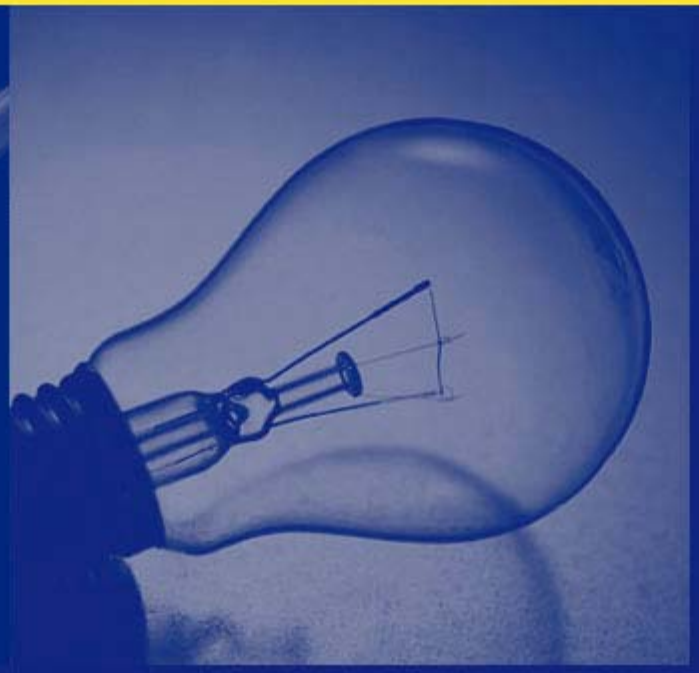
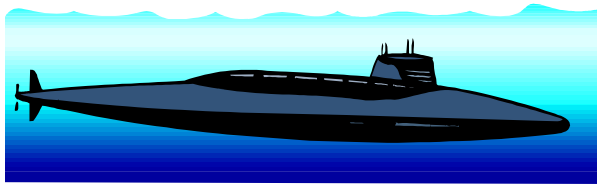
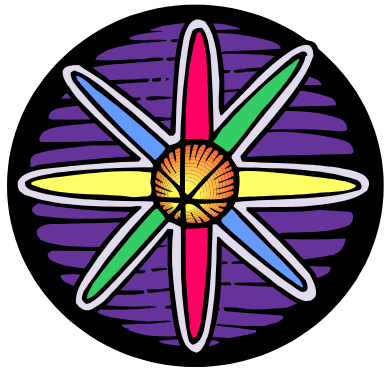
# Goal! Success Through Storage Tuning



Mike Ault
Oracle Domain Specialist
Quest Software
SOUG 2008

# Michael R. Ault
# Oracle Specialist



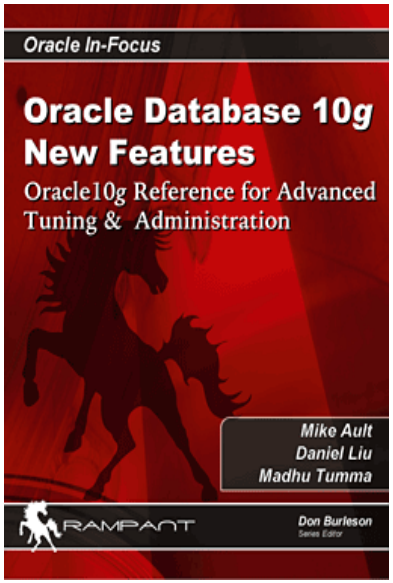


- Nuclear Navy 6 years
- Nuclear Chemist/Programmer 10 years
- Kennedy Western University Graduate
- Bachelors Degree Computer Science
- Certified in all Oracle Versions Since 6
- Oracle DBA, author, since 1990

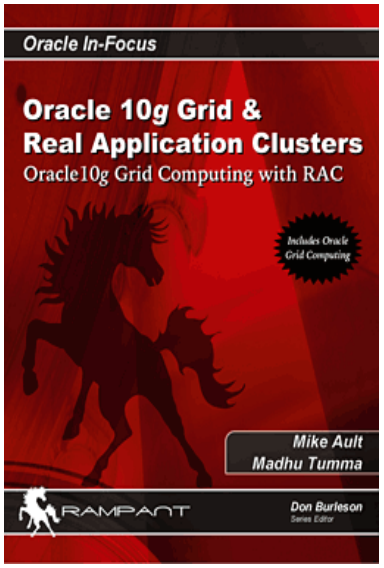
ORACLE | CERTIFIED PROFESSIONAL

QUEST SOFTWARE

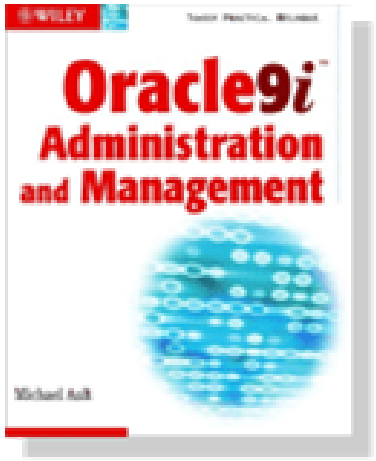
# Books by Michael R. Ault

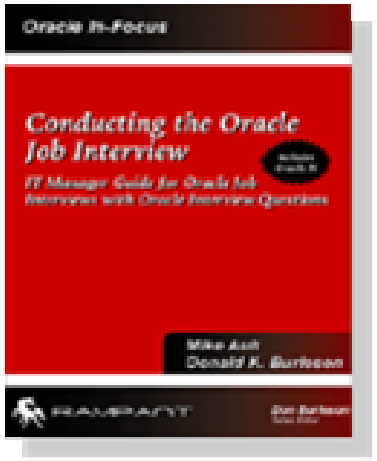


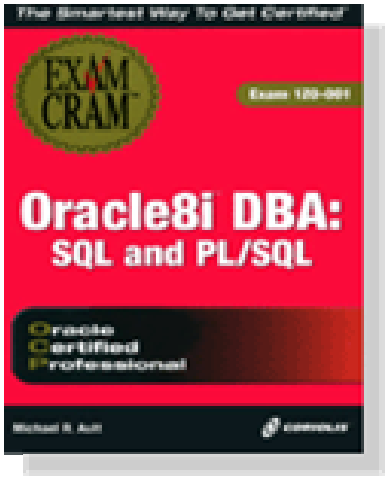


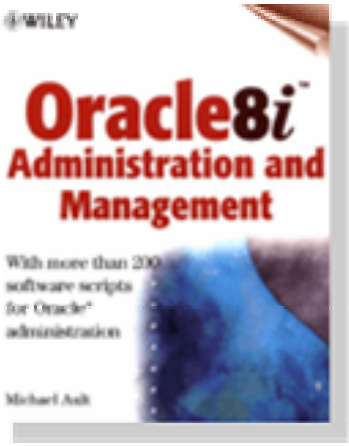






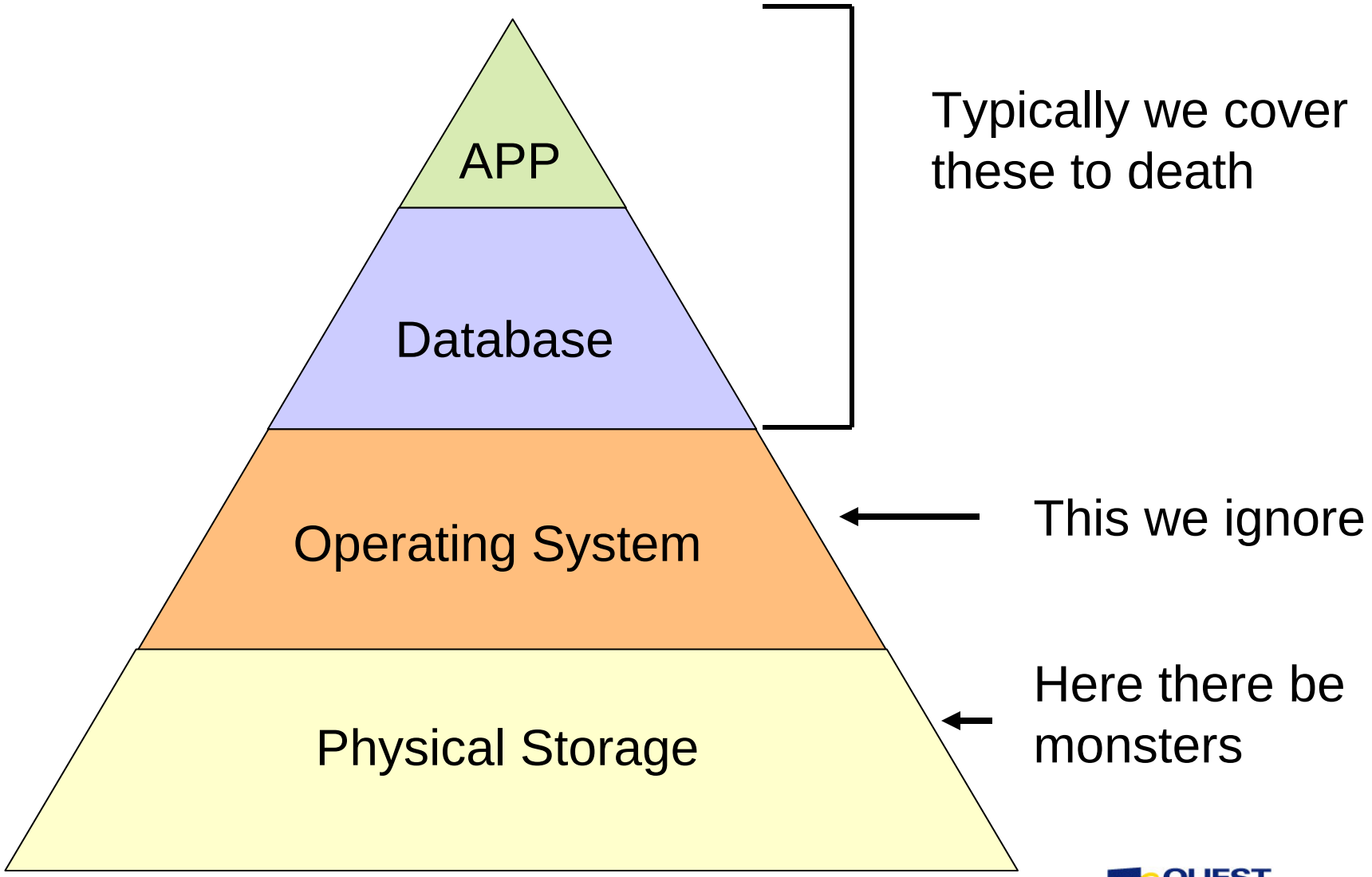




QUEST SOFTWARE

# The Many Tiers of Tuning




QUEST SOFTWARE

# Let's Tame the Physical Storage Monster

- Finding the optimal block size
- Matching striping parameters to I/O
- Fixing fragmented indexes
- Partitioning data to boost performance
- Understanding the underlying structure of ASM

QUEST SOFTWARE®

# Finding the Optimal Block Size

- Two dimensions to block size
  - OS block size
  - Oracle block size
- Usually we only control one
- OS reads in specific sizes
  - For example MAX_PHYS_IO
- OS usually in 512 byte increments
- Oracle is in 512 Byte increments but usually 2K, 4K, 8K, 16K or 32K

But what is optimal?

