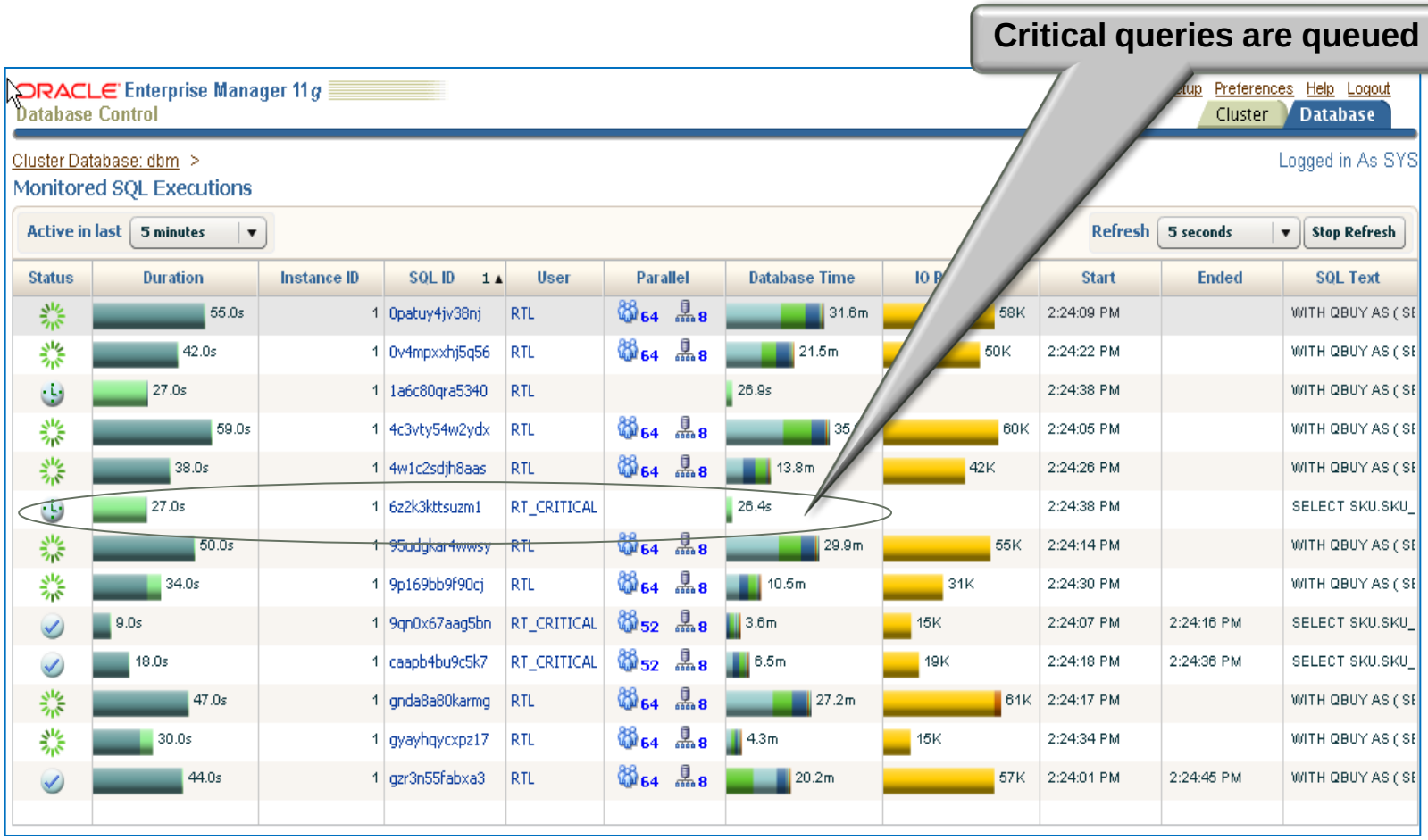

- Automatic DOP of 128 chosen by the system
  - Does not ensure requested concurrency

