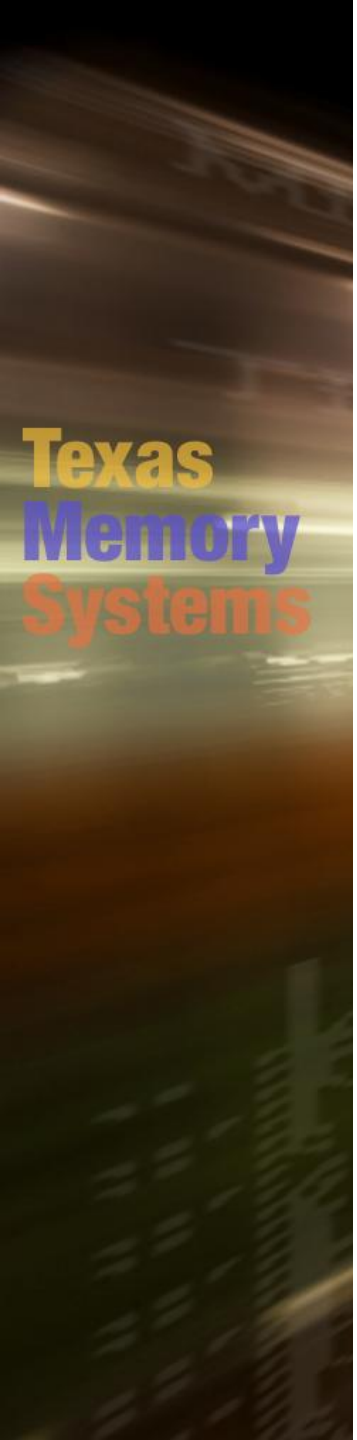
Holistic Oracle Tuning

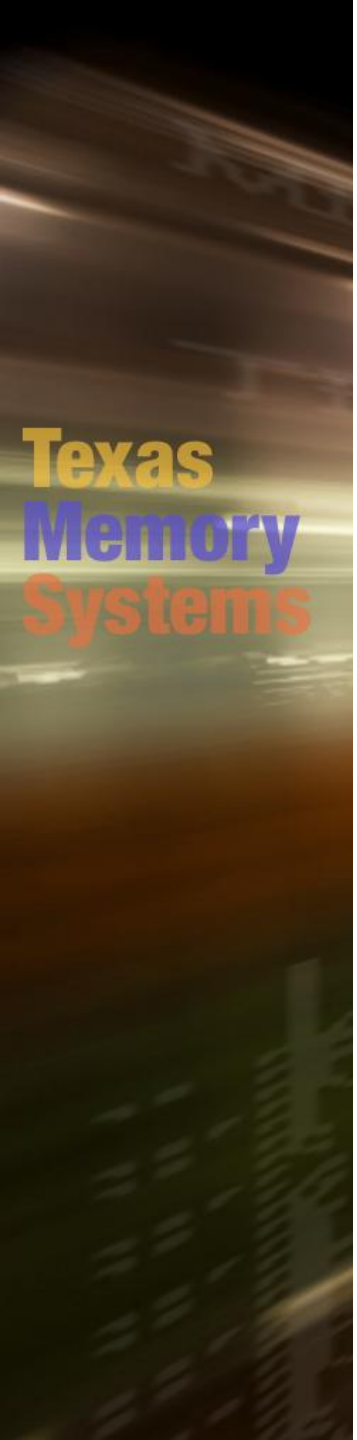
Seeing the forest in the trees

Mike Ault

Oracle Guru, TMS, Inc



Introduction



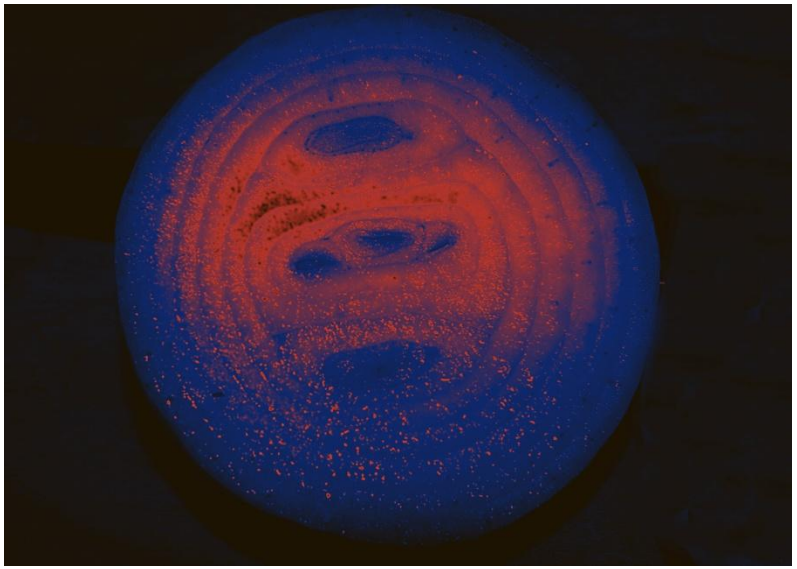
Holistic

Holistic:
1 : of or relating to holism
2 : relating to or concerned with wholes or with complete systems rather than with the analysis of, treatment of, or dissection into parts <holistic medicine attempts to treat both the mind and the body> <holistic ecology views humans and the environment as a single system>

Texas Memory Systems

Oracle Holism

- Oracle is like an onion
- When you dig into it, it makes you cry
- Actually, I meant to say it has many layers…



Texas Memory Systems

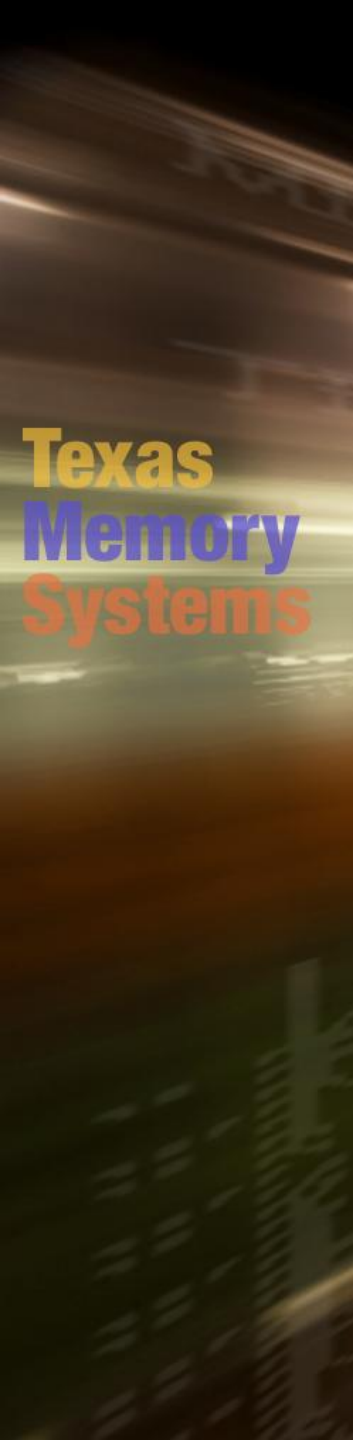
Oracle's Many Layers

- Physical – IO subsystems and the user read and database write processes
- Memory – caches, pools, latches
- Kernel – All those wonderful Oracle processes
- Structure – Tables, indexes, views, LOBs, etc
- Code – SQL, PL/SQL, JAVA, C+, D- (oops),
- Middleware – Oracle's and third party
- Network – SQLNet
- RAC – (let's not go there…)

Texas Memory Systems

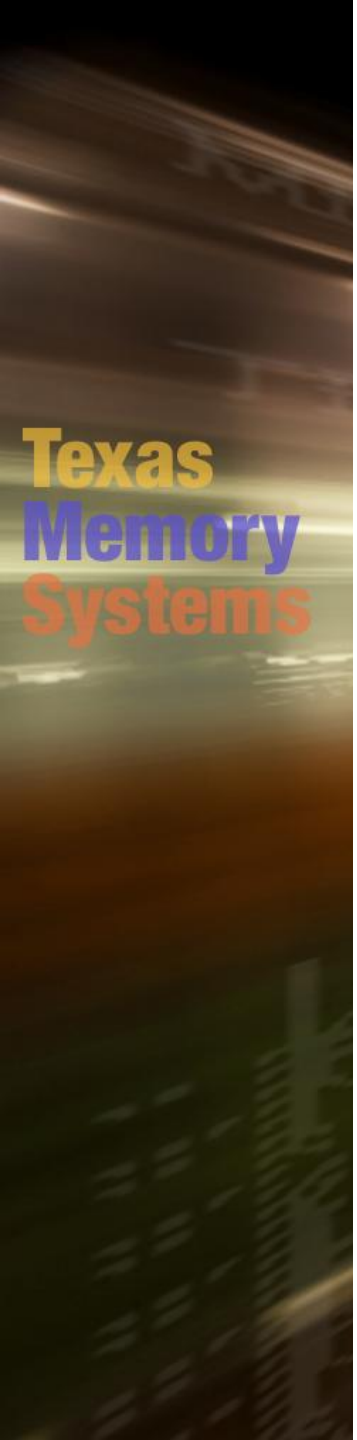
Oracle the Onion

- Each layer rests on the one underneath
- If any layer is “rotten” then the entire structure is in danger.
- Likewise, a change to any layer may propagate effects up, and down through other layers.