QUEST SOFTWARE

# Optimal OS Block Size

- May be fixed by your OS
- Usually not of major concern to Oracle
- As long as physical blocks and Oracle blocks align usually OK
- For example, 512 byte OS and any size Oracle block size align
- 4k OS and 2k Oracle do not
- OS block size should be a equal divisor of any expected Oracle block size

# Optimal Database Block Size

- Usually a tradeoff
- Bigger blocks give better storage efficiency
- Smaller blocks can help reduce contention
- Smaller blocks are usually better for RAC
- Default for 9i and later versions is 8k, before that was 4k

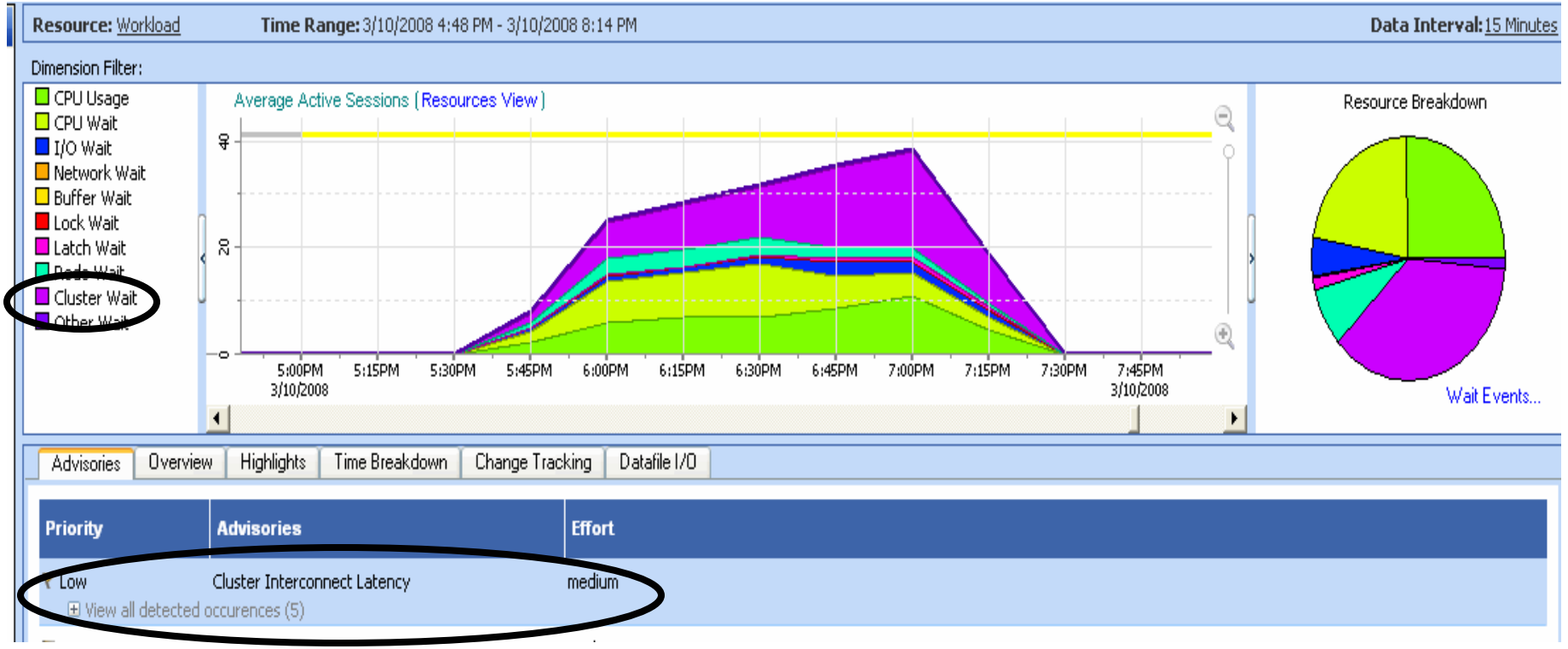
For Example, let's see what 8k looks like under moderate load in TPCC on RAC

Setup:

- 32 Warehouse TPCC
- 2-Node 32bit 3GHz hyper-threaded CPU RAC
- 3G memory each node
- ASM SAME for storage

QUEST SOFTWARE

# The Major Wait is Cluster Related

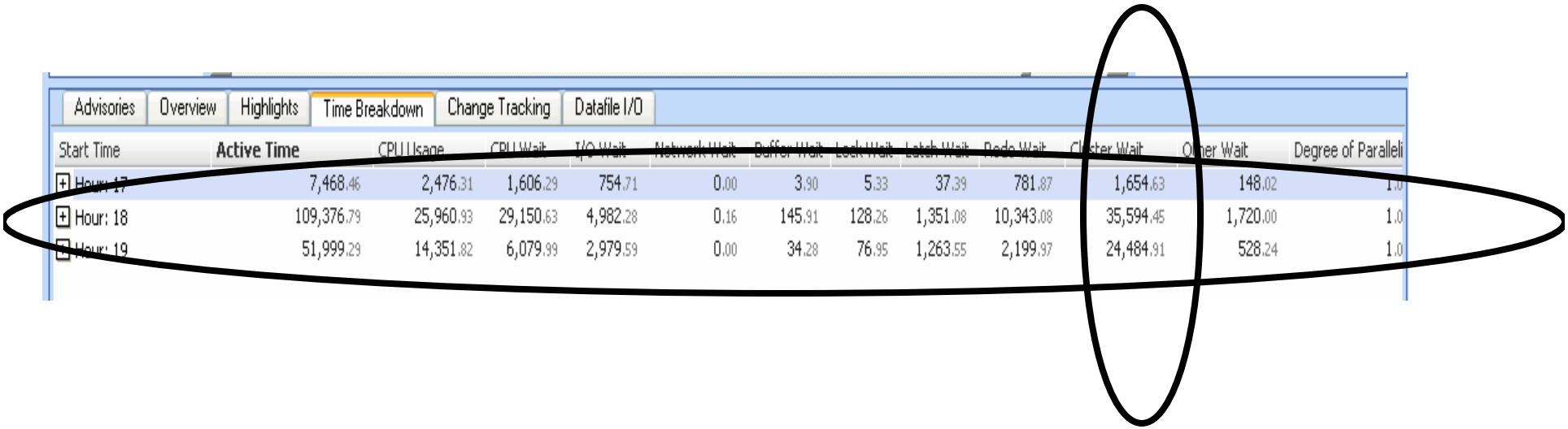



User load is 20-40 users incremented by 1

This is an aggregated view over cluster

QUEST SOFTWARE

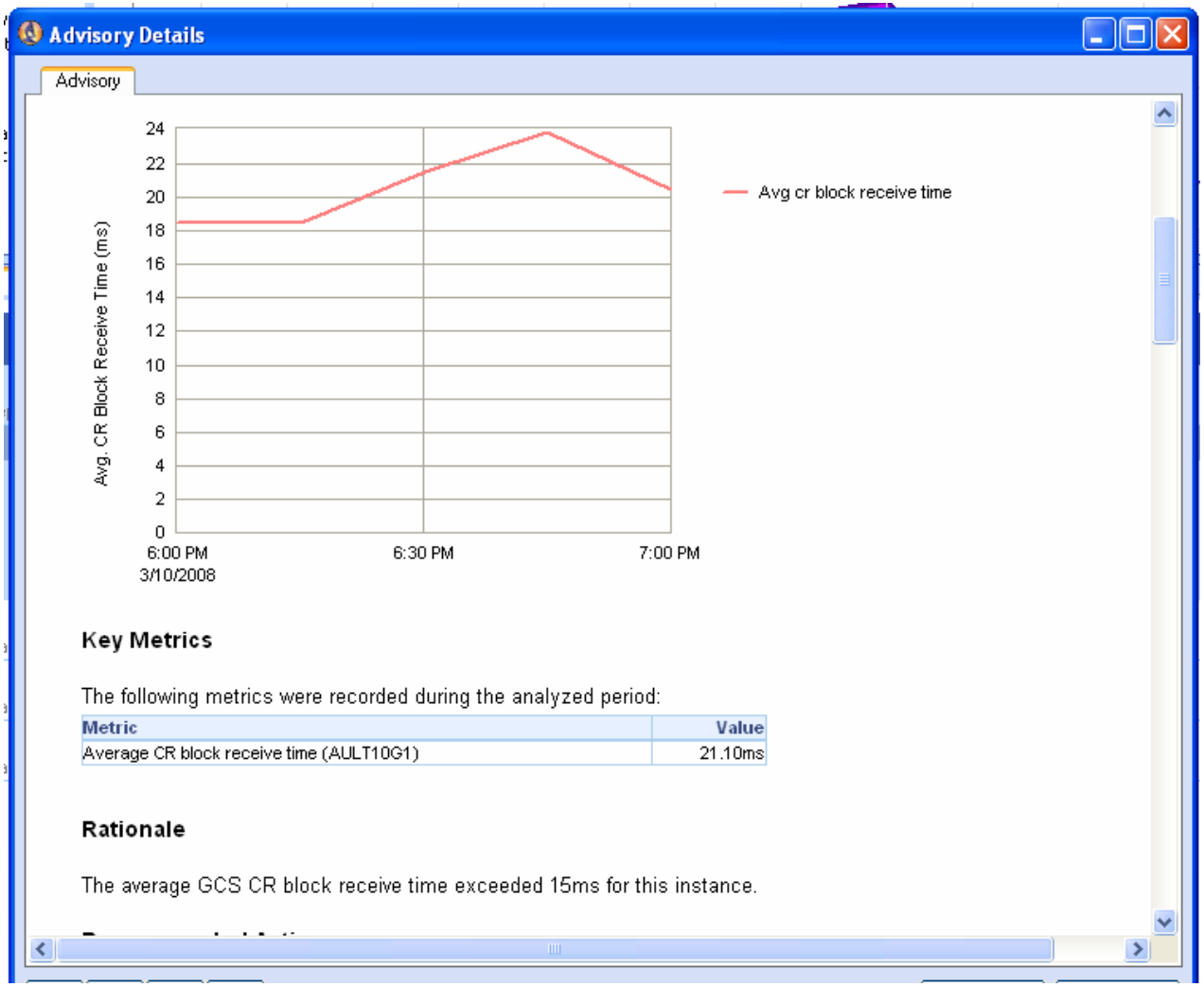
# A More Detailed Look




| Start Time | Active Time | CPU Usage | CPU Wait | I/O Wait | Network Wait | Buffer Wait | Lock Wait | Latch Wait | Redo Wait | Cluster Wait | Other Wait | Degree of Parallelism |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hour: 17 | 7,468.46 | 2,476.31 | 1,606.29 | 754.71 | 0.00 | 3.90 | 5.33 | 37.39 | 781.87 | 1,654.63 | 148.02 | 1.0 |
| Hour: 18 | 109,376.79 | 25,960.93 | 29,150.63 | 4,982.28 | 0.16 | 145.91 | 128.26 | 1,351.08 | 10,343.08 | 35,594.45 | 1,720.00 | 1.0 |
| Hour: 19 | 51,999.29 | 14,351.82 | 6,079.99 | 2,979.59 | 0.00 | 34.28 | 76.95 | 1,263.55 | 2,199.97 | 24,484.91 | 528.24 | 1.0 |

QUEST SOFTWARE

# The Actual Numbers




QUEST SOFTWARE

# The Results

## Run Information

| | | | |
|---|---|---|---|
| Test Run Id | 131 | Status | Completed |
| Start Time | 3/10/2008 5:54:23 PM | Stop Time | 3/10/2008 7:21:41 PM |
| Comment | | | |

## Profile Information

| | |
|---|---|
| Profile Name | ault10g |
| Driver Name | Oracle |
| Net Service Name | ault10g |
| User Name | oltp |
| Password | ****** |
| Use Clustering(Node%) | Yes (50,50) |

## Statistics for the Userload:

| User Load | TPS | kBPS | Avg. Response Time (sec) | Avg. Transaction Time (sec) | Total Executions | Total Rows | Total Errors |
|---|---|---|---|---|---|---|---|
| 23 | 106.16 | 45.340 | 0.200 | 0.215 | 19123 | 25837 | 0 |

QUEST SOFTWARE

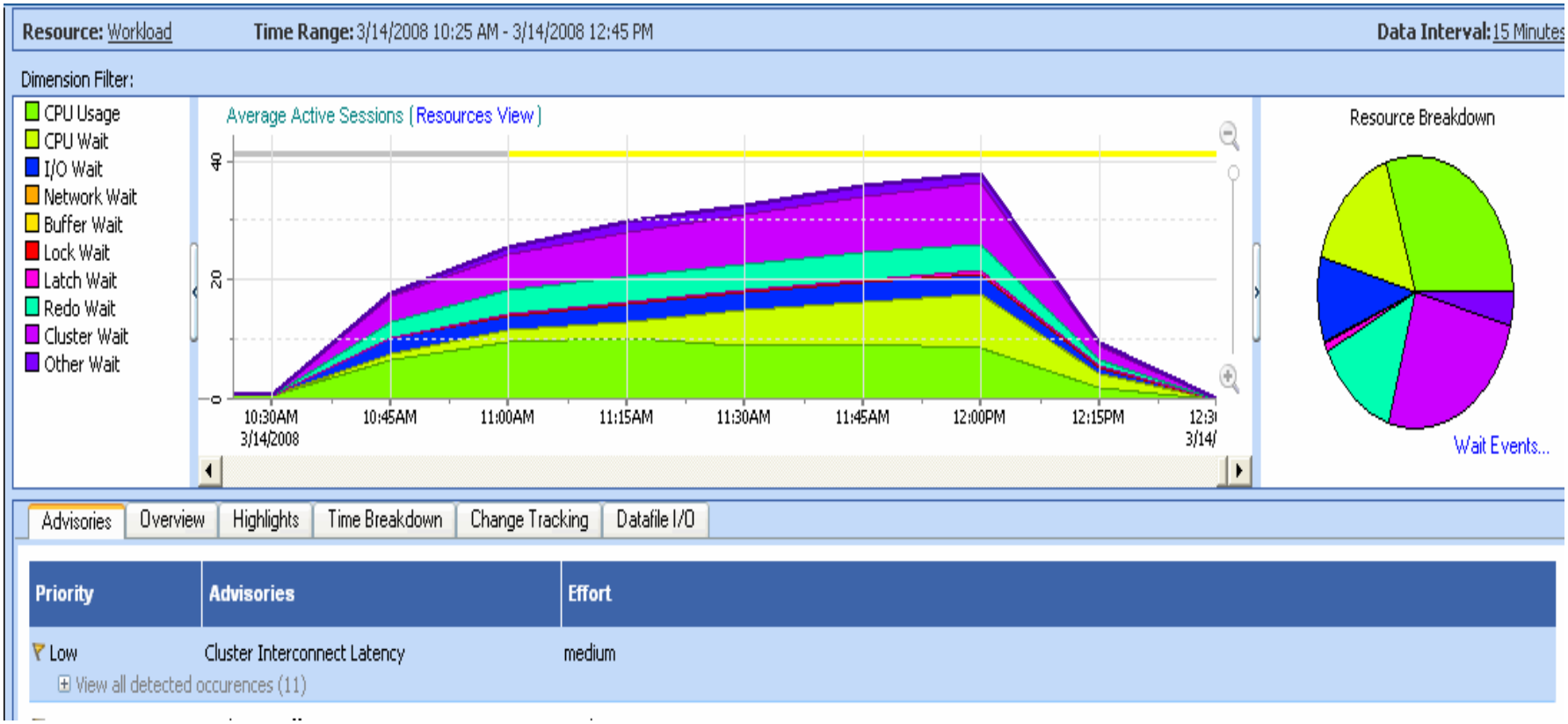
# Factors Effecting Optimal Block Size

- OS limitations
- Type of Application
- Type of Database (Single instance or RAC)

Does it really make a difference?