# Critical Queries with no load on the system and no plan activated




- NO RESOURCE PLAN ACTIVATED
- NO LOAD ON THE SYSTEM
- DOP OF 52 CALCULATED BY THE SYSTEM

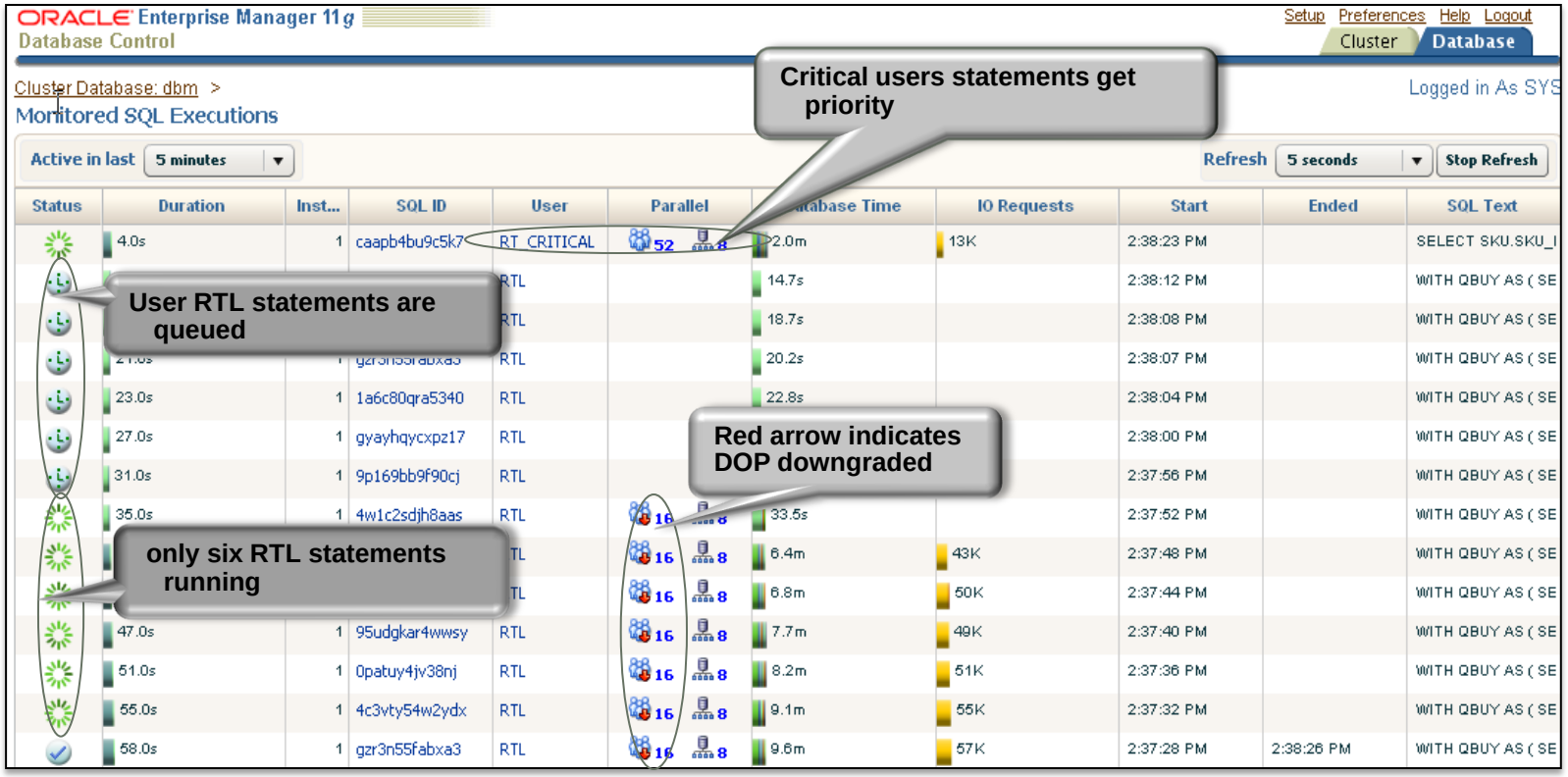
# Scenario we want to avoid !!