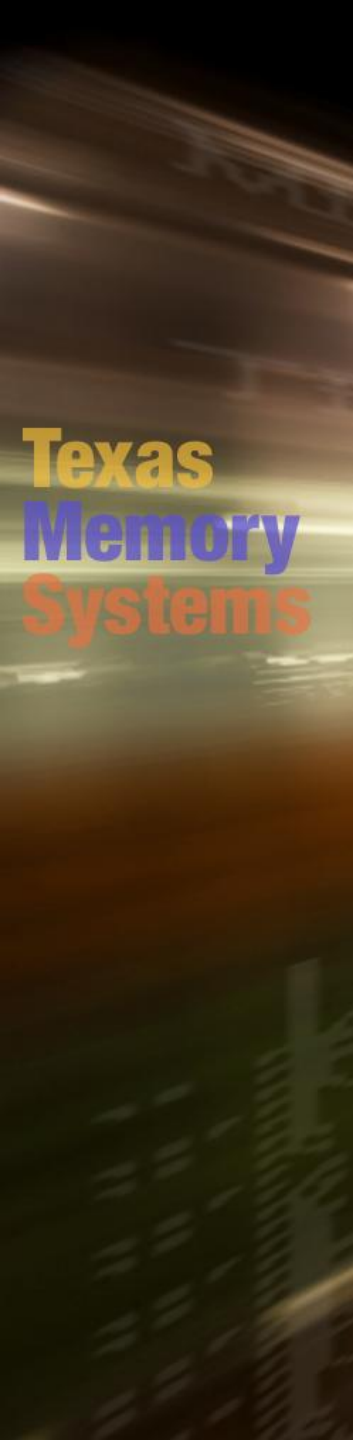
Structure Layer Example

- **Given:** Statement X is generating excessive physical IOPS when it executes
- **Local Fix**: Add indexes
- **Immediate local affect**: Statement X runs faster with fewer IOPS
- **Holistic effects**: Statements Y and Z switch from proper indexes for them to indexes added for statement X
- **Holistic Result**: X runs faster, but Y and Z run slower, overall result: Slower system even though the tuning effort on X was a success!
- **Act locally, think globally!**



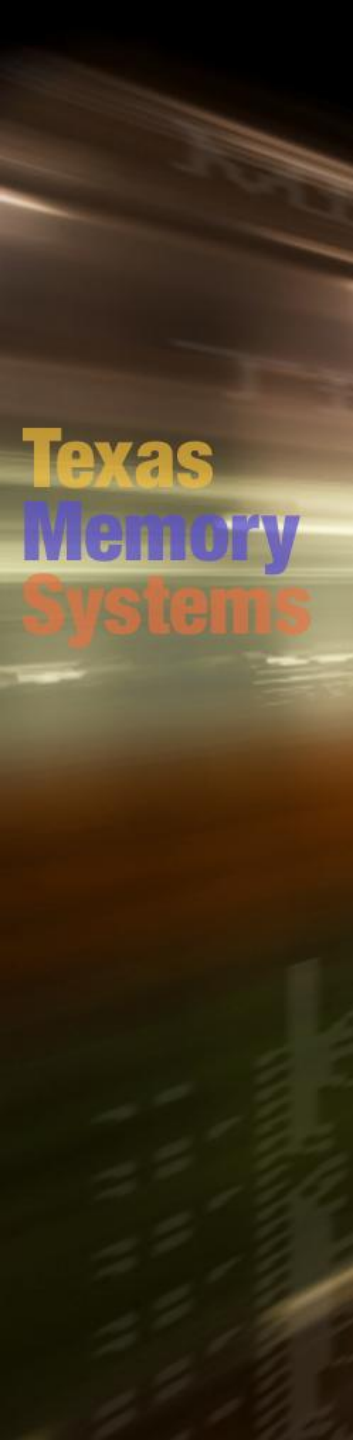
However…

- Not all local actions have a negative affect
- A switch to a new index is because the optimizer has looked at statistics
- However, we all know this may not always work as designed!



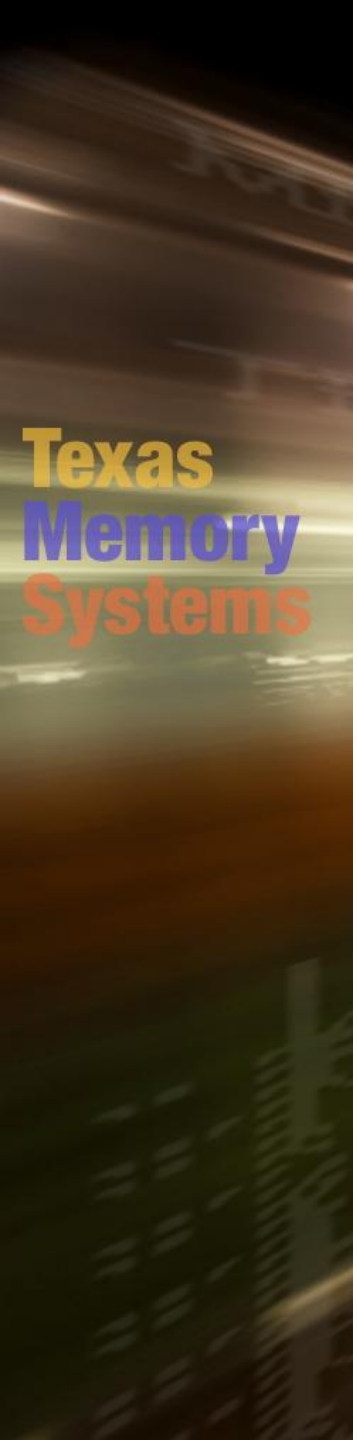
Memory Example

- Generally caching is a good thing
- The more we cache the faster the application will run.
- This comes at a cost.
  - Latches
  - Enqueues
- Now IO has to be handled by the CPU using logical IO.
- Logical IO is faster than physical IO
- Except if it introduces addition CPU cycles and the system becomes CPU bound.



Seeing the Forest

- Without enough IO bandwidth, adding memory or CPU to a system that is IO bound may not help at all.
- We have to understand the holistic view to take the proper course of action.

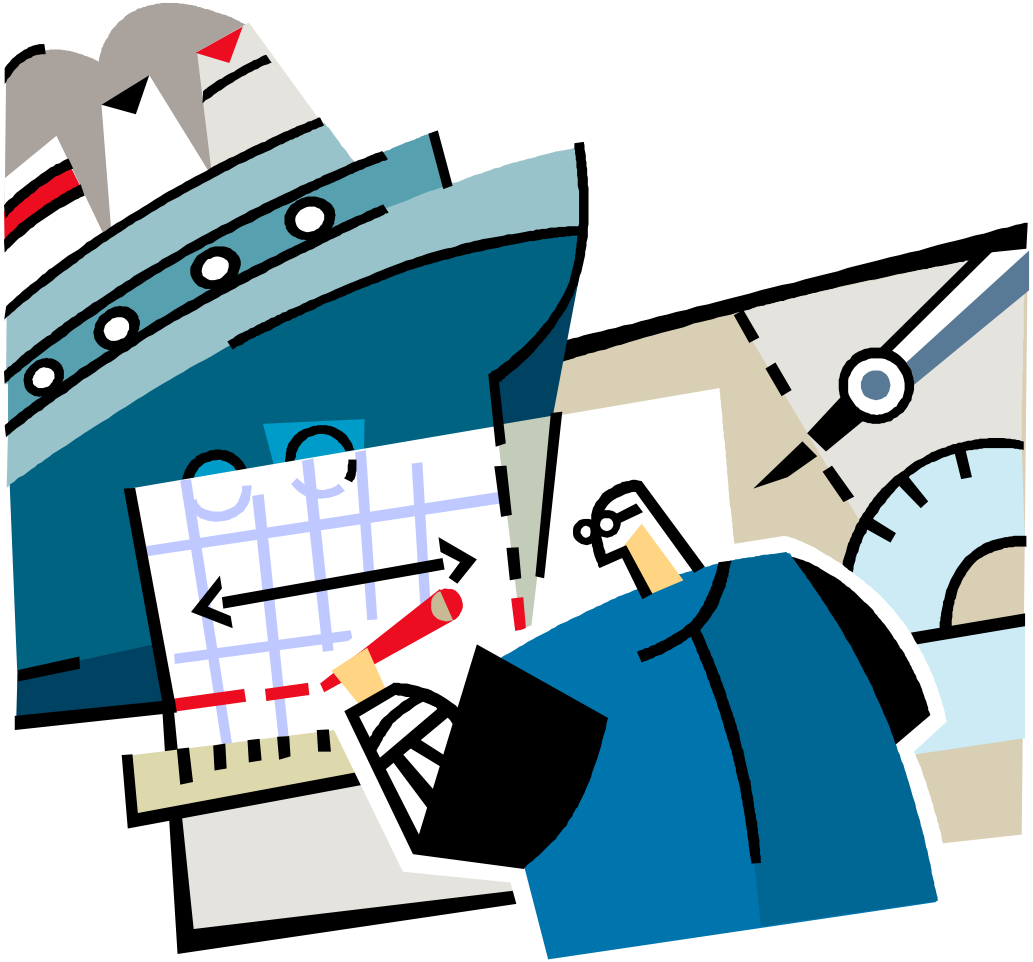


Local Actions with Negative Holistic Affects

- **Structure Layer**
  - **Bad table design**
  - **Bad indexes**
  - **Insufficient Indexes**
- **Code Layer:**
  - **No Bind Variables**
  - **Use of Distinct**
  - **Insufficient Joins**
  - **Bad joins**
- **IO Subsystem**
  - **Low Capacity IO Subsystem**
  - **Improperly configured IO Subsystem**
- **Memory Layer**
  - **Insufficient Memory Resources**
  - **Improperly configured memory**
- **CPU Layer**
  - **Insufficient CPU Resources**
- **Middleware/Application Layer**
  - **Bad code generators**
  - **Improperly Configured Middleware/software**