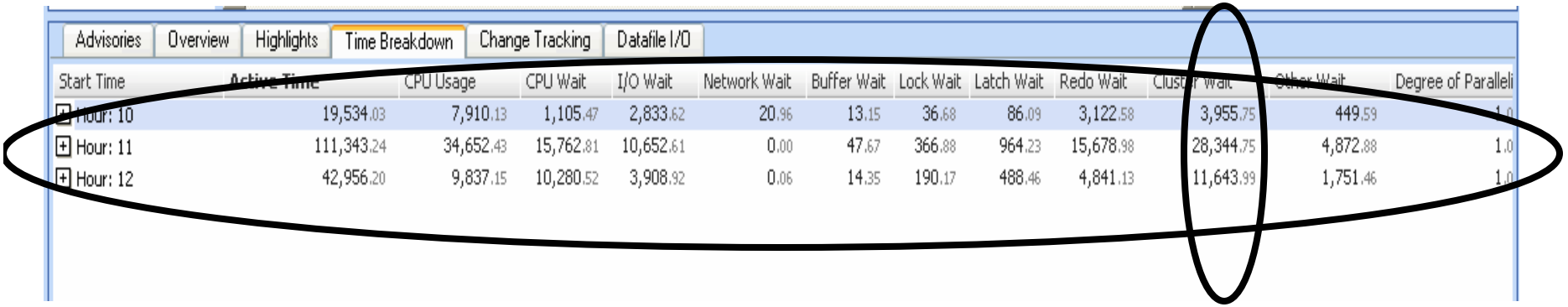
Let's rebuild the example database with a 2K block size for tables and indexes and rerun the test on the same exact database and hardware (only differences are a 512 megabyte 2k buffer area per instance and the 2-2K block size tablespaces for data and indexes.)

# The Second Run




QUEST SOFTWARE

# A More Detailed Look

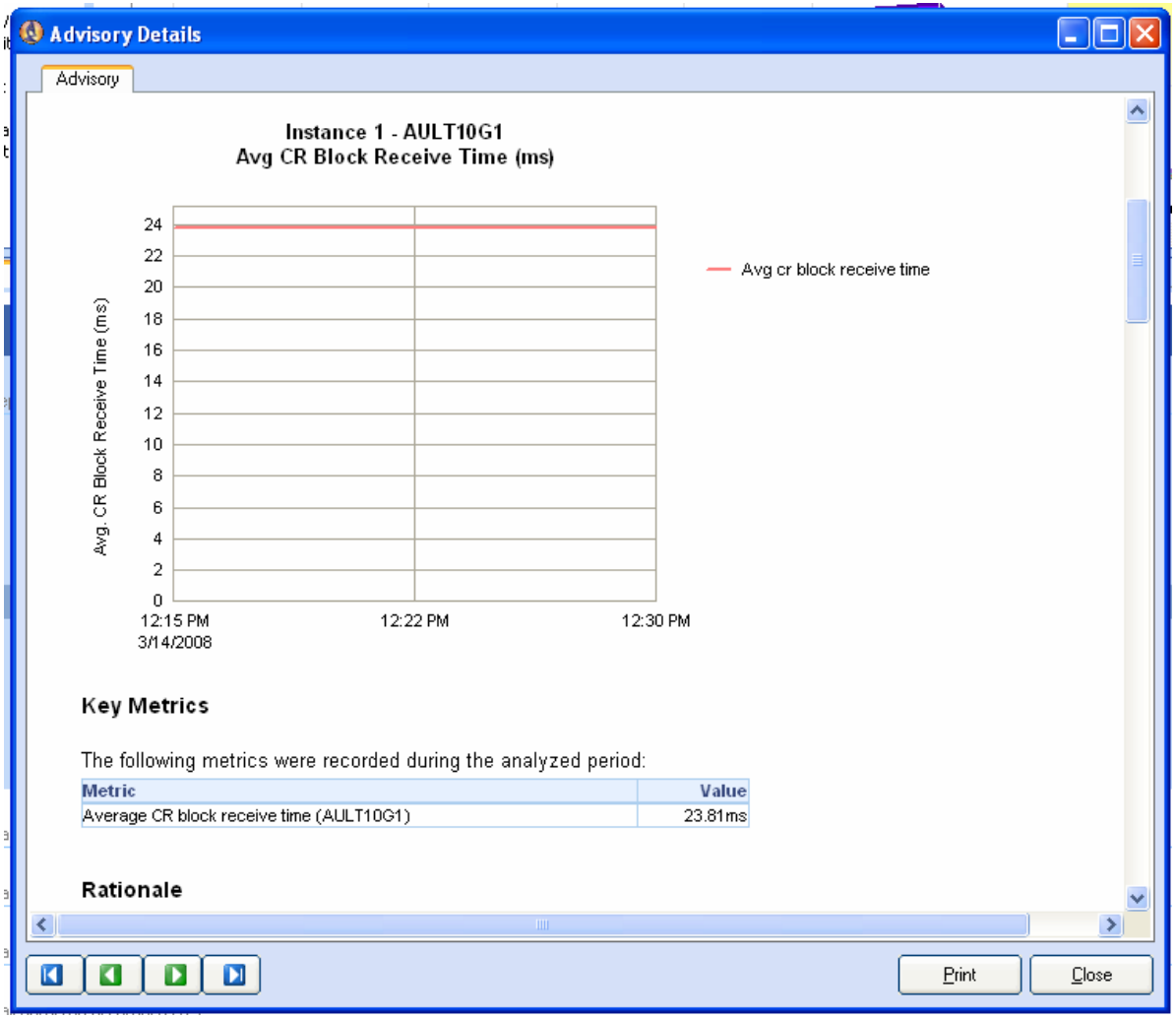


Advisories | Overview | Highlights | Time Breakdown | Change Tracking | Datafile I/O

| Start Time | Active Time | CPU Usage | CPU Wait | I/O Wait | Network Wait | Buffer Wait | Lock Wait | Latch Wait | Redo Wait | Cluster Wait | Other Wait | Degree of Parallelism |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hour: 10 | 19,534.03 | 7,910.13 | 1,105.47 | 2,833.62 | 20.96 | 13.15 | 36.68 | 86.09 | 3,122.58 | 3,955.75 | 449.59 | 1.0 |
| Hour: 11 | 111,343.24 | 34,652.43 | 15,762.81 | 10,652.61 | 0.00 | 47.67 | 366.88 | 964.23 | 15,678.98 | 28,344.75 | 4,872.88 | 1.0 |
| Hour: 12 | 42,956.20 | 9,837.15 | 10,280.52 | 3,908.92 | 0.06 | 14.35 | 190.17 | 488.46 | 4,841.13 | 11,643.99 | 1,751.46 | 1.0 |

QUEST SOFTWARE

# The Actual Numbers




QUEST SOFTWARE

# The Results

**Run Information**

| | | | |
|---|---|---|---|
| Test Run Id | 144 | Status | Completed |
| Start Time | 3/14/2008 10:48:22 AM | Stop Time | 3/14/2008 12:18:08 PM |
| Comment | | | |

**Profile Information**

| | |
|---|---|
| Profile Name | ault10g |
| Driver Name | Oracle |
| Net Service Name | ault10g |
| User Name | oltp |
| Password | ****** |
| Use Clustering(Node%) | Yes (50,50) |

**Statistics for the Userload:**

| User Load | TPS | kBPS | Avg. Response Time (sec) | Avg. Transaction Time (sec) | Total Executions | Total Rows | Total Errors |
|---|---|---|---|---|---|---|---|
| 25 | 113.64 | 48.576 | 0.197 | 0.218 | 20442 | 27683 | 0 |

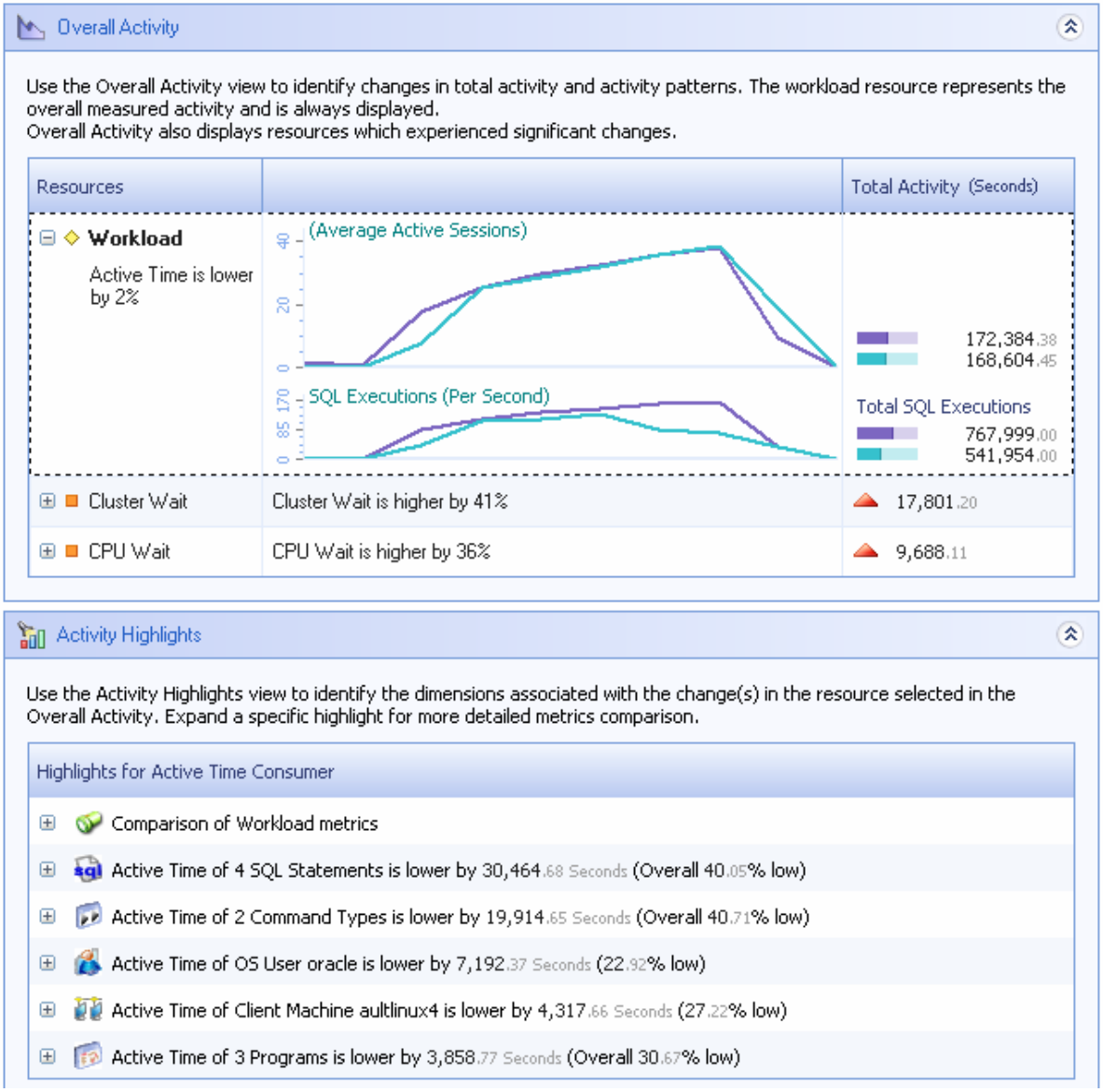
QUEST SOFTWARE

# It Appears to have helped but can we know for sure?

- Need to compare the two time periods
- Could do it with AWR reports
- Naw…let's do it the easy way!