**When the System is loaded with long running analytical statement**

**CRITICAL QUERIES are queued and have inconsistent elapsed times**

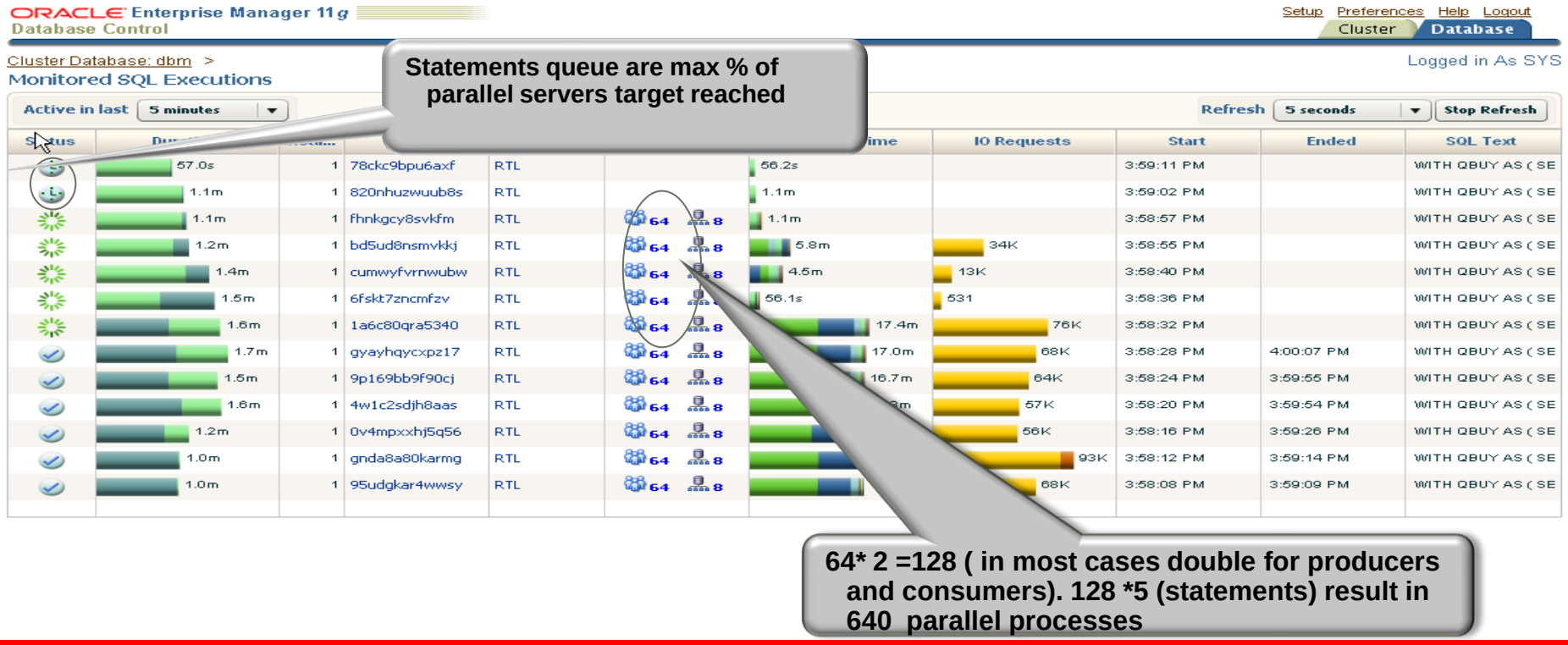
# CRITICAL_QUERIES_DAY_PLAN ACTIVATED



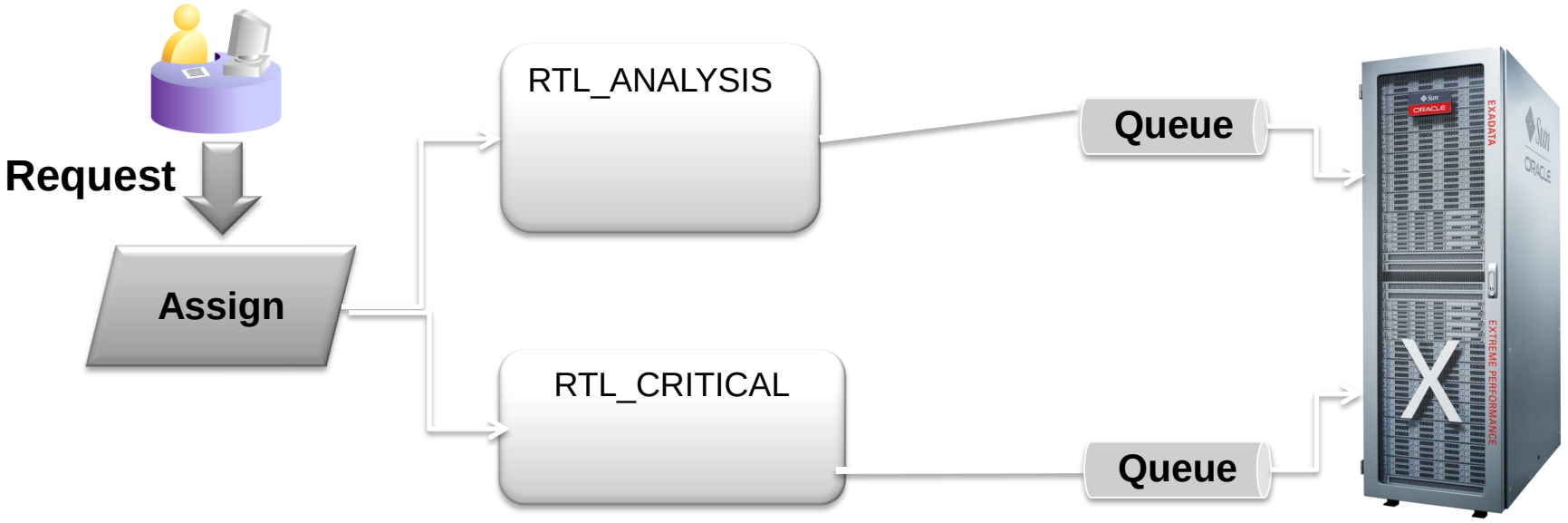

- Why are statements queued when only six statements are running? 6*16 = 96
- RTL _ANLAYSIS group is allowed to up to 20%of 1024 = 204
- ANSWER: STATEMENT REQUIRES TWO SETS OF PX PROCESS FOR PRODUCERS & CONSUMERS
- 32*6 = 192 .. the seventh query would take it over total allowed .. 192+32 =224

# Analysis Batch Plan

- RT_Analaysis  group : Max Percentage of parallel servers target =70% = 70*1024/100 =719