Texas Memory Systems

Structure Layer

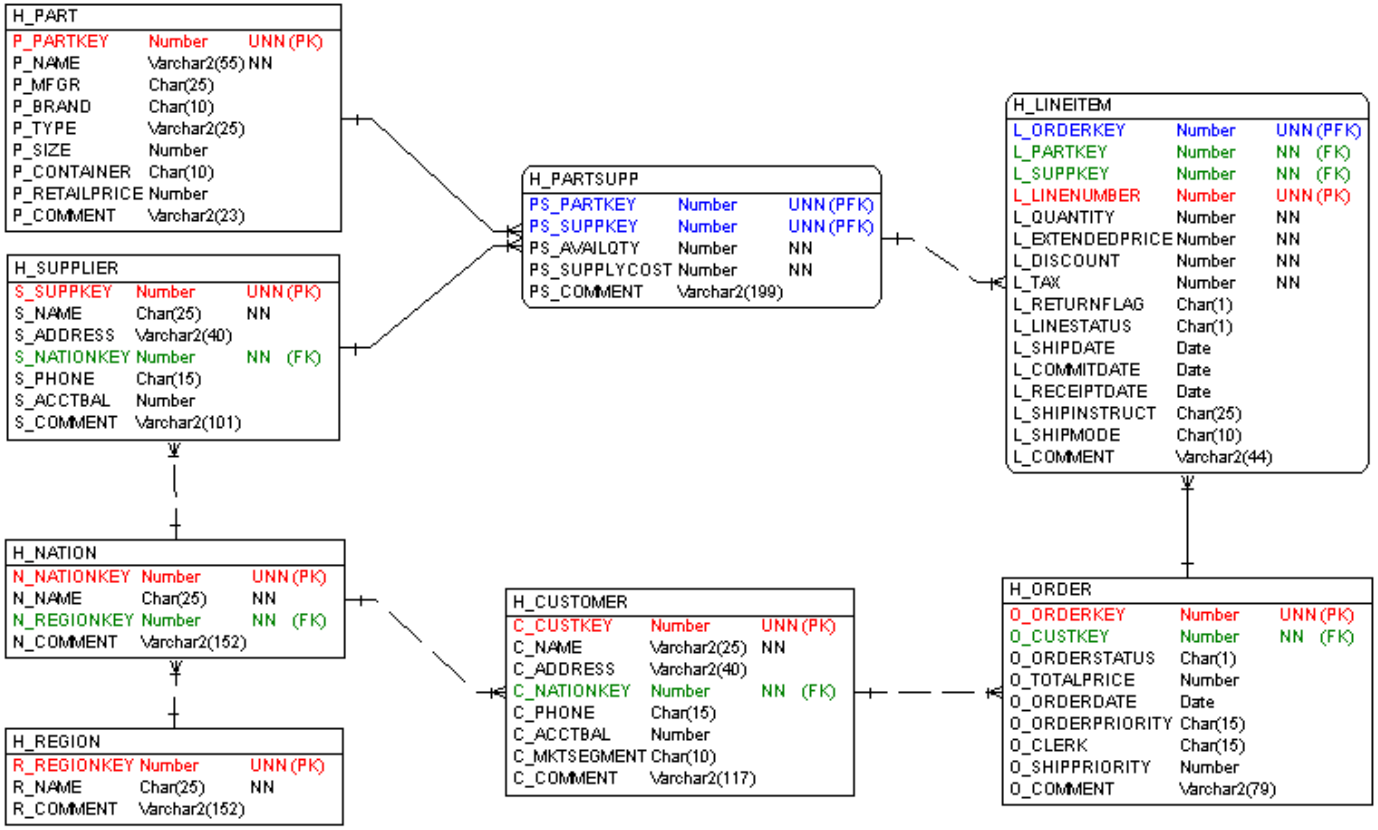


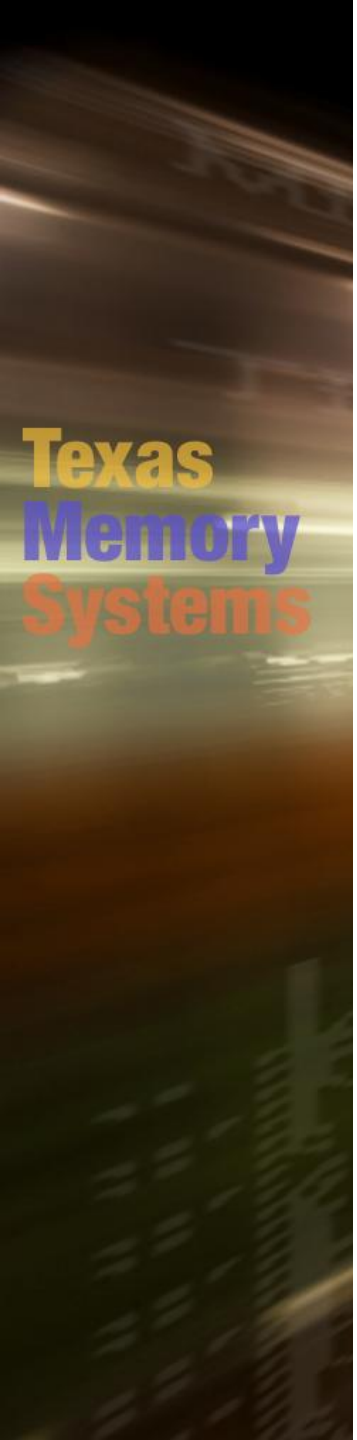
Structure Layer

- Internal physical structures that make up the database.
- Was just tables and indexes
- Now several types of tables (IOT, External, Global Temporary)
- Now several types of indexes (B-tree, bitmap, concatenated, bitmap join)
- Materialized views (Full, on demand, fast, on commit)
- LOBs (BLOB, CLOB, NCLOB)
- Objects (types, methods, varrays, etc)
- Many other structures