QUEST SOFTWARE®

# Comparing the Full Periods




Observed Base

Observed time range and dimension activity

**Time range:**
3/10/2008 5:29 PM - 3/10/2008 7:45 PM

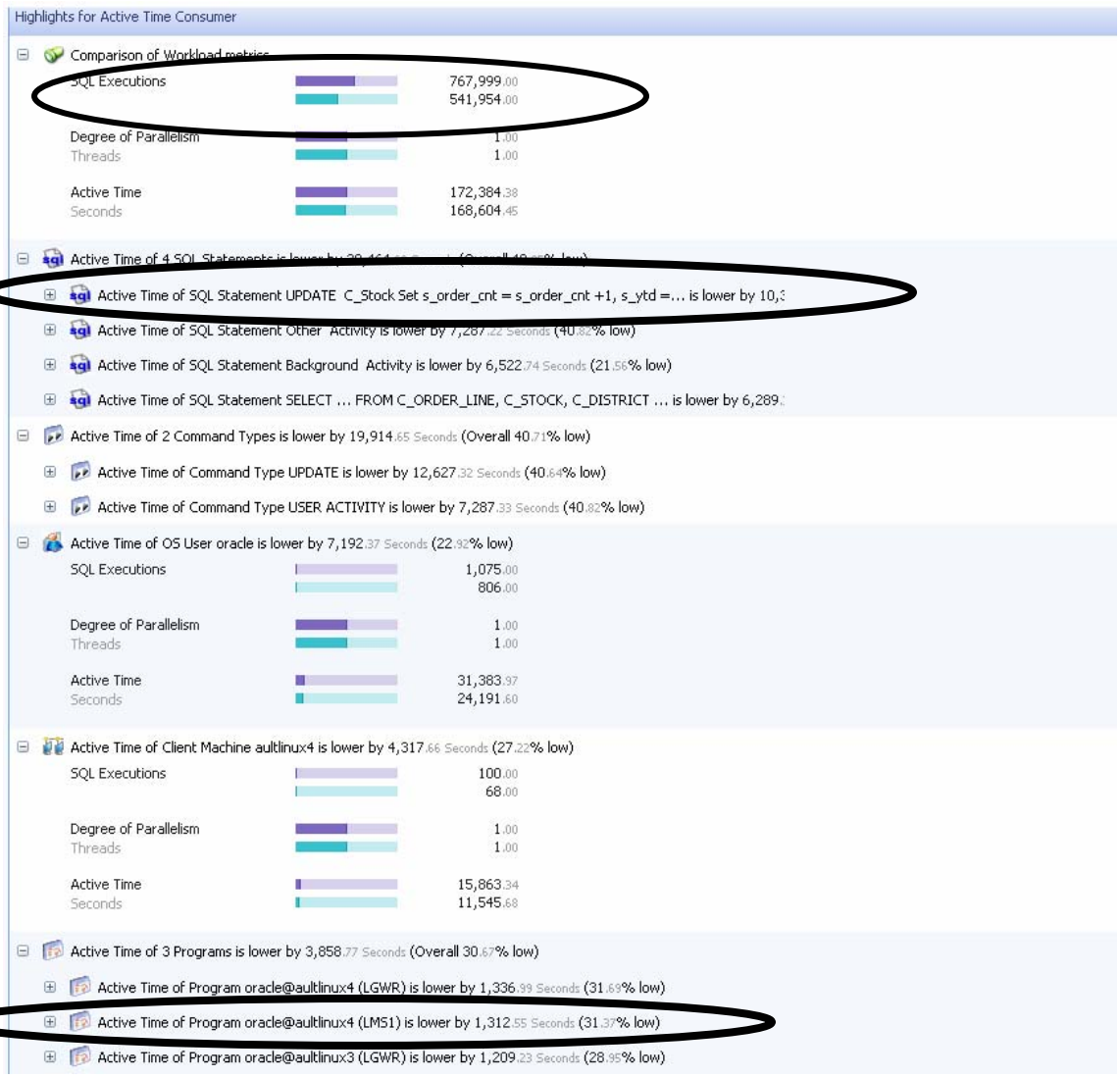
**Dimension Activity:**
Total Instance Activity

Base time range and dimension activity

**Time range:**
3/14/2008 10:29 AM - 3/14/2008 12:45 PM

QUEST SOFTWARE

# SQL, User and Process Details




Observed Base

More SQL

Faster UPDATES

LMS Less Loaded

QUEST SOFTWARE

# Comparing Cluster Waits


QUEST SOFTWARE

# Did Going to 2K Help?

- Overall waits were lower
- Was able to do more TPS
- Response time was faster

YES!

QUEST SOFTWARE®

# Matching striping parameters to I/O

- RAID allows us to stripe data cross volumes
- Striping allows for a larger relative IO bandwidth
- You have to avoid blocking reads at the disk level

QUEST SOFTWARE®

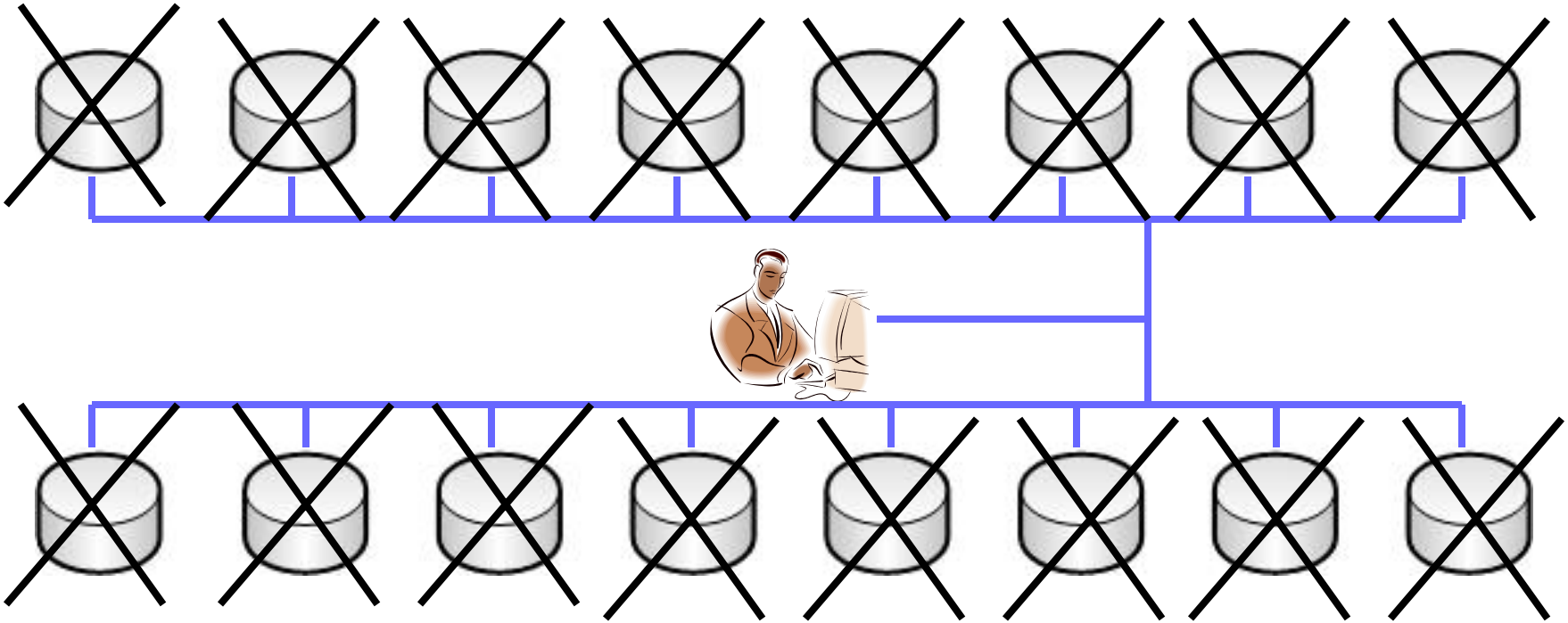
# What is a blocking read?

- DB_FILE_MULTI_BLOCK_READ_COUNT 16
- DB_BLOCK_SIZE 8K
- Stripe depth per disk 8K
- Stripe width 128K (16 disks by 8K per disk)

Is this a good setup?

QUEST SOFTWARE®

# User 1 Does SELECT *



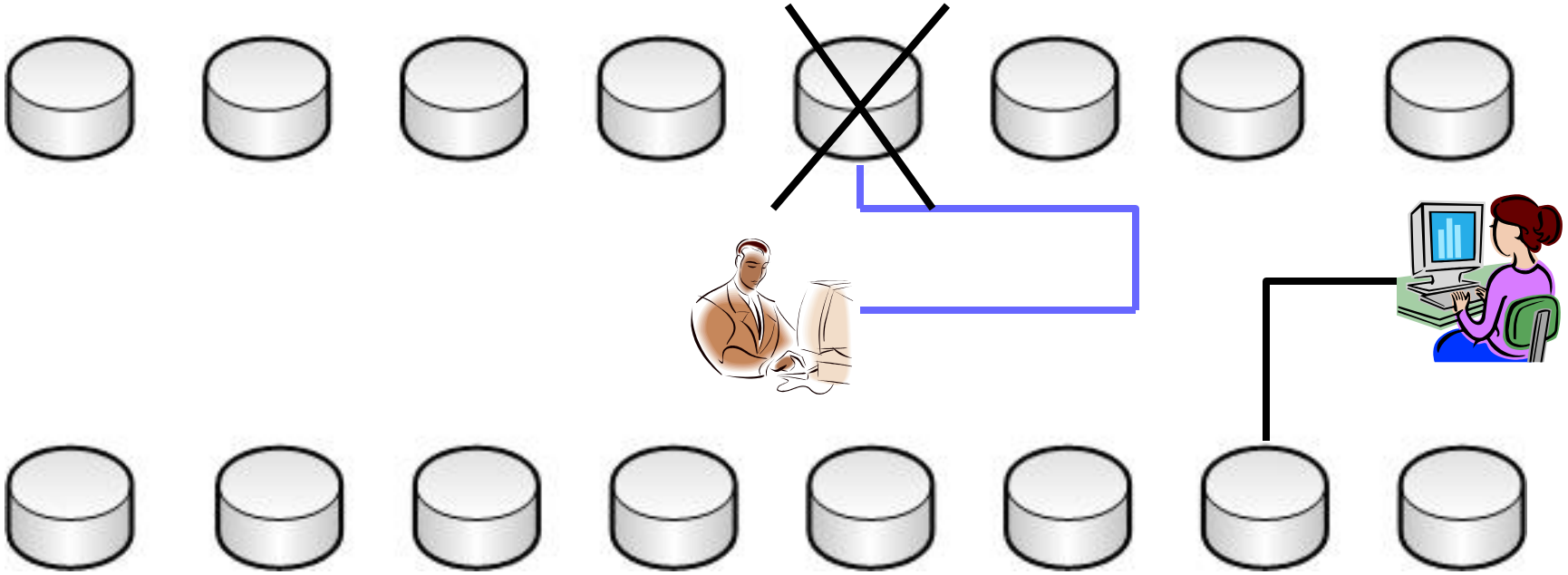
All 16 disks are read for a single 8k piece at once, all other users are blocked

QUEST SOFTWARE

# Better

- DB_FILE_MULTI_BLOCK_READ_COUNT 16
- DB_BLOCK_SIZE 8K
- Stripe depth per disk 128K
- Stripe width 2048K (16 disks by 128K per disk)

# User 1 Does SELECT *



Only 1 disk is read for a single 128k read at once, other users are blocked only if they try to read the same disk

QUEST SOFTWARE

# Of Course

- This is over simplified
- There are many other aspects
- But the principle is still valid

QUEST SOFTWARE®

# So…

- Size stripe depth per disk at largest read size
- Align DBFMBRC * DBBS to stripe depth
- This allows max concurrency
- This allows minimal contention

QUEST SOFTWARE®

# Fixing Fragmented Indexes

- What is meant by fragmented index?
- Can an index become unbalanced?
- What is index white space?