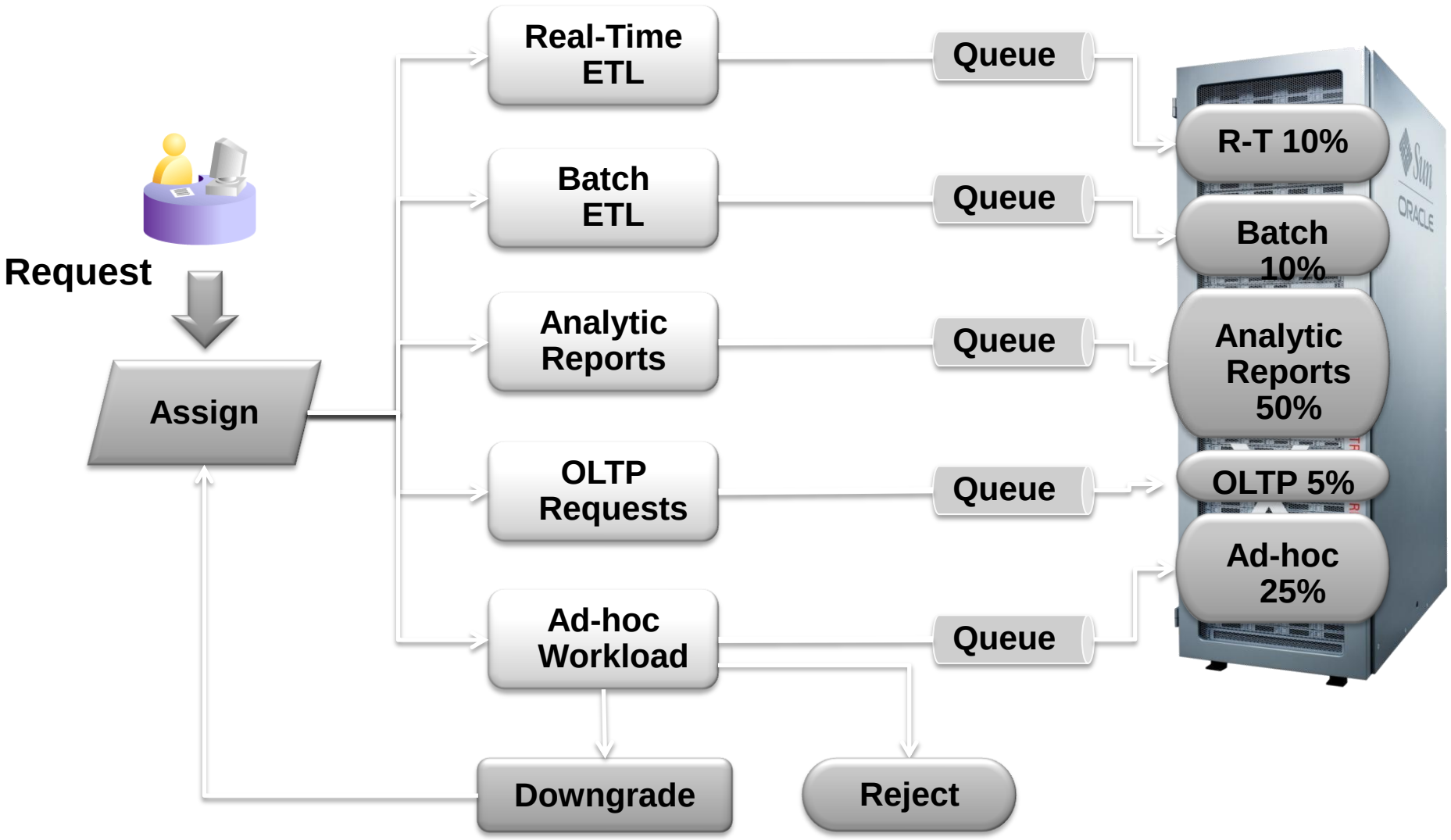


# Resource Manager - Statement Queuing




- Queuing is embedded with DBRM
- One queue per consumer group

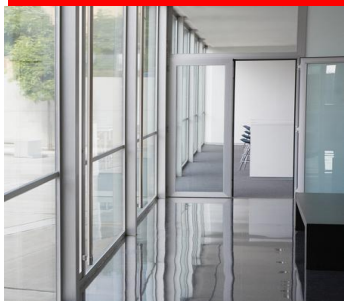
# Workload Management



# Step 3: Run and Adjust the Workload

- Run a workload for a period of time and look at the results
- DBRM Adjust:
  - Overall priorities
  - Scheduling of switches in plans
  - Queuing
- System Adjust:
  - How many PX statements
  - PX Queuing levels vs. Utilization levels (should we queue less?)

# Agenda



- Parallel Execution
- Workload Management on a Data Warehouse
- Oracle Exadata Database machine

# Oracle Exadata Database Machine Family

## Oracle Exadata Database Machine X2-8


**Oracle Database Server Grid**

- 2 8-processor Database Servers
  - 128 CPU Cores
  - 2 TB Memory
  - Oracle Linux or Solaris 11 Express