Texas Memory Systems

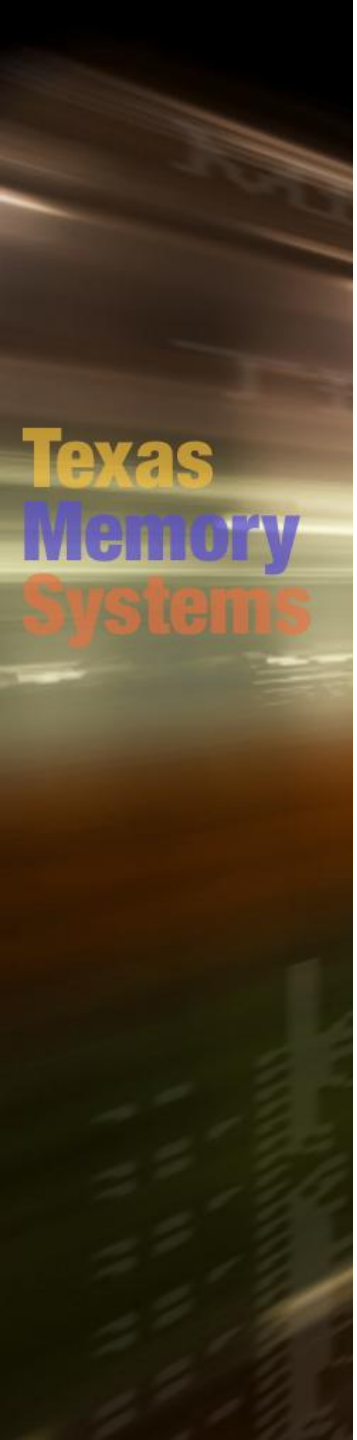
Tables





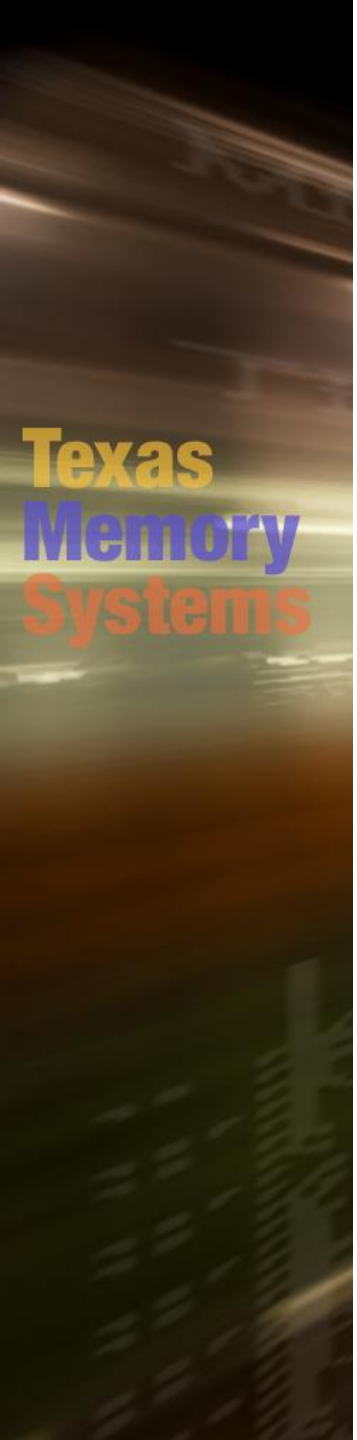
Improperly Designed Tables

- Under normalized
- Over normalized
- Improper data types
- Recursive tables



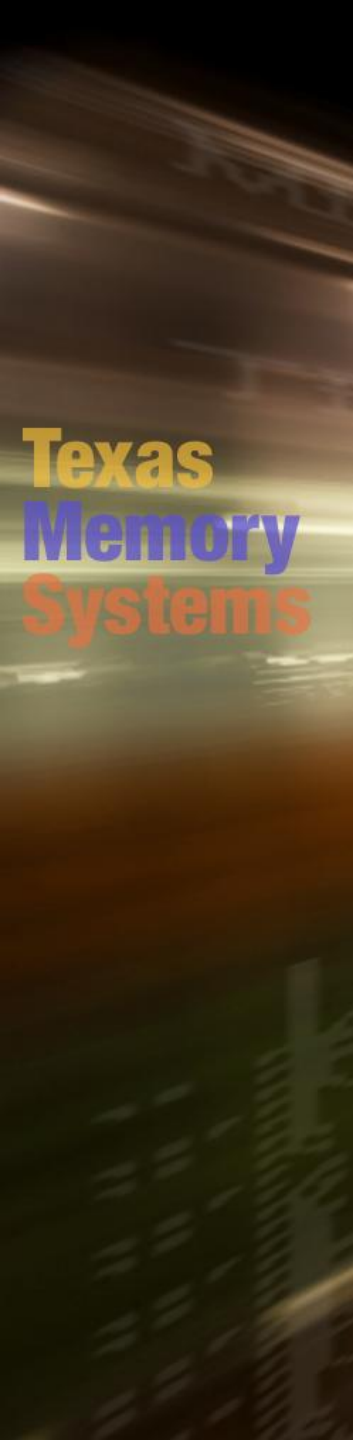
Under Normalized

- Under normalized has repeating values
  - A contact list where the address is stored with each contact
  - In companies with multiple contacts, the address is stored multiple times
  - Exception – denormalized reportign materialized views, fact tables in DWH



Over Normalized

- Over Normalized
  - 4th Normal form
  - Boyce-Codd 5th Normal form
  - Even 3rd normal in some cases
- Too many joins, if every query requires 4-5 or more joins, you are over normalized
- Exception – data warehouse with fact and dimension tables
- Oracle uses _optimizer_max_permutations of 2000, limits number of pathways (6-7 tables)



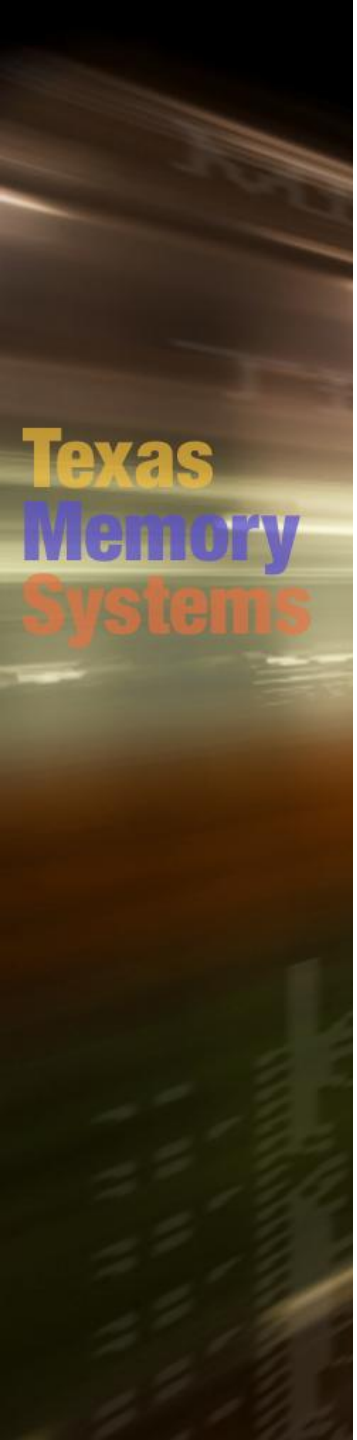
Improper Data Types

- Using one size fits all (VARCHAR2 unnamed columns)
  - Forcing self-joins
  - Depending on other columns to determine what type a given column should be
  - Used in “flex-field” designs
- CHAR for VARCHAR2
- CHAR or VARCHAR2 for NUMBER
- CHAR or VACHAR2 for DATE

Texas Memory Systems

Recursion in Tables

- Bill-of-materials problem
  - Assembly is made of other assemblies that are made of parts
- Typified by “ears” on ERD diagrams
- Can sometimes be corrected by proper normalization
- Also caused by “flex-fields”

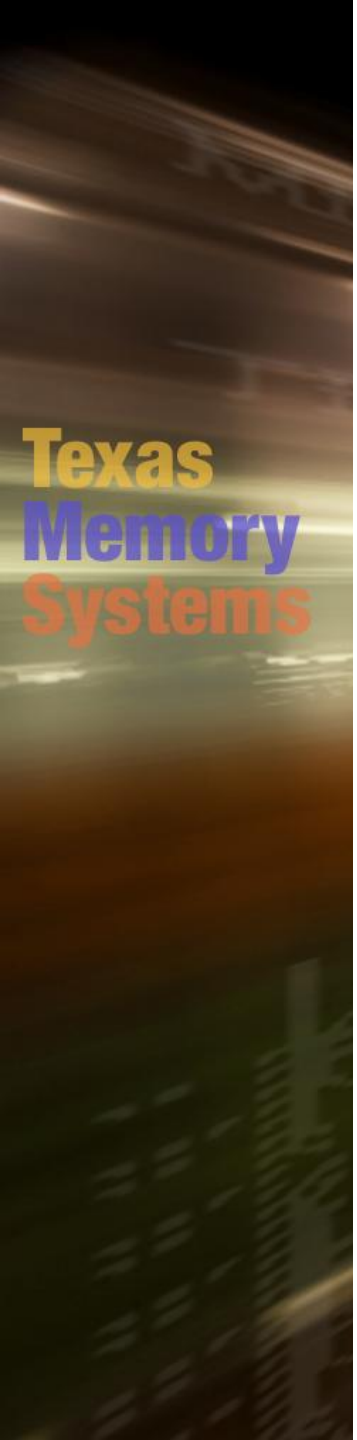


Improper Table Use

- Using normal tables as temporary tables (intermediate report tables)
  - Use global temporary tables instead
- Using too complex ETL
  - Use external tables instead
- Using IOT improperly
  - Not good for non-primary key reads
  - Not good in high DML use

Texas Memory Systems

Indexes



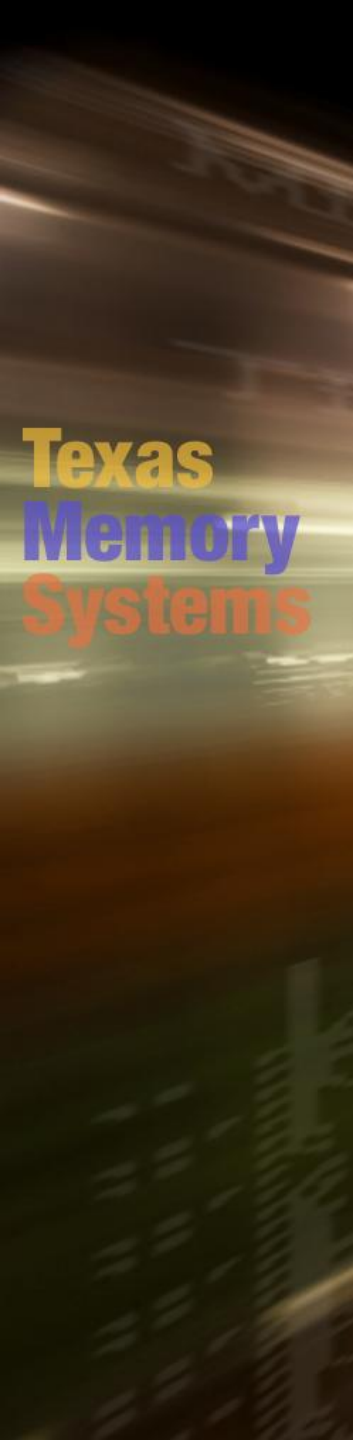
Improperly Implemented Indexes

- Insufficient indexes
- Too many indexes
- Incorrect Index Use

Texas Memory Systems

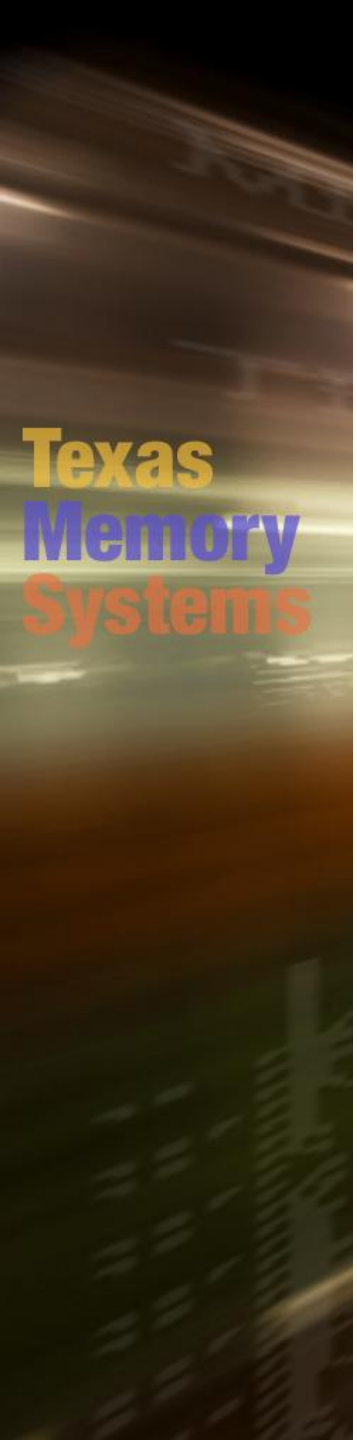
Insufficient Indexes

- Typified by too many full table scans
  - Large numbers of physical reads
  - Large numbers of db_file_scattered_read waits
  - Look at v$sqlplan table
  - Index columns used in joins
  - Use function based indexes on columns searched by functions