QUEST SOFTWARE®

# Let's Look! First create a Table

```
SQL> create table fragit(fragitpk
number,fragitchar varchar2(256), fragitnum
number);

Table created.

SQL> create sequence fragit_seq;

Sequence created.
```

QUEST SOFTWARE

# Let's Populate the PK Monotonically

```
SQL> create or replace trigger pk_fragit_trigger
  2  before insert on fragit
  3  for each row
  4  begin
  5  select fragit_seq.nextval into :new.fragitpk
from dual;
  6  end;
  7  /

Trigger created.

SQL> alter table fragit add constraint pk_fragit
primary key (fragitpk);

Table altered.
```

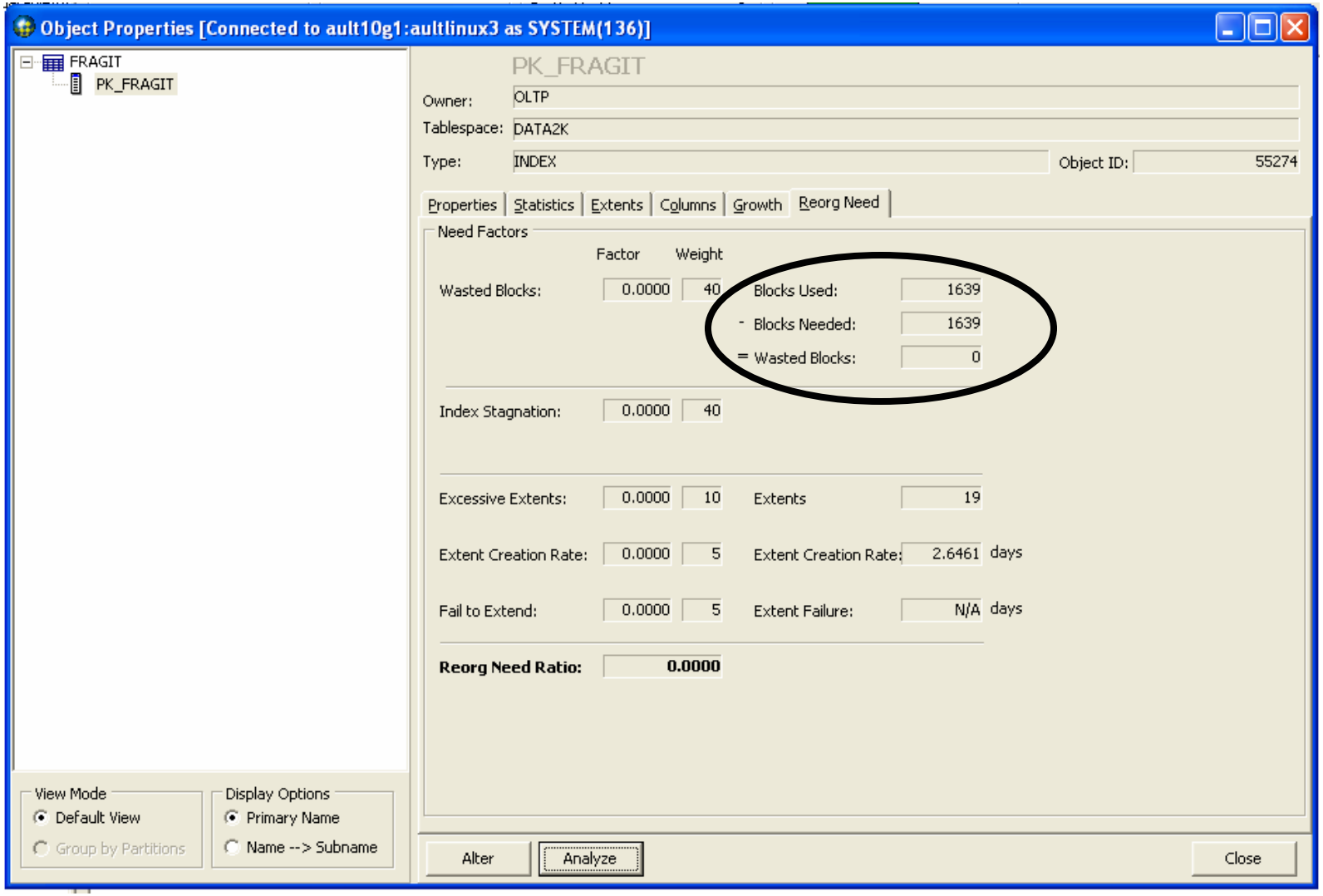
QUEST SOFTWARE

# Add some rows so we can populate Table/Index

```
SQL> insert into fragit(fragitchar,fragitnum) select
object_name,object_id from dba_objects;
50418 rows created.
SQL> /
50418 rows created.
SQL> /
50418 rows created.
SQL> /
50418 rows created.

SQL> commit;
Commit complete.

SQL> analyze table fragit compute statistics;
Table analyzed.

SQL> analyze index pk_fragit validate structure;
Index analyzed.
```

QUEST SOFTWARE

# Structure of Fully Populated Index




QUEST SOFTWARE

# More Statistics

```
SQL> select blocks,btree_space,
  2* used_space,pct_used from index_stats;

    BLOCKS BTREE_SPACE USED_SPACE   PCT_USED
---------- ----------- ---------- ----------
      2048     3070968    3053997        100

1 row selected.
```

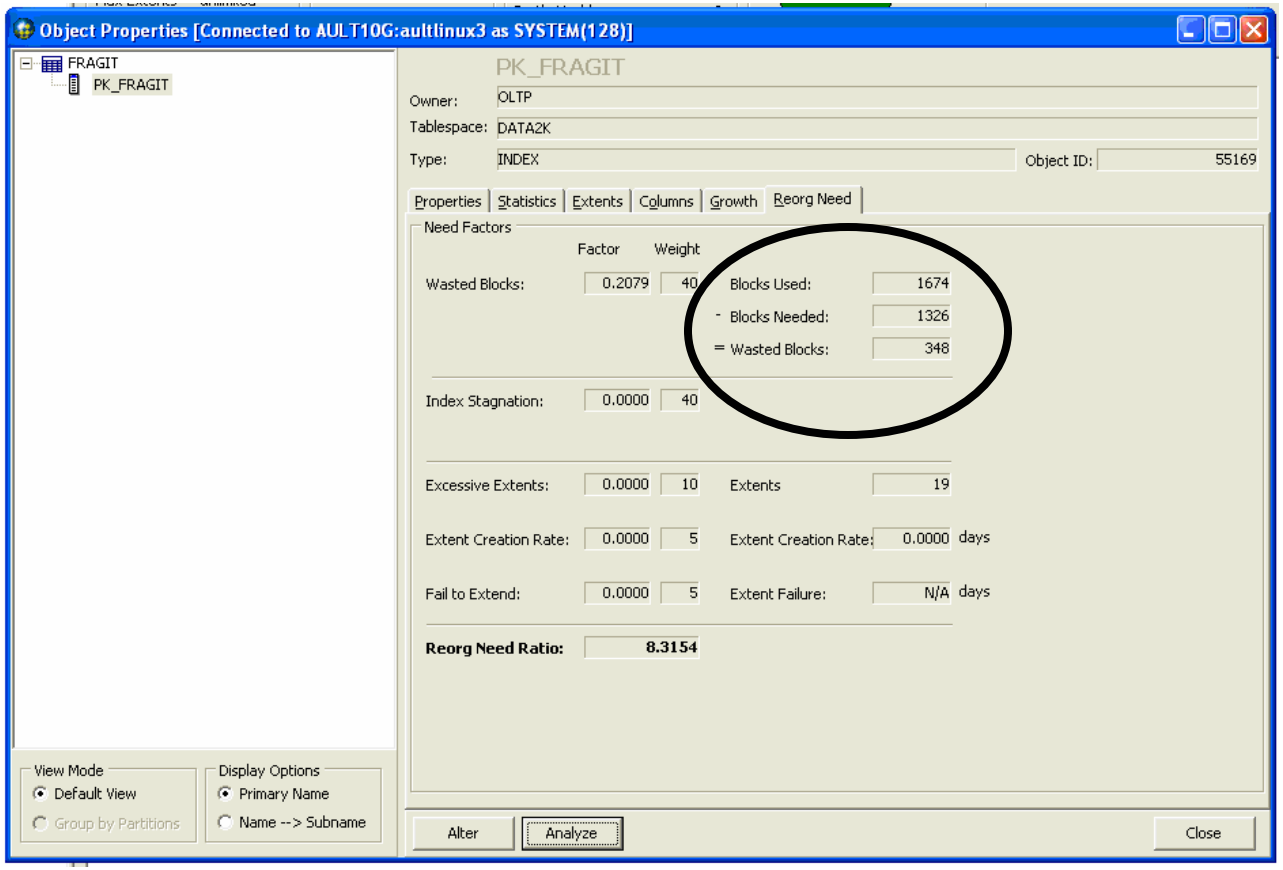
QUEST SOFTWARE

# Let's Internally Fragment the Index

```
SQL> delete fragit where mod(fragitpk,2)=0;

101603 rows deleted.

SQL> commit;

Commit complete.
```

QUEST SOFTWARE

# What does it do to Structure?




Nothing! But index now has white space that won't be used

QUEST SOFTWARE

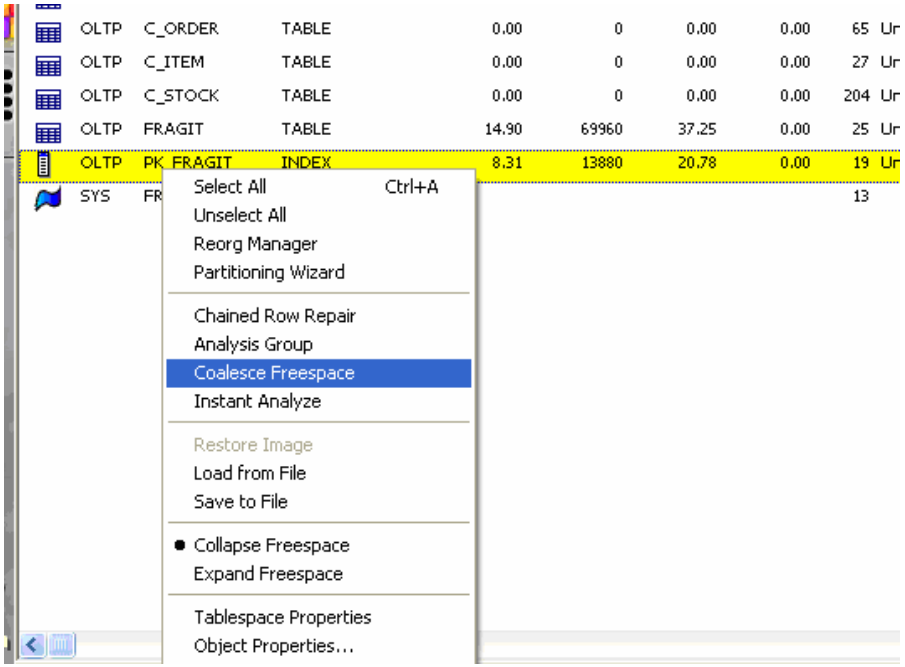
# Nope, no change here either

```
SQL> select blocks,btree_space,
  2* used_space,pct_used from index_stats;

    BLOCKS BTREE_SPACE USED_SPACE   PCT_USED
---------- ----------- ---------- ----------
      2048     3070968    3053997        100

1 row selected.
```