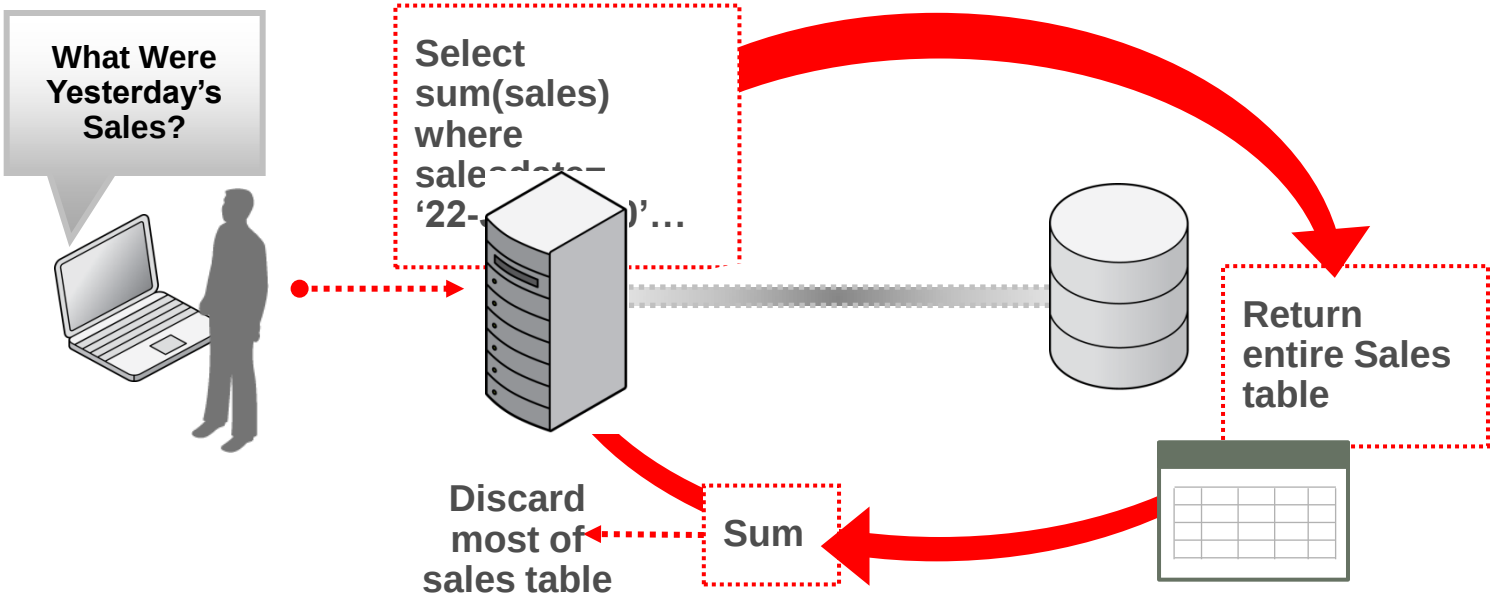
**Exadata Storage Server Grid**

- 14 Storage Servers
  - 5 TB Smart Flash Cache
  - 336 TB Disk Storage

**Unified Server/Storage Network**

- 40 Gb/sec Infiniband Links

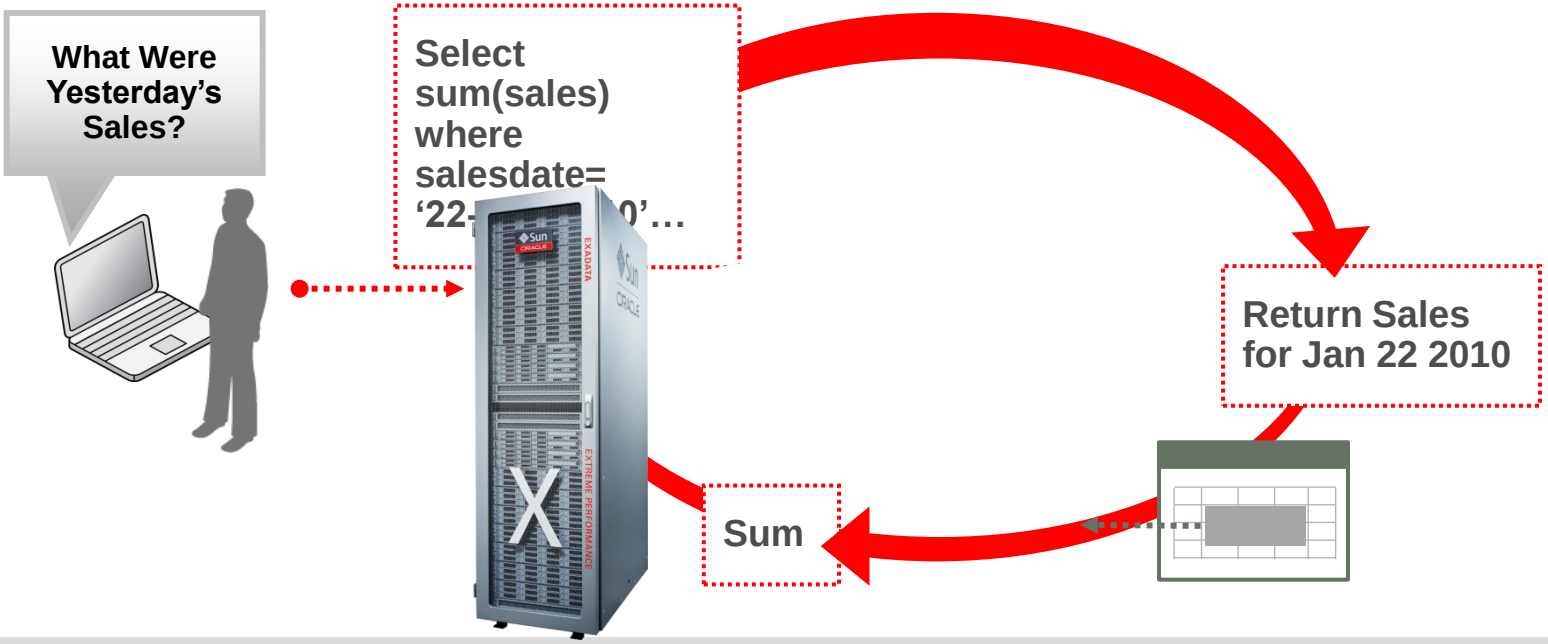
# Traditional Query Problem




- Data is pushed to database server for processing
- I/O rates are limited by speed and number of disk drives
- Network bandwidth is strained, limiting performance and concurrency