Too Many Indexes

- Typical in third party apps
- They don’t know what to index, so they index everything
- Drives up cost of DML
- Again use v$sqlplan to see what is used
- Also look at index monitoring
- Index join columns and searched columns only



Incorrect Indexes

- Use btree for high DML high cardinality: DATE
- Use bitmap for low DML low cardinality: SEX
- INSERT ok for bitmap, DELETE and UPDATE bad.
- Same constraints on bitmap-join indexes as are for bitmap indexes
- Multi-column indexes must be designed properly
  - Proper order
  - Excessive skip-scan searches on multi-column indexes indicates order issues.
- Low selectivity indexes (frequent full scans)

Texas Memory Systems

Materialized Views

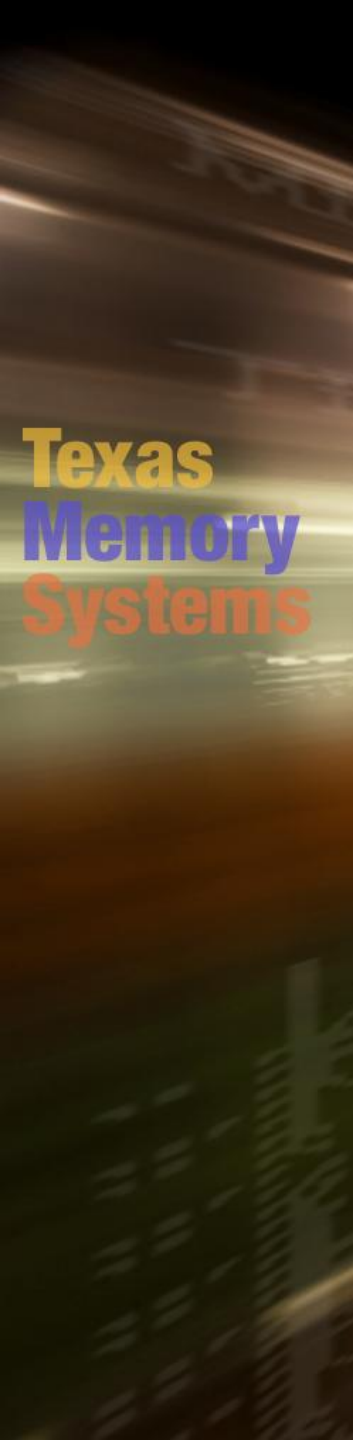
Texas Memory Systems

Materialized Views

- Were called snapshots
- Very under-utilized
- Use for reports, summaries, intermediate result tables
- Gives the impression of instant response
- Can be ON-DEMAND, or ON COMMIT
- Can be built on pre-built tables (partitions)
- Can be independently indexed
- Automatically recognized by the optimizer

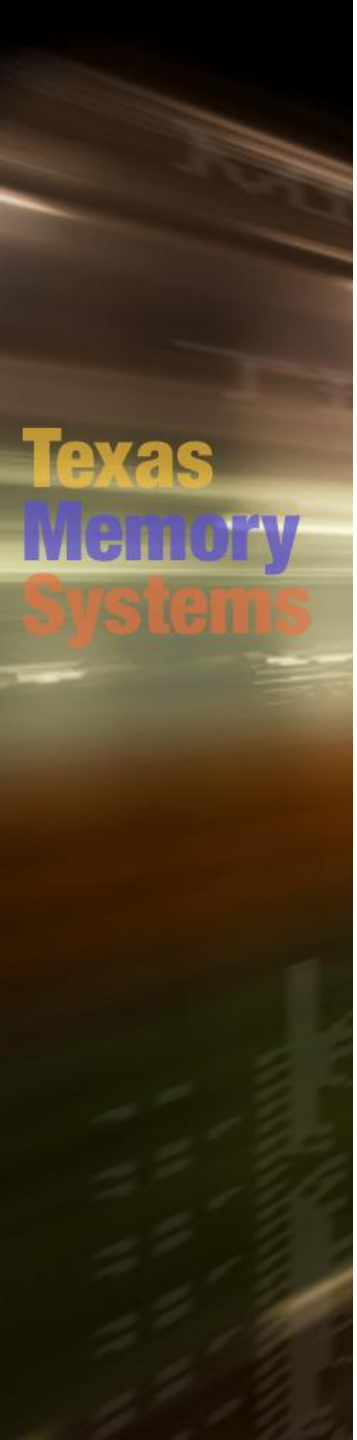
Texas Memory Systems

Other Physical Layer Items



LOBS

- Large Objects
  - Table reference
  - Actual LOB
  - LOB Index
- Avoid in-line storage
  - Places index with data
- Properly configure LOB storage
- Do not store with main data

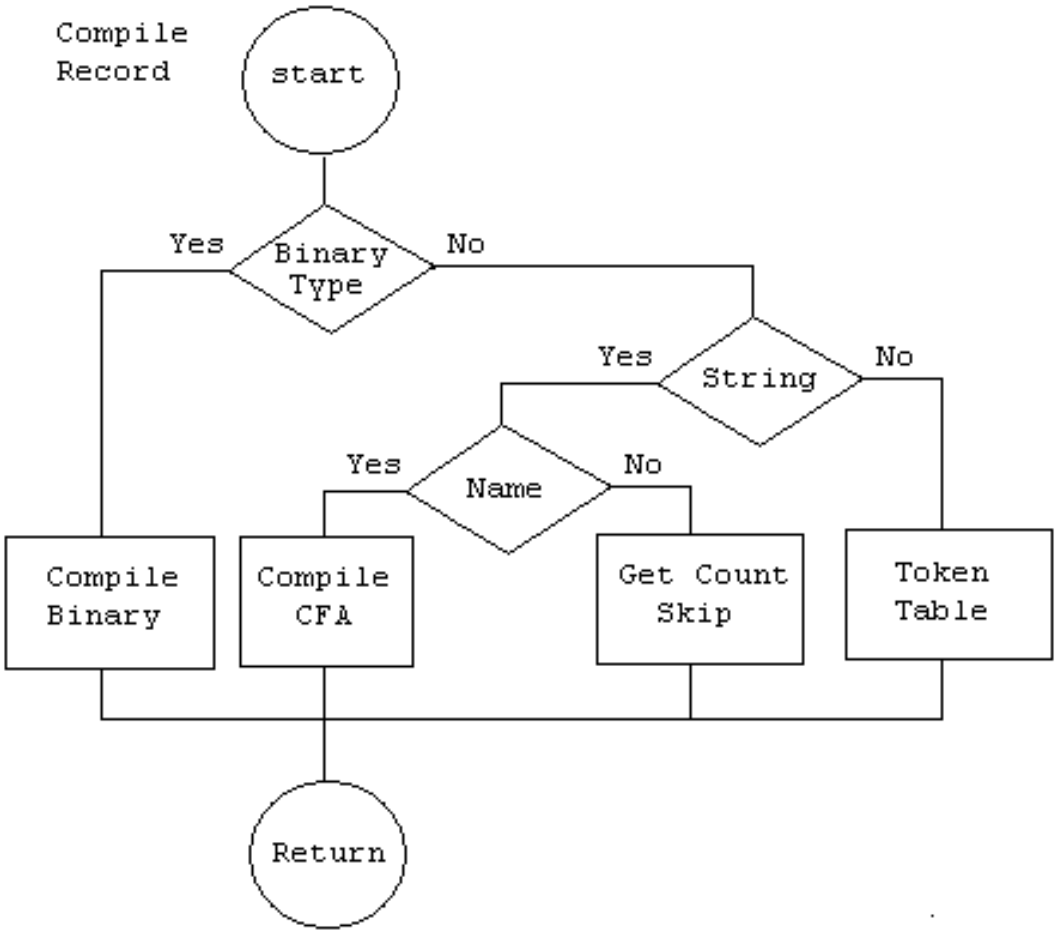


Objects and Methods

- Objects are VARRAYS, TYPES
- METHODS tell Oracle what to do with them
- Stored as inline/out of line LOBS
- Rarely perform as well as normal tables
- Use sparingly

Texas Memory Systems

Code layer



Texas Memory Systems

Code Layer Issues

- Restricted to SQL
- Each other code base have general problems as well
  - JAVA
  - C
  - C+, C++, C#
  - etc

Texas Memory Systems

No Bind Variables

- Probably the number one code related issue in SQL
- Allow a statement to be reused.
- Without them identical statements (identical except for literals) result in new version of the statement in shared pool
- The optimizer has to re-evaluate the statement causing recursive SQL.