QUEST SOFTWARE

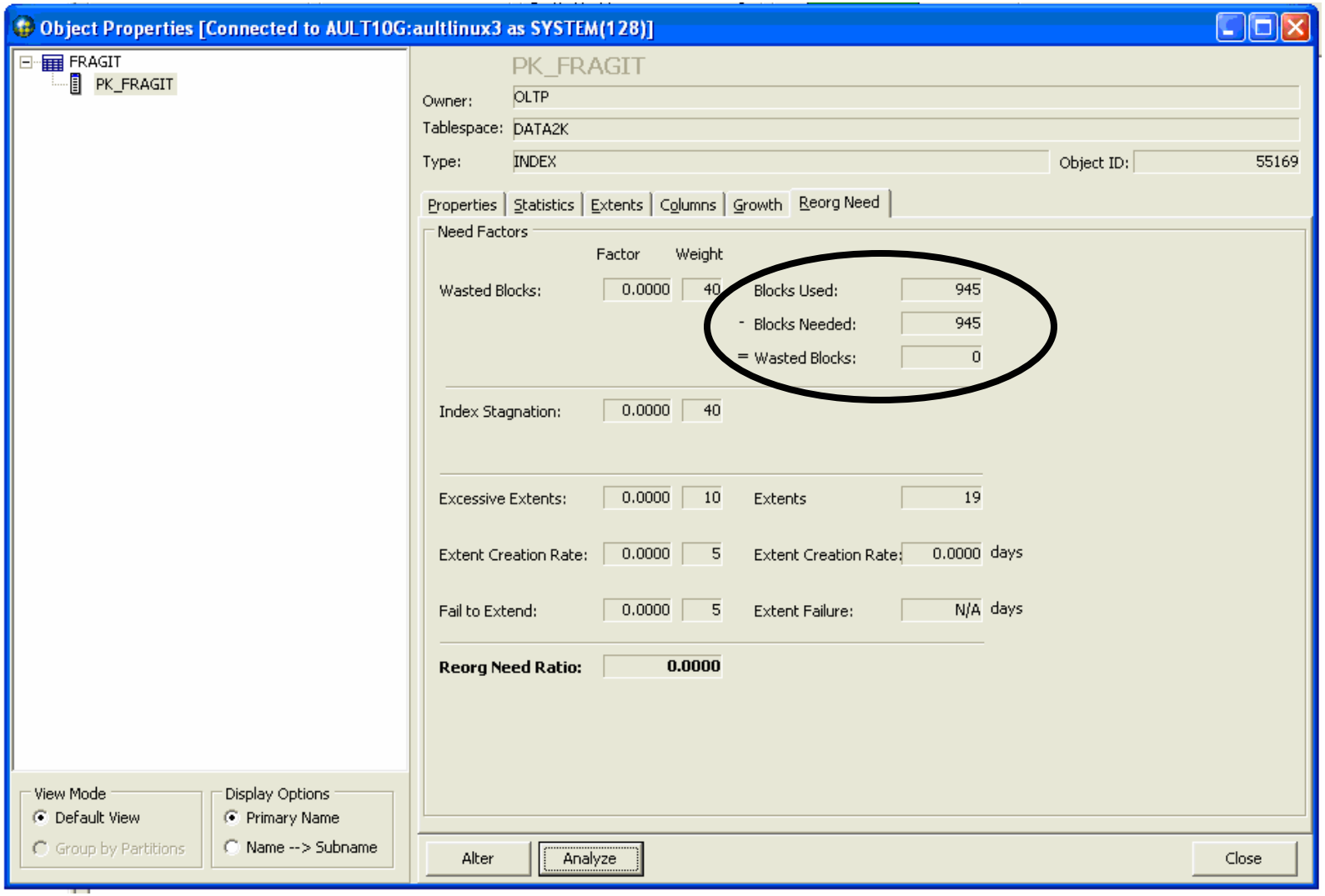
# Let's Coalesce

```
SQL> alter index pk_fragit
coalesce;

Index altered.

SQL> analyze table fragit
compute statistics;

Table analyzed.

SQL> analyze index pk_fragit
validate structure;

Index analyzed.
```



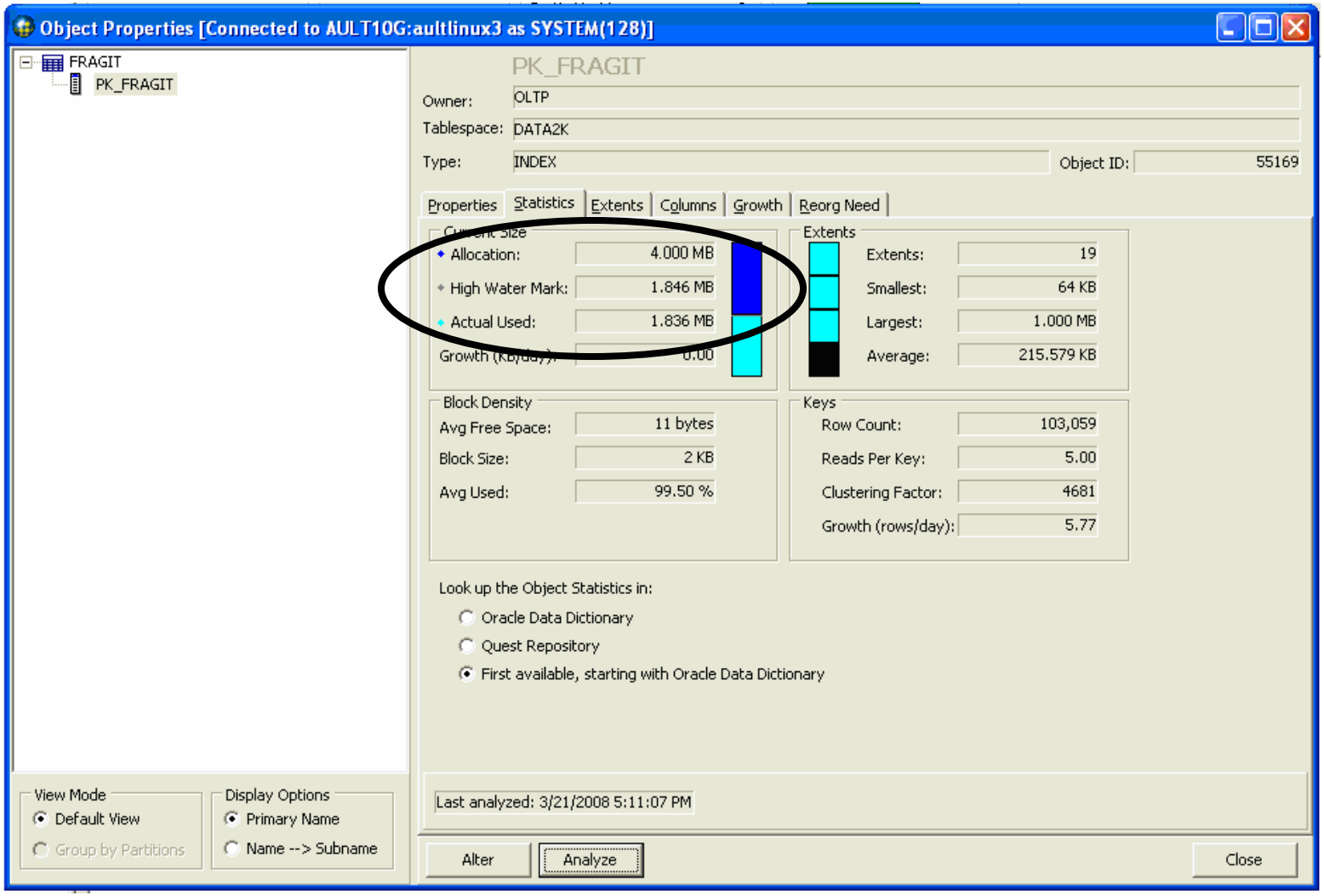

QUEST SOFTWARE

# Now we see a difference!




QUEST SOFTWARE

# But Size is the Same




QUEST SOFTWARE

# We see it here as well

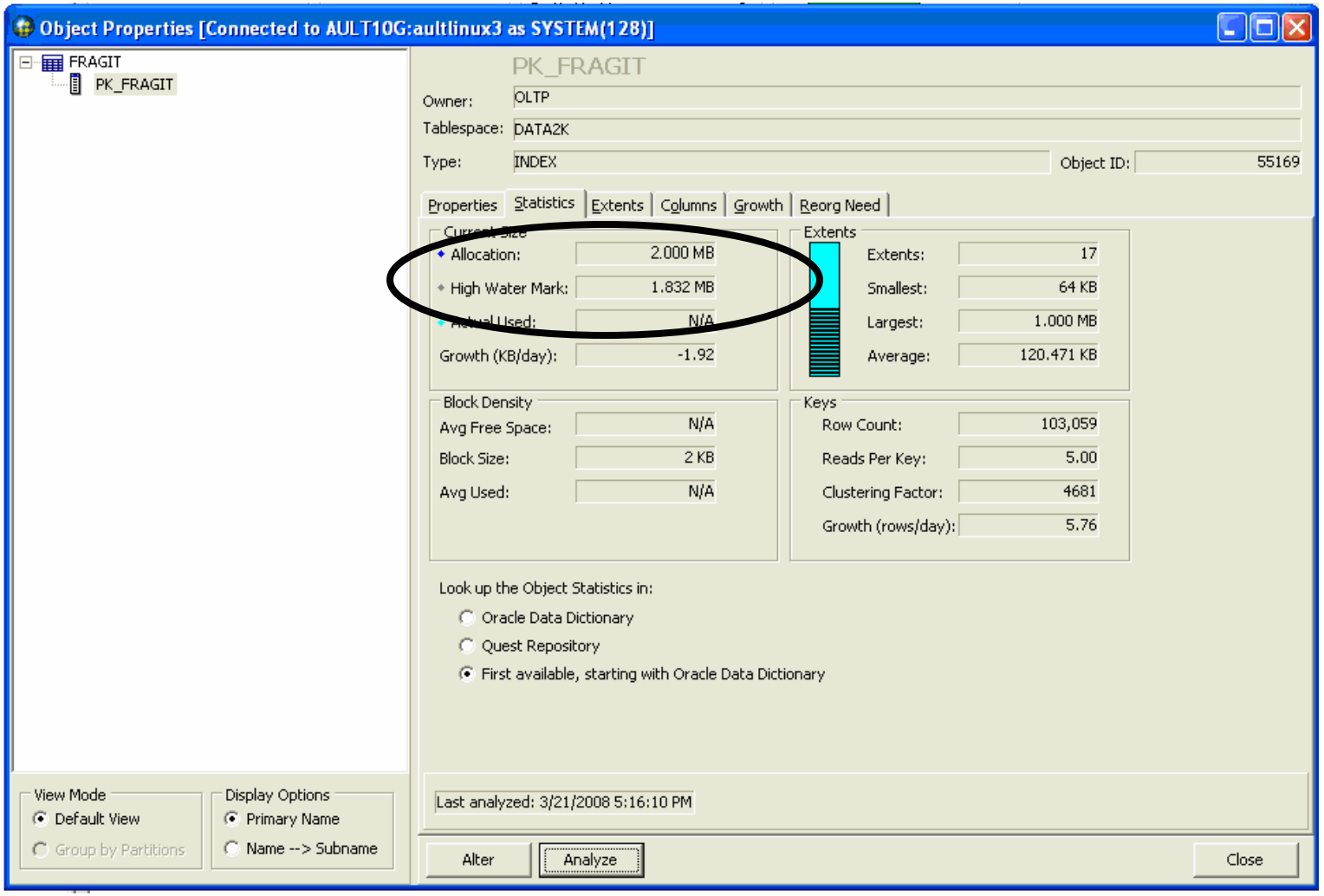
```
SQL> select blocks,btree_space,used_space,pct_used from
index_stats;

    BLOCKS BTREE_SPACE USED_SPACE   PCT_USED
---------- ----------- ---------- ----------
      2048     1741232    1529152         88

1 row selected.
```

QUEST SOFTWARE®

# Let's Rebuild

```
SQL> alter index pk_fragit rebuild;

Index altered.

SQL> analyze table fragit compute statistics;

Table analyzed.

SQL> analyze index pk_fragit validate structure;

Index analyzed.
```

# Now size changes as well




QUEST SOFTWARE

# Confirmed, Index is Optimized for structure and size

```
SQL> select blocks,btree_space,used_space,pct_used from
index_stats;


    BLOCKS BTREE_SPACE USED_SPACE   PCT_USED
---------- ----------- ---------- ----------
      1024     1718880    1529102          89
```

QUEST SOFTWARE

# Unbalanced Index

- Not possible
- Unbalanced is a misnomer
- Actually refers to "right handed" index
- Occurs in monotonically populated indexes
- Caused by block split where right hand block gets active entries
- Oracle handles by placing entries in block split in left hand block
- Can cause hot blocks

# Partitioning Data to Boost Performance

- Oracle does FTS at 7-10% of data
- A FTS on millions of rows is costly
- By partitioning the table allows a partition to be scanned instead of full table
- Partition pruning is the main performance benefit
- Parallel query also helps with partitioned tables
- Indexes are usually partitioned as well

QUEST SOFTWARE®

# When Should I Partition?

- When FTS is the method of access
- When FTS is taking large amounts of time
- When table can be partitioned on a clear partitioning key or keys
- If needed for data archive/roll-off

QUEST SOFTWARE®

# Partitioning Example: First a Normal Table

```
CREATE TABLE H_LINEITEM
(
l_orderkey        NUMBER            NOT NULL,
l_partkey         NUMBER            NOT NULL,
l_suppkey         NUMBER            NOT NULL,
l_linenumber      NUMBER            NOT NULL,
l_quantity        NUMBER            NOT NULL,
l_extendedprice   NUMBER            NOT NULL,
l_discount        NUMBER            NOT NULL,
l_tax             NUMBER            NOT NULL,
l_returnflag      CHAR (1),
l_linestatus      CHAR (1),
l_shipdate        DATE,
l_commitdate      DATE,
l_receiptdate     DATE,
l_shipinstruct    CHAR (25),
l_shipmode        CHAR (10),
l_comment         VARCHAR (44)
)
NOLOGGING
PARALLEL (DEGREE 8 INSTANCES 2)
NOCACHE
COMPRESS;
```