# Exadata Smart Scan

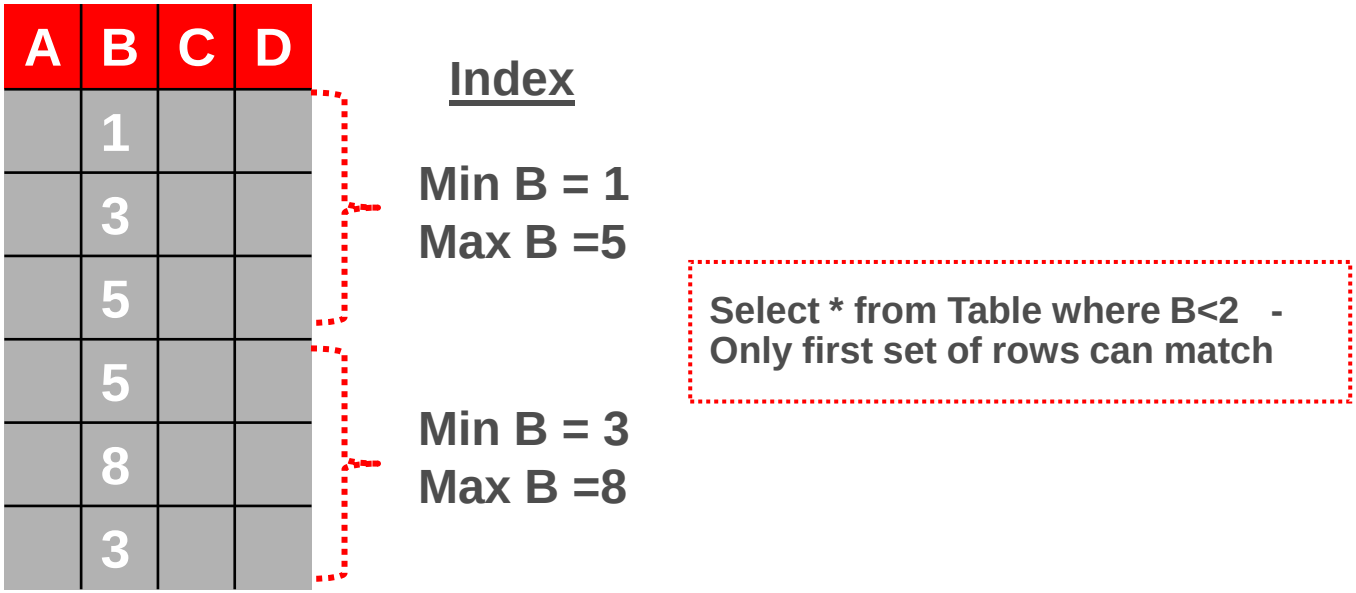
## Improve Query Performance by 10x or More




- Off-load data intensive processing to Exadata Storage Server
- Exadata Storage Server only returns relevant rows and columns
- Wide Infiniband connections eliminate network bottlenecks