Texas Memory Systems

Indications of Bind Variable Issues

- Large shared pool (sometimes larger than the db cache area)
- High recursive SQL
- High use of shared pool memory
- Low reuse for statements with greater than 1 execution
- And of course, literals instead of bind variables in the code

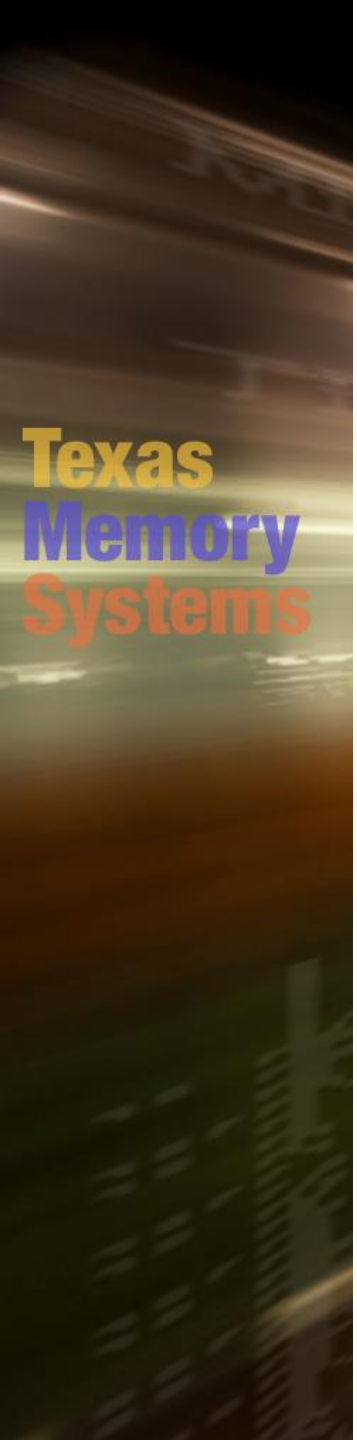
Bind Variable Issues

Cache Sizes

	Begin	End		
Buffer Cache:	640M	592M	Std Block Size:	8K
Shared Pool Size:	1,168M	1,216M	Log Buffer:	14,384K

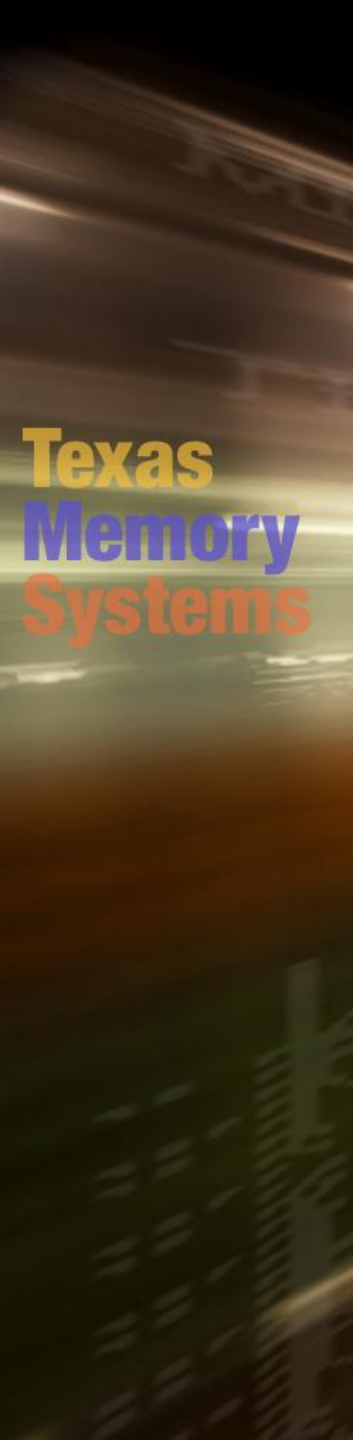
Load Profile

% Blocks changed per Read:	0.35	Recursive Call %:	33.43
Rollback per transaction %:	2.02	Rows per Sort:	5.45



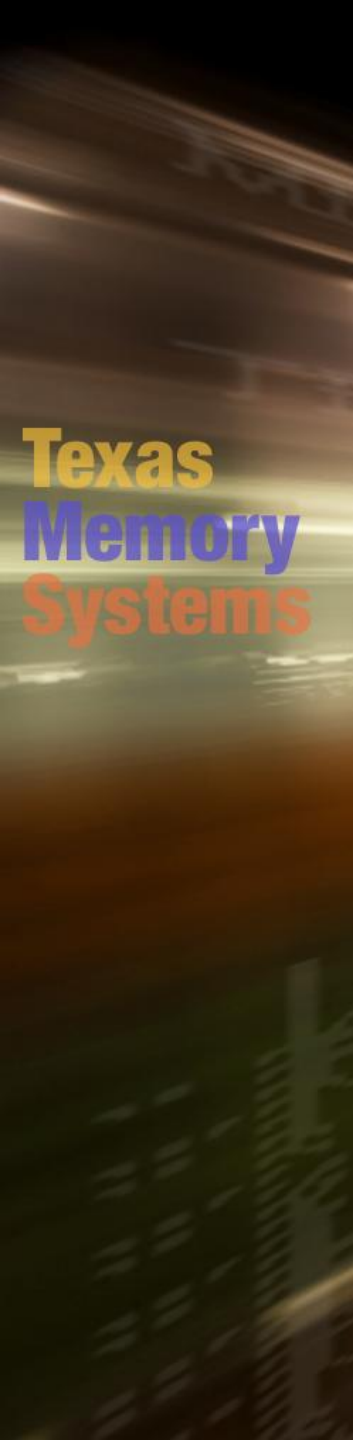
Fixing Bind Variables

- quick band aid set the initialization parameter CURSOR_SHARING to SIMILAR or FORCE
- Ultimate fix is to correct the bad coding and teach good bind variable usage going forward.



Improper Use of DISTINCT

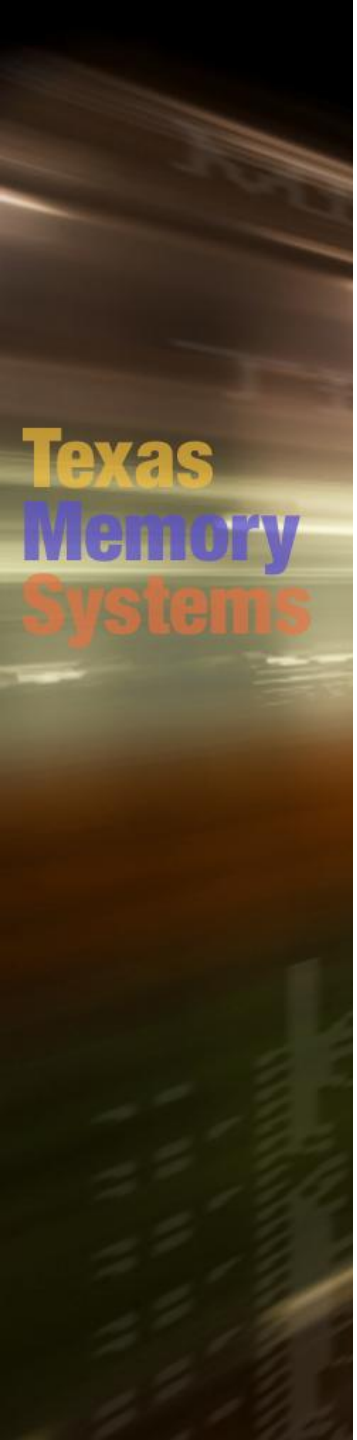
- Usually a quick code fix
- What performs in a 100 row test environment will fail in a 10,000,000 row real environment
- Causes sorts and CPU cycles
- Affects CPU, Memory, and IO Subsystem



Improper Use of DISTINCT

Elapsed Time (s)	CPU Time (s)	Executions	Elap per Exec (s)	% Total DB Time	SQL Id
414	25	7,382	0.1	30.5	g7a19b1wmsvgq
SELECT DISTINCT …					
323	34	4,048	0.1	23.8	50gk22jk74jvy
SELECT …					
183	24	1,666	0.1	13.5	c1ysvka1vfyfd
SELECT …					
117	9	4,809	0.0	8.6	c9humn05ydj4q
SELECT DISTINCT …					
60	3	678	0.1	4.4	5c4kar29763mv
SELECT DISTINCT …					
30	1	1	30.2	2.2	2f33rza4xwrk7
SELECT …					
15	1	351	0.0	1.1	b6pdf4vxgtgr7
SELECT DISTINCT …					

So in the above example of the top 7 statements for elapsed time, 4 used a SELECT DISTINCT.