QUEST SOFTWARE

# Worst Case: Full Table Scan

```
SQL> set timing on
SQL> select count(*) from h_lineitem where l_receiptdate
is null or l_receiptdate is not null;

  COUNT(*)
----------
  59986052

1 row selected.

Elapsed: 00:01:32.06
```

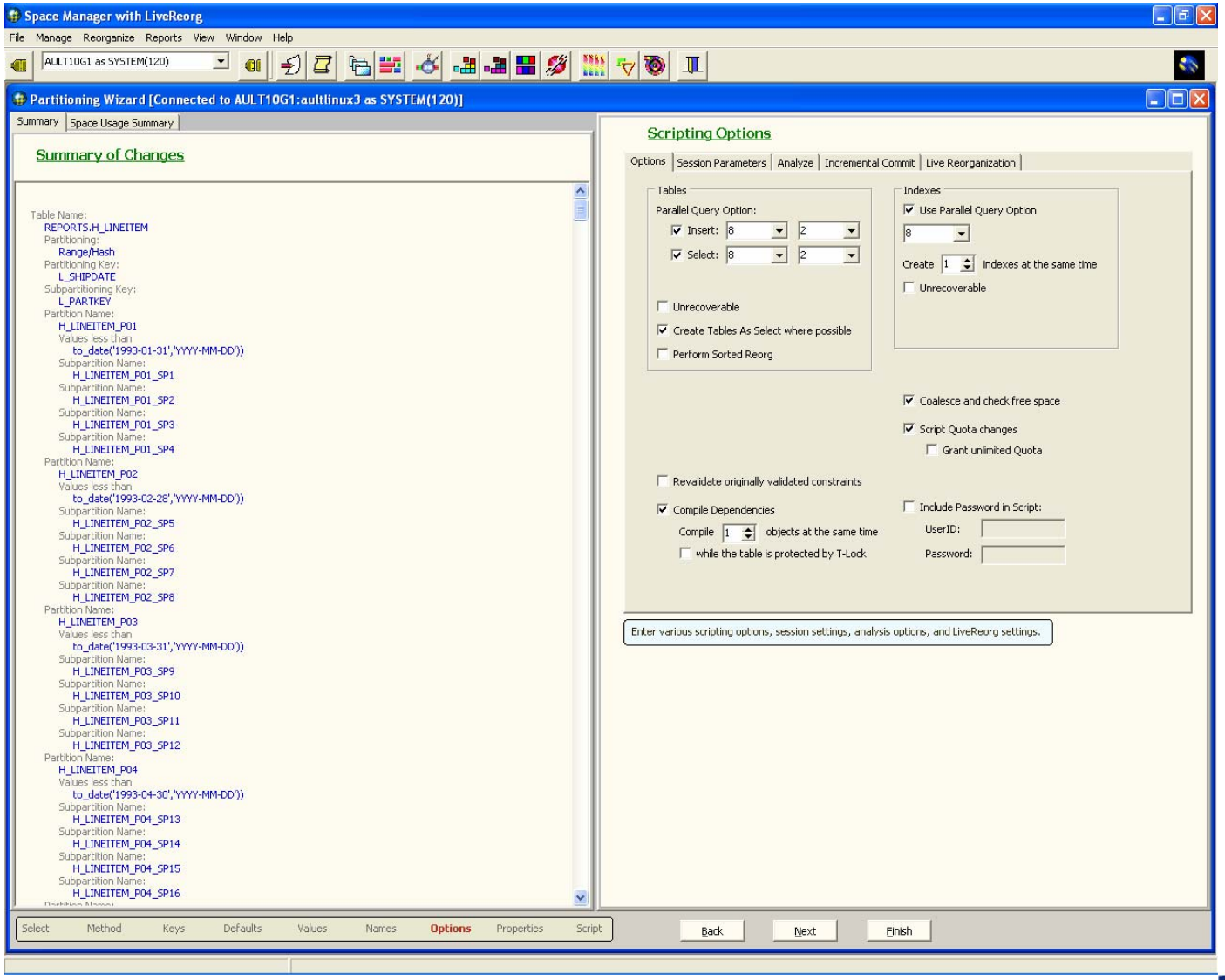
QUEST SOFTWARE

# Now Lets Partition

```
CREATE TABLE H_LINEITEM (
l_orderkey     NUMBER          NOT NULL,
l_partkey      NUMBER          NOT NULL,
…
l_commitdate   DATE,
l_receiptdate  DATE,
l_shipinstruct CHAR (25),
l_shipmode     CHAR (10),
l_comment      VARCHAR (44))
NOLOGGING PARALLEL (DEGREE 8 INSTANCES 2) NOCACHE COMPRESS
partition by range(l_shipdate)
subpartition by hash(l_partkey)
subpartitions 4
store in (quest_dat2, quest_dat1)
(  partition lin_1993_01 values less than (to_date('1993-01-
31','YYYY-MM-DD')) tablespace quest_dat2,
  …
  partition lin_1999_12 values less than (to_date('1999-12-
31','YYYY-MM-DD')) tablespace quest_dat1,
--
  partition ord_max_val values less than (maxvalue));
```

QUEST SOFTWARE

# Or…




QUEST SOFTWARE®

# Did it Help?

```
SQL> select count(*) from h_lineitem_part where
l_receiptdate is null or l_receiptdate is not null
SQL> /

  COUNT(*)
----------
  59986052

1 row selected.

Elapsed: 00:00:55.93
```

An improvement of 36 seconds – that's 39% worse case!

QUEST SOFTWARE

# Without Parallel Query

```
SQL> select /*+ NO_PARALLEL (h_lineitem) */
  2*  count(*) from h_lineitem where l_receiptdate is null or
l_receiptdate is not null;

  COUNT(*)
----------
  59986052

1 row selected.

Elapsed: 00:03:51.21

SQL> select /*+ NO_PARALLEL (h_lineitem_part) */
  2* count(*) from h_lineitem_part where l_receiptdate is null or
l_receiptdate is not null;

  COUNT(*)
----------
  59986052

1 row selected.

Elapsed: 00:04:06.00
```

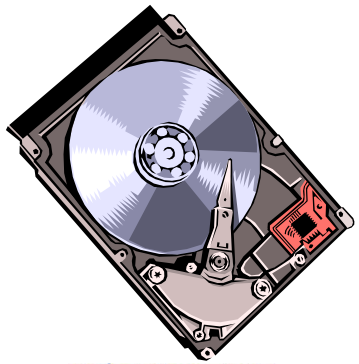
QUEST SOFTWARE

# Automatic Storage Management

ASM features allows for the automatic stripe-and-mirror everywhere

Allows for load balancing of the disk I/O sub-system

Removes the need for the DBA to specify physical file locations when allocating a tablespace



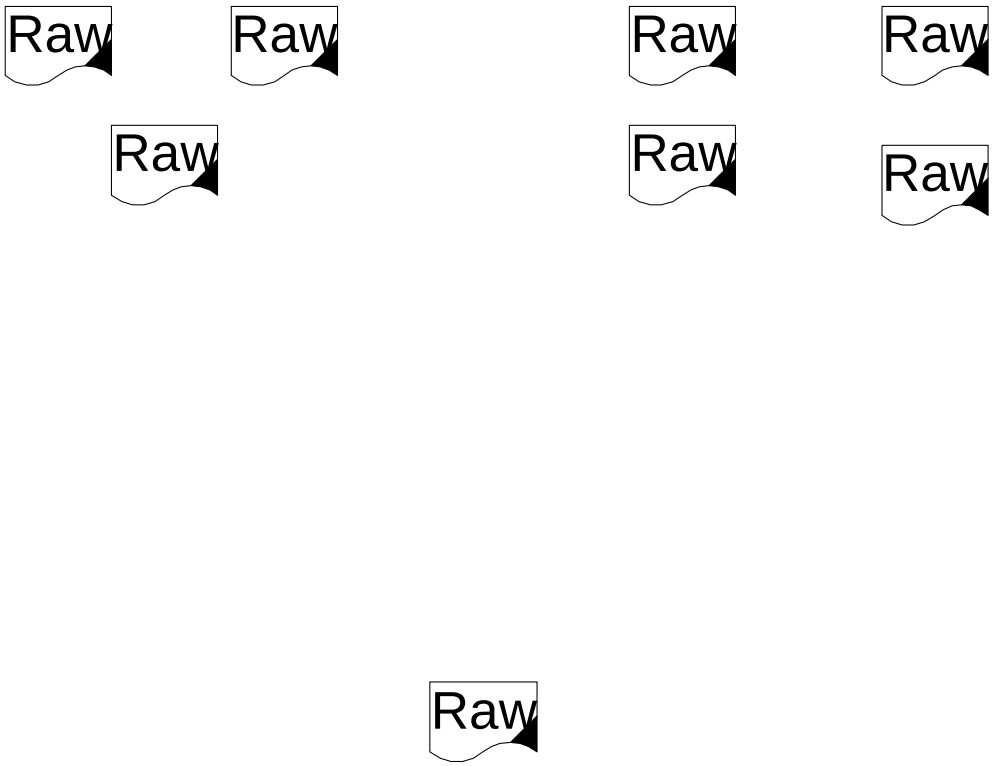
QUEST SOFTWARE®

# Automated Storage Management (ASM)

- Designed to simplify storage management
- ASM extends OMF (Oracle Managed Files)
- Adds disk mirroring and striping
- Can have both ASM and normal file management