# Exadata Storage Index

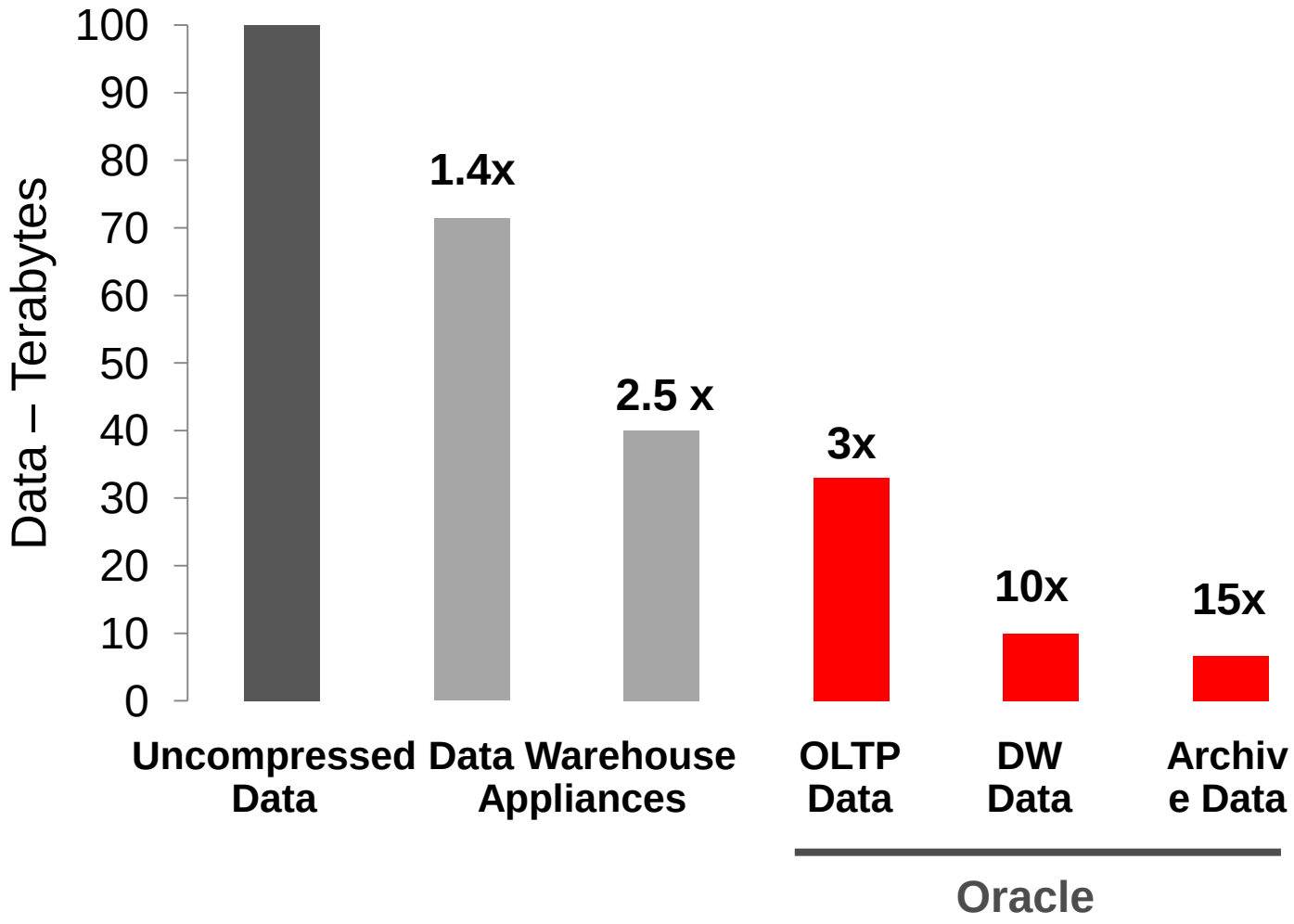
## Transparent I/O Elimination with No Overhead




- Maintain summary information about table data in memory
- Eliminate disk I/Os if MIN / MAX never match “where” clause
- Completely automatic and transparent

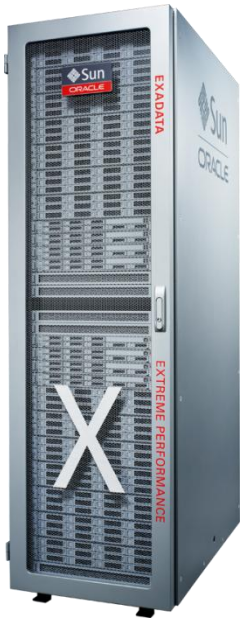
# Exadata Hybrid Columnar Compression

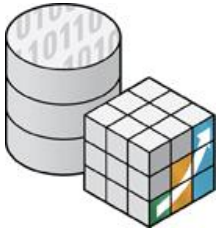
## Reduce Disk Space Requirements



# Built-in Analytics

## Secure, Scalable Platform for Advanced Analytics



**Oracle OLAP**

Analyze and summarize

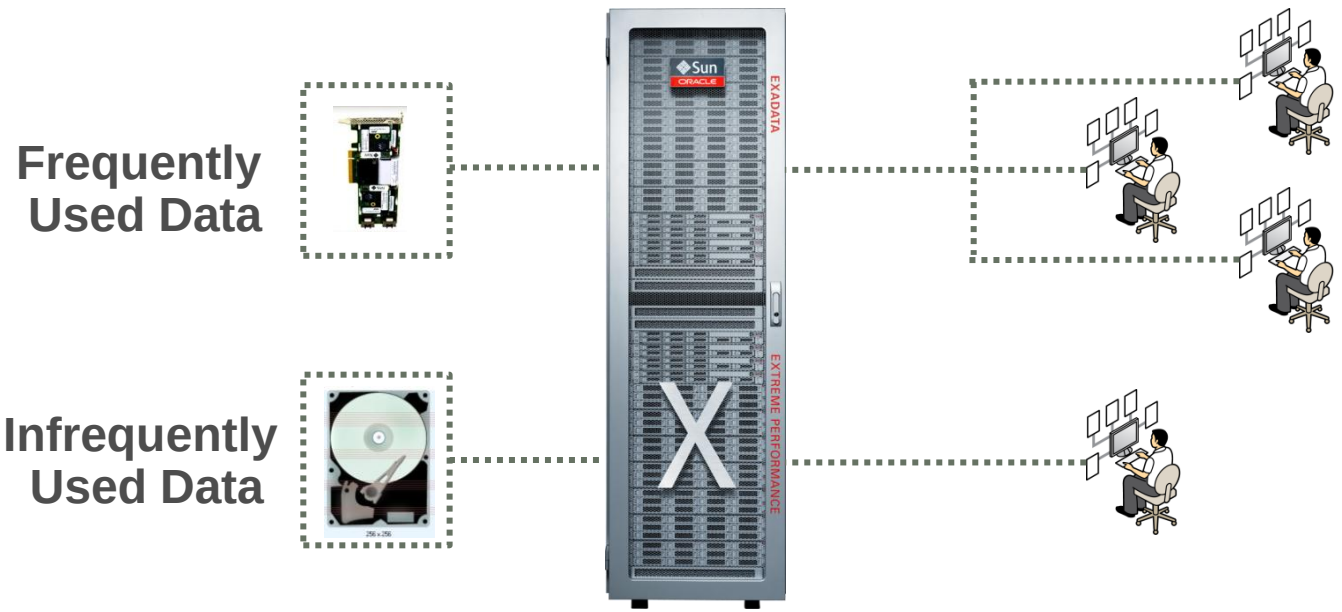
**Oracle Data Mining**

Uncover and predict

- Complex and predictive analytics embedded into Oracle Database 11g
- Reduce cost of additional hardware, management resources
- Improve performance by eliminating data movement and duplication

# Exadata Smart Flash Cache

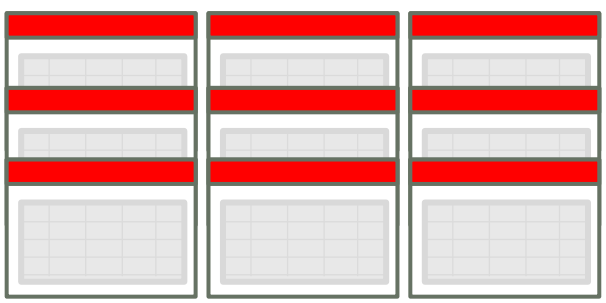
## Extreme Performance for OLTP Applications



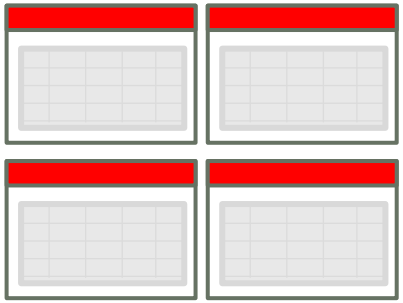

- Automatically caches frequently-accessed ‘hot’ data in flash storage
- Assigns the rest to less expensive disk drives
- Know when to avoid trying to cache data that will never be reused
- Process data at 50GB/sec and up to 1million I/Os per second

# Benefits Multiply

## Converting Terabytes to Gigabytes

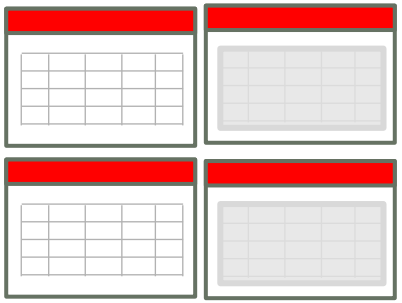


10 TB of User Data



1 TB of User Data

With 10x Compression



100 GB of User Data

With Partition Pruning

20 GB of User Data

With Storage Indexes

5 GB of User Data

With Smart Scan


Sub second “10 TB” Scan

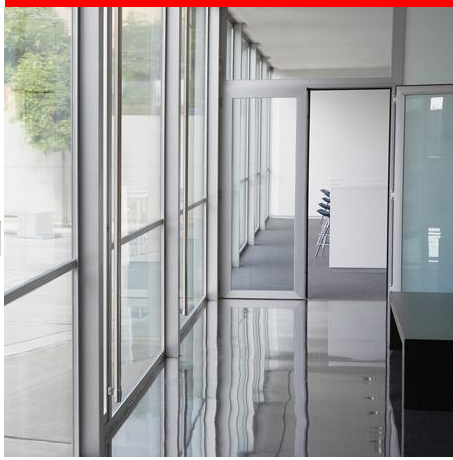
No Indexes

# Turkcell Runs 10x Faster on Exadata

## Compresses Data Warehouse by 10x

- Replaced high-end SMP Server and 10 Storage Cabinets
- Reduced Data Warehouse from 250TB to 27TB
  - Using OLTP & Hybrid Columnar Compression
  - Ready for future growth where data doubles every year
- Experiencing 10x faster query performance
  - Delivering over 50,000 reports per month
  - Average report runs reduced from 27 to 2.5 mins
  - Up to 400x performance gain on some reports

# Summary



- Implement the three Ps of Data Warehousing
  - Power – balanced hardware configuration
    - Make sure the system can deliver your SLA
  - Partitioning – Performance, Manageability, ILM
    - Make sure partition pruning and partition-wise joins occur
  - Parallel – Maximize the number of processes working
    - Make sure the system is not flooded using DOP limits & queuing
- Workload Management on a Data Warehouse
  - Use Database Resource Manager
    - Control maximum DOP each user can have
    - Control when statements should begin queue
    - Control what happens to “run away” queries

# Learn More

## Parallel Execution

- Oracle University Class
  - Choose between live virtual class or instructor-led
  - http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getCourseDesc?dc=**D71882GC10**

Read our blogs:

http://blogs.oracle.com/optimizer

http://blogs.oracle.com/datawarehousing

Best practices papers can be found here:
**http://www.oracle.com/technetwork/database/focus-areas/bi-datawarehousing/index.html**

# Q & A