Indications of Improper DISTINCT Usage

- Large numbers of sorts
- Large numbers of sort rows
- WHERE clause evaluation.
- If there X tables in a query then need X-1 join conditions

Texas Memory Systems

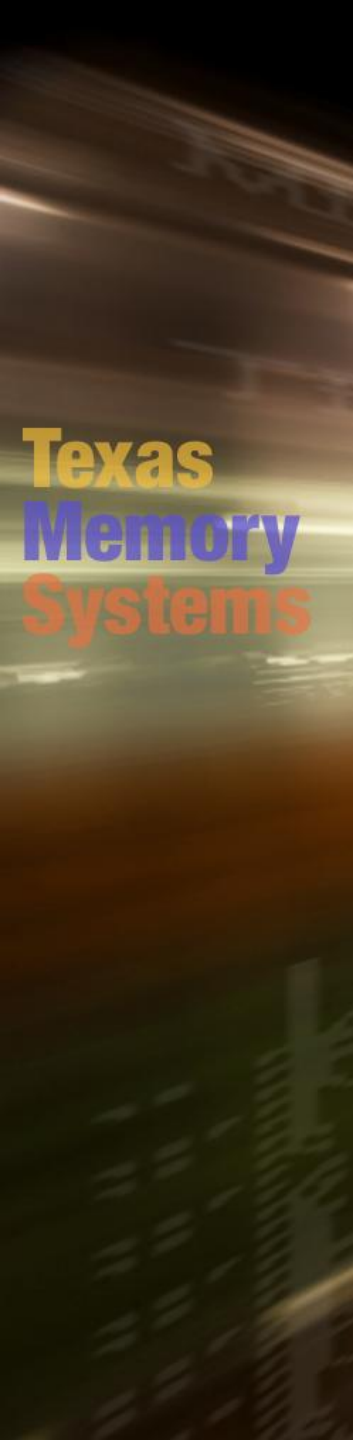
IO Subsystem

=

Texas Memory Systems

IO Subsystem Layer Issues

- Basically two issues
  - Capacity
  - Configuration



IO Capacity

- What exactly are we talking about?
- Storage capacity is only one facet
- You have to consider IOPS as well
- If you size for IOPS usually you will be over specified for storage capacity *with disks*

IO IOPS Capacity Issue

Tablespace IO Stats DB/Inst: TEST/test Snaps: 21560-21658
-> ordered by IOs (Reads + Writes) desc

Tablespace

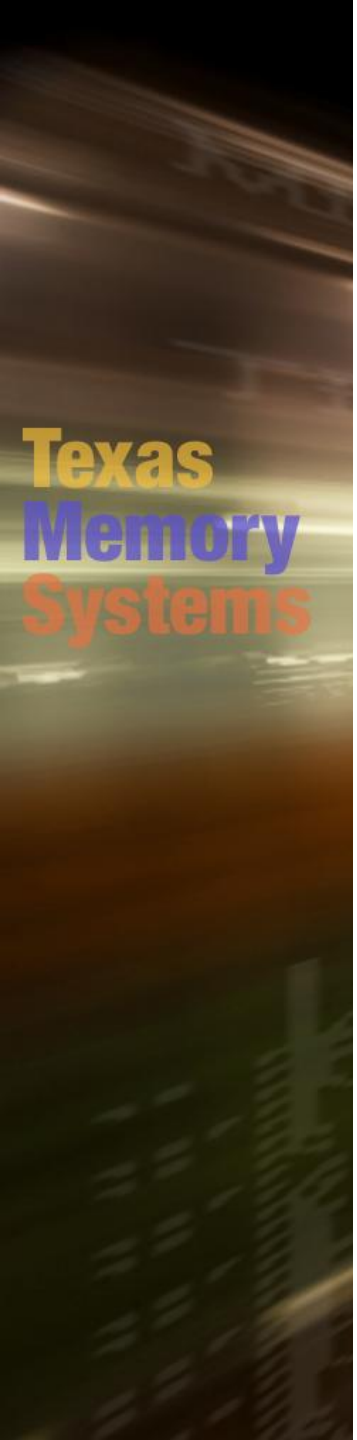
	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
TEST2_SSA_ZIP_AD								
	16,897,918	48	13.7	1.0	15,433,132	44	27	7.4
TEST2_SSA_ST_AK								
	16,700,222	47	14.0	1.0	15,314,651	43	67	5.2
TEST2_SSA_ST_SN								
	16,573,311	47	11.5	1.0	15,322,214	43	798	2.7
TEST2_SSA_ST_HN								
	16,306,046	46	14.2	1.0	14,916,357	42	0	0.0
TEST2_SSN01_ID								
	16,087,914	46	13.1	1.0	14,781,609	42	2	15.0
TEST2_PKT								
	18,200,932	52	9.6	1.0	11,313,171	32	0	0.0

- Notice latency is >5ms
- Notice Buffer Waits – usually indicates write time issue
- Notice these times are for single block reads!

Texas Memory Systems

Why?

- Obviously overloaded disks (200 IOPS/disk)
- For maximum IOPS and lowest latency use striping
- Only utilize a maximum of 30-50% of each disk
- Many say only 20-30% of a disk should be used
- This is called short-stroking the disk
- This means that you must buy at least 3-4 times the amount of disks than you need for storage capacity



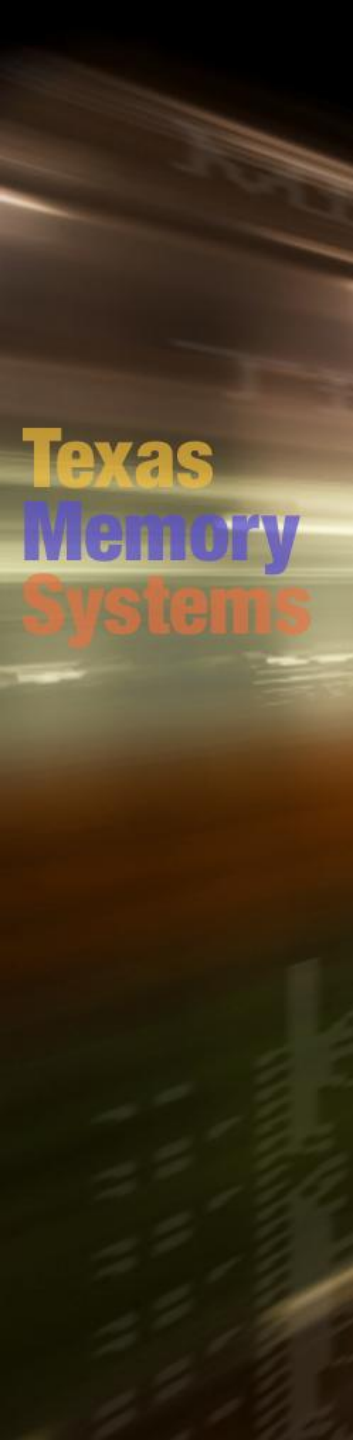
Disk Issues

- Rotational and Seek latency
- Only one request per disk at a time
- Controllers optimize disk access but are still limited
- This why, regardless of storage capacity, disks are limited to 200 IOPS
- Note that with multiple users, the number disks must go up

Texas Memory Systems

What Can be Done?

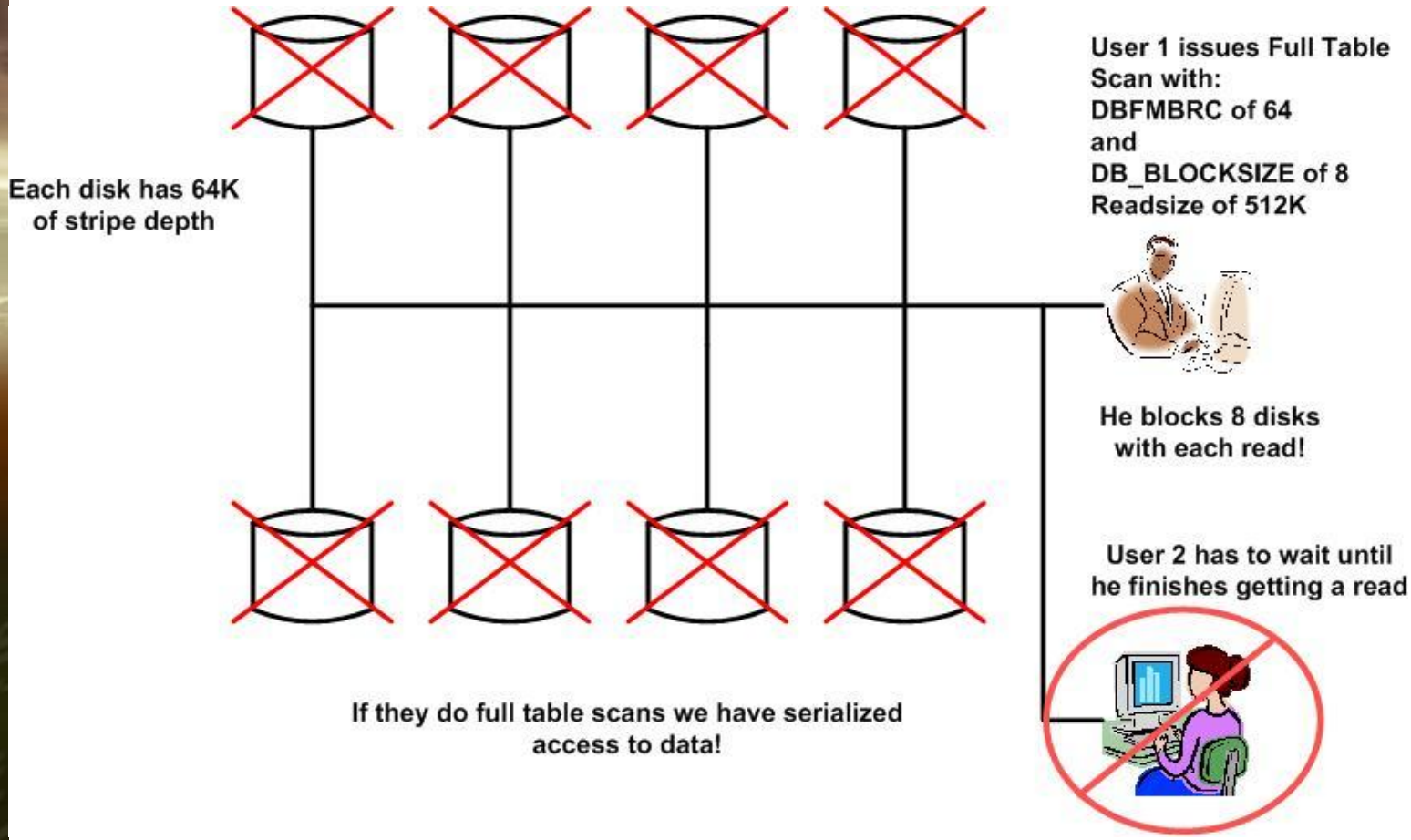
- Inexpensive SSD technology (when measured in terms of $/IOPS or storage density $/IOPS/GB)
- Eliminates rotational and lateral latency and nearly eliminates the simultaneous block access limitations
- Buy just the capacity you need and still get needed IOPS
- Most SSDs provide in excess of 80,000 IOPS with latencies less than 1 ms



Improper Disk Configuration

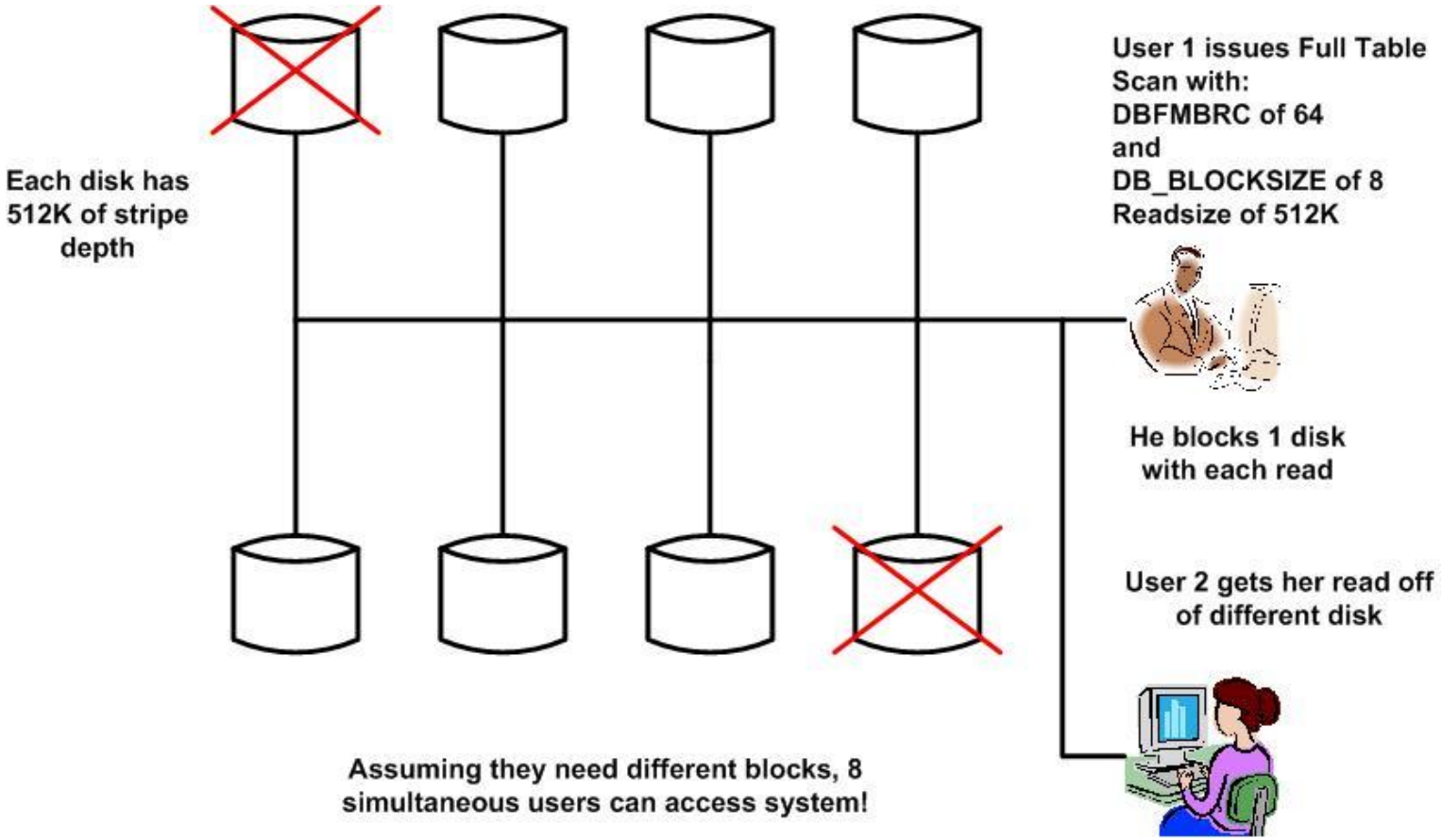
- Drives IO waits on properly sized IO subsystem
- The system or storage administrator must understand the system may have few write processes, but has many read processes
- This leads to, in an improperly configured disk system, that a single process doing a full index or table scan will block many other users.

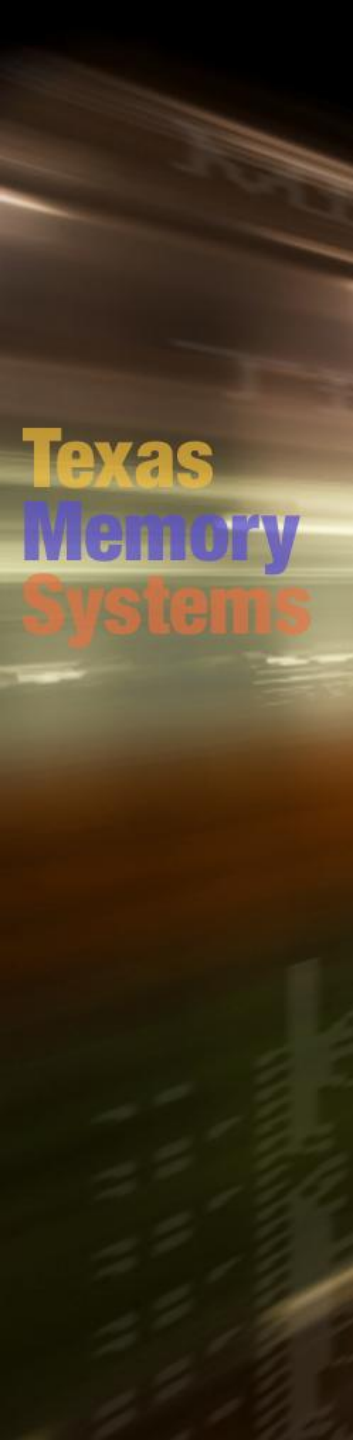
Blocking Read



Texas Memory Systems

Non-Blocking Read



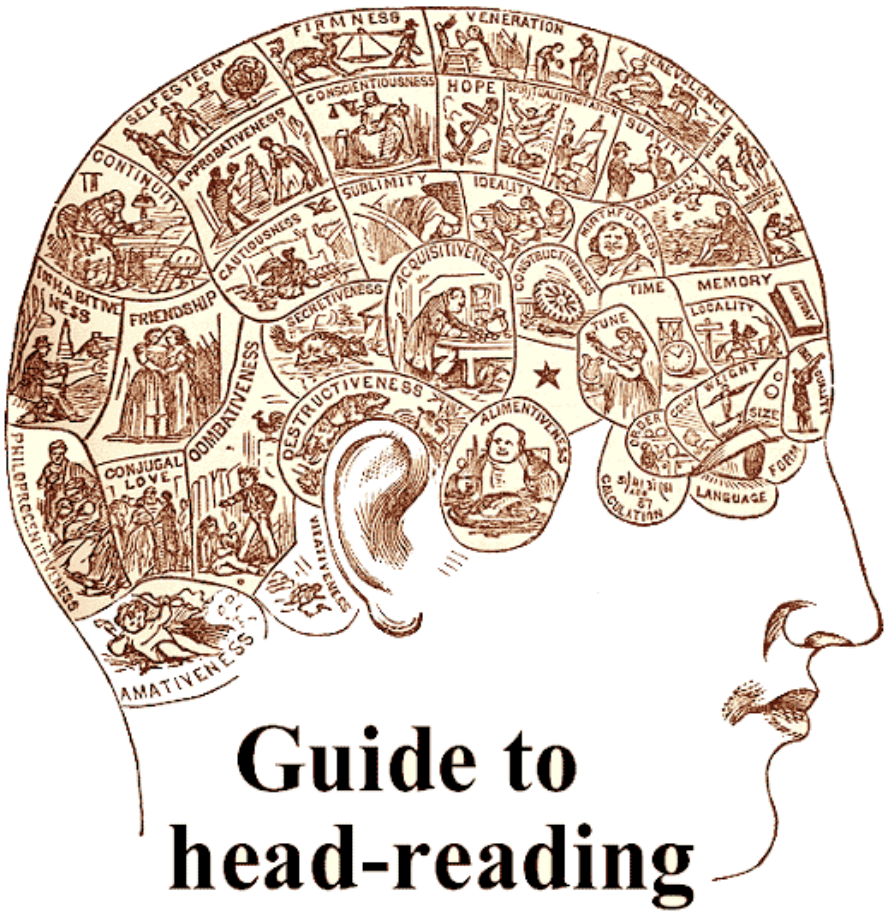


IO Subsystem

- Actual disk operations are more complex than shown,
- The general concept is valid
- You want as few a number of disks as possible servicing a single read for concurrency, while you want as many as possible for speed of access
- It is a tough balancing act for the IO subsystem due to the nature of physical disk drives and how the data is actually retrieved form the drives
- Note that SSD type systems usually eliminate blocking of this type.
- The storage manager must understand how Oracle accesses the IO subsystem and how to best optimize that IO subsystem for Oracle
- One size doesn’t fit all!

Texas Memory Systems