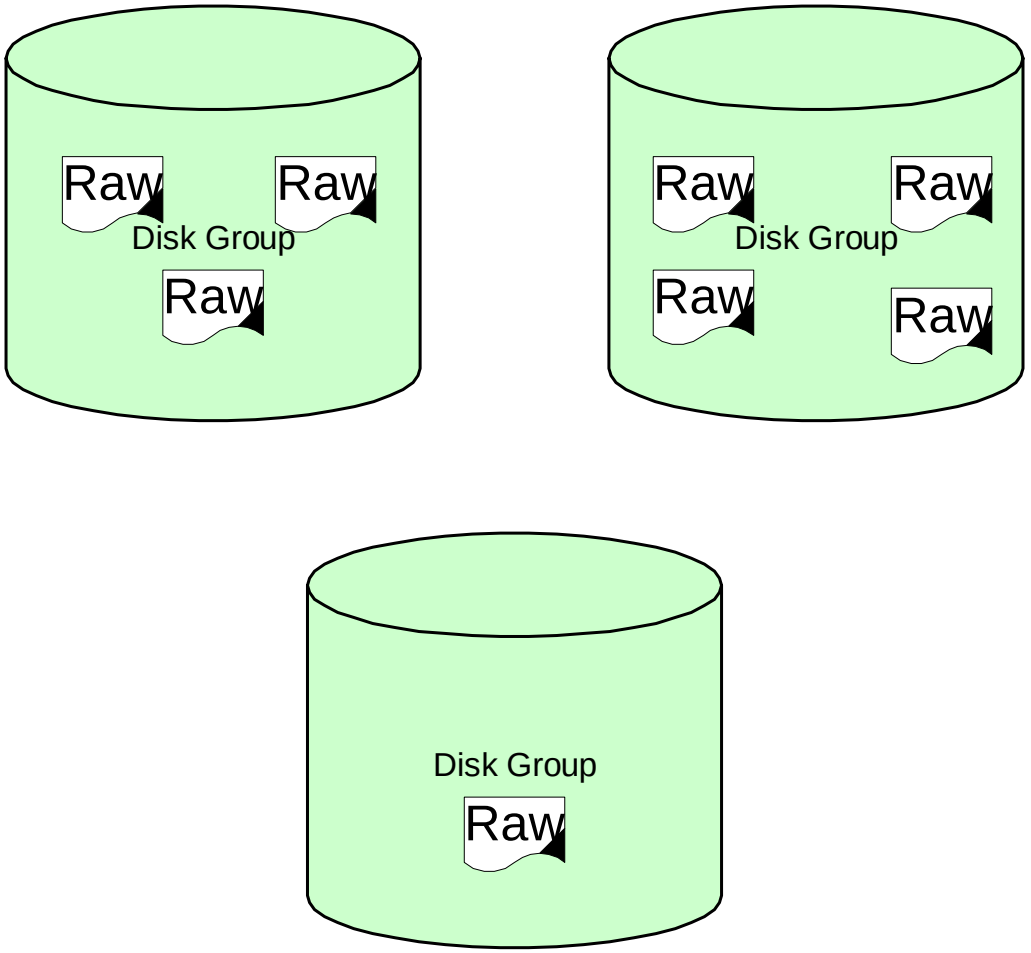
QUEST SOFTWARE®

# Automated Storage Management (ASM)



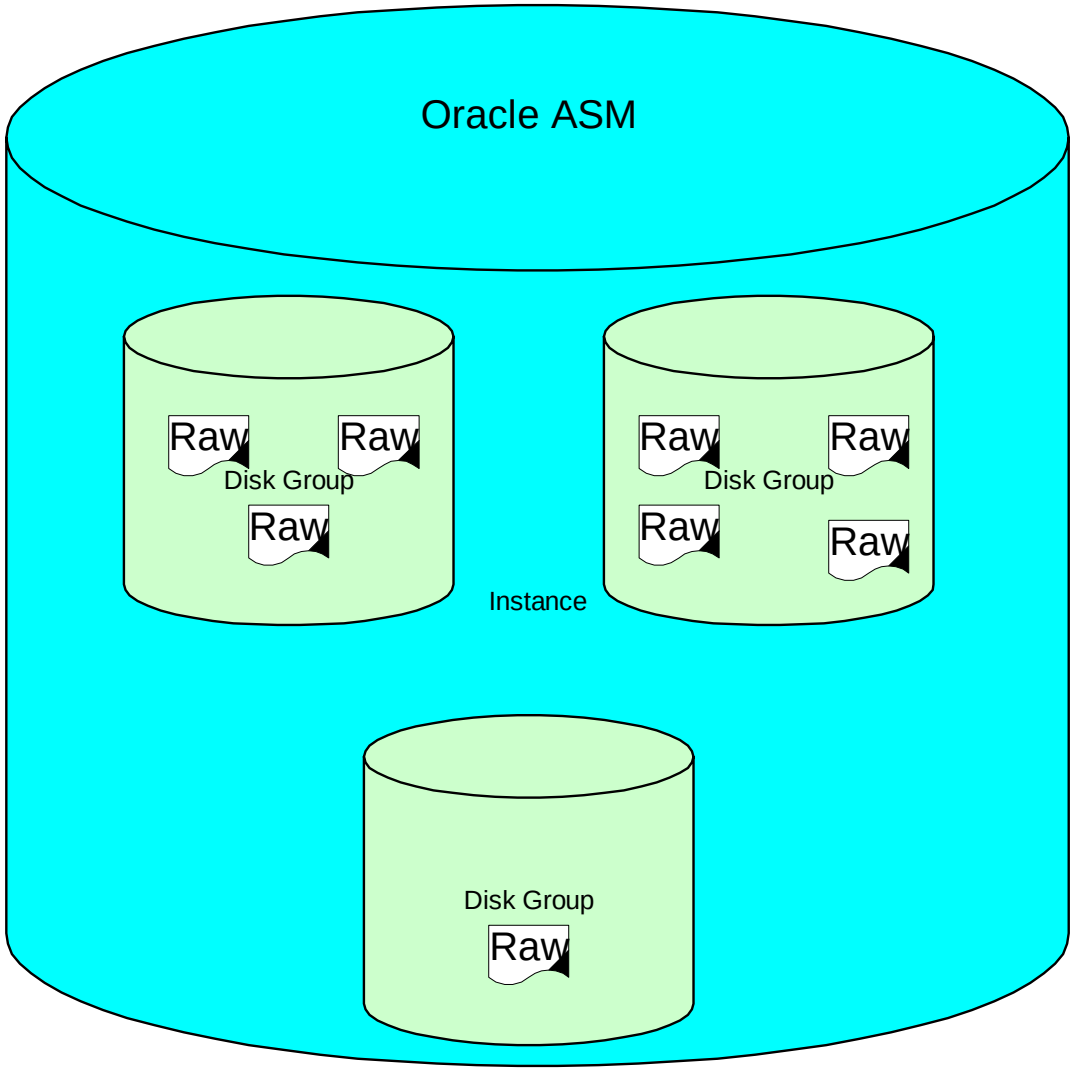

QUEST SOFTWARE®

# Automated Storage Management (ASM)



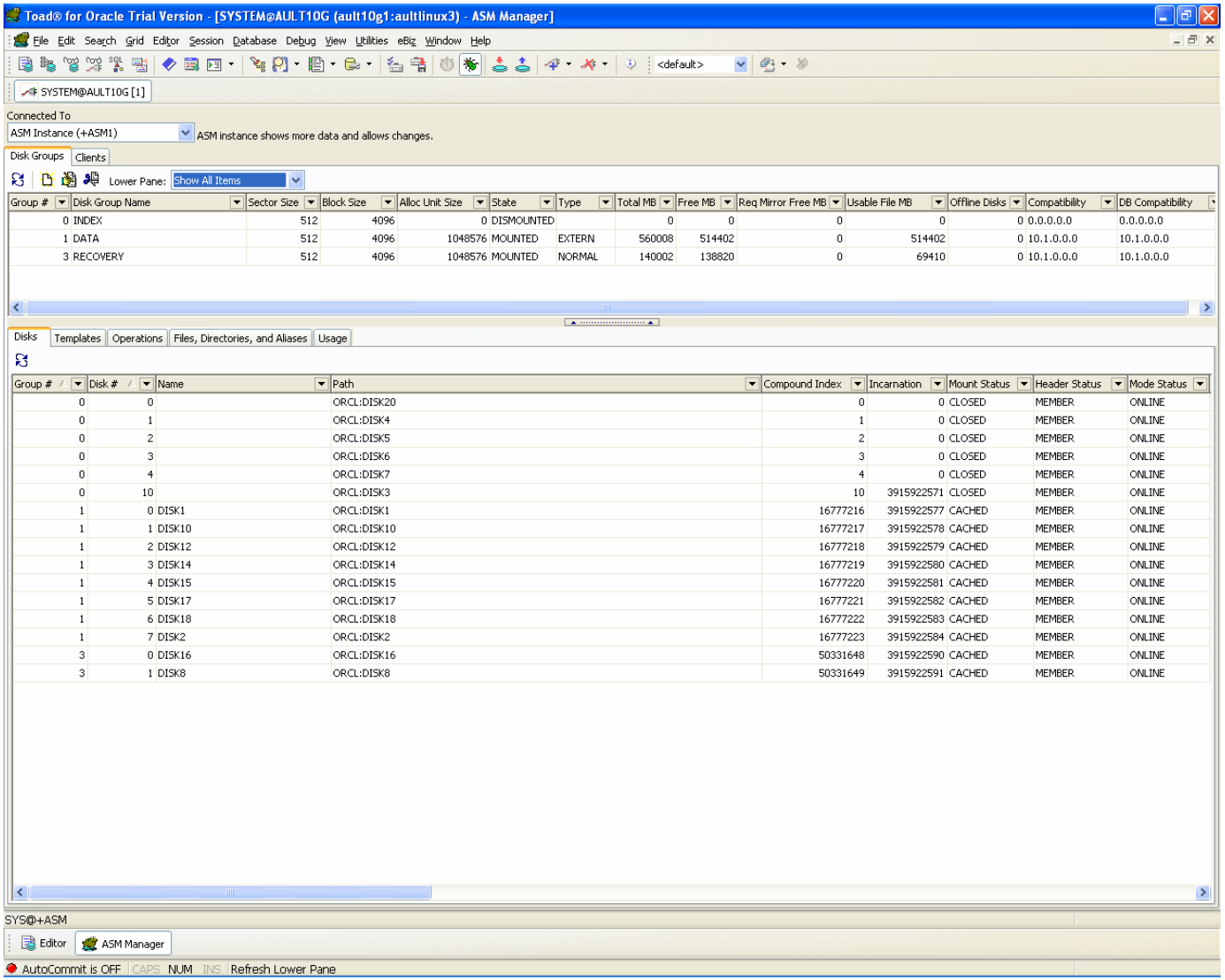

QUEST SOFTWARE

# Automated Storage Management (ASM)




QUEST SOFTWARE

# Actual ASM Instance

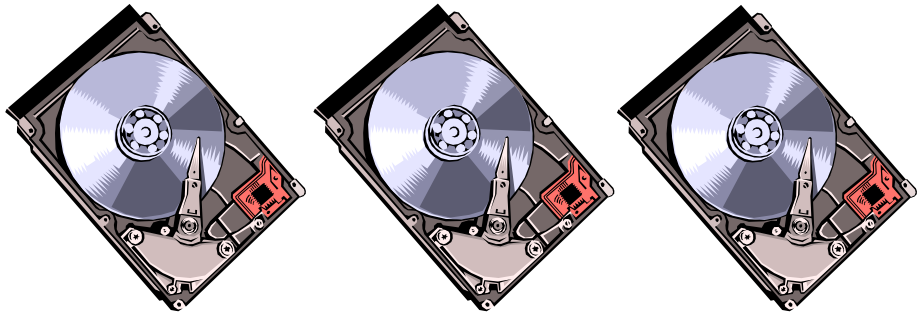



QUEST SOFTWARE®

# Automated Storage Management (ASM)

Usually Multiple Disk Groups are created:

- To group disks of different manufacturers, different sizes or performance characteristics.
- To group disks with different external redundancy together.
- To separate work and recovery areas for a given database.



QUEST SOFTWARE®

# Automated Storage Management (ASM)

- ASM breaks each file into multiple extents and spreads the extents for each file evenly across all of disks in a disk group.
- Once ASM disk groups are established, the Oracle database automatically allocates storage space from these disk groups for creating and deleting files.
- Rather than requiring a manually issued command as in previous versions, under ASM, unneeded data files are automatically deleted when they are no longer needed.
- ASM enhances database integrity for databases operating on disks that are not extremely reliable.
- In 10g ASM uses 1 meg AU for Data 128k AU for non-data
- In 11g ASM can use multiple AU in 1 meg multiple for Data, 128k for non-data

# Automated Storage Management (ASM)

There are three types of ASM disk groups:

- Normal redundancy
- High redundancy
- External redundancy

QUEST SOFTWARE®

# Automated Storage Management (ASM)

- Normal and High Redundancy -- The disk group template specifies the attributes for the ASM redundancy for all files in the disk group. Configuration of ASM high redundancy provides a greater degree of protection.
- External Redundancy -- ASM does not provide any redundancy for the disk group. In external redundancy the underlying disks in the disk group must provide redundancy (for example, using a RAID storage array.)

# Automated Storage Management (ASM)

The unit of storage for ASM disk groups is ASM Disks.

- An ASM disk corresponds to a Raw device. An ASM disk name is common to all nodes of the cluster.
- The disk name can be specified by the administrator, or it will be automatically generated by ASM when a disk is added to a disk group.
- Since different hosts can use different operating system names to refer to the same ASM disk, the ASM disk name abstraction is required.
- To reduce the chances of losing data in case of single disk failure, ASM provides mirroring.

# Automated Storage Management (ASM)

## Failure Groups

- Failure groups are manually defined groups of disks sharing a common resource.
- Failure groups definitions determine which ASM disks are used for storing mirror copies of data.
- Failure groups ensure that data and its redundant copy do not both reside on disks that are likely to fail together.
- Each set of failure groups is site-specific.
- Each disk is assigned to its one failure group in ASM by default.
- A failure group is maintained in a disk group, and multiple failure groups can exist within any given disk group.

# Automated Storage Management (ASM)

## ASM Instances

- The ASM instance must be configured and running for the database instance to access Automated Storage Management files.
- ASM instances are not used to mount databases, they are simply used to coordinate data layout for database instances.
- Multiple, separate database instances can share ASM disk groups for their files.
- In a Real Application Cluster (RAC) environment, there is typically one ASM instance on each node in a RAC configuration.
- All ASM management commands are directed through the ASM instance.

# Automated Storage Management (ASM)

**ASM Instance Background Processes**

- RBAL - coordinates rebalance activity for disk groups
- ORB0, ORB1… - These perform the actual rebalance data extent movements. There can be many of these at a time

QUEST SOFTWARE®

# Automated Storage Management (ASM)

**Database Instance ASM Background Processes**

- Any database using an ASM disk group will contain a background process called OSMB.
- The OSMB process is responsible for communication with the ASM instance.
- A second additional background process, called RBAL (just like in the ASM Instance) performs a global open on ASM disks.
- A global open means that more than one database instance can be accessing the ASM disks at a time.

# In Summary, Points to Take Away

- Block size can make a performance difference
- Full scan indexes should be re-organized periodically if they are prey to white space
- There is no such thing as an unbalanced index (in Oracle anyway)
- Partitioning usually in concert with parallel query can help performance
- ASM helps virtualize storage making it easier to manage and utilize

QUEST SOFTWARE®

# Questions?

QUEST SOFTWARE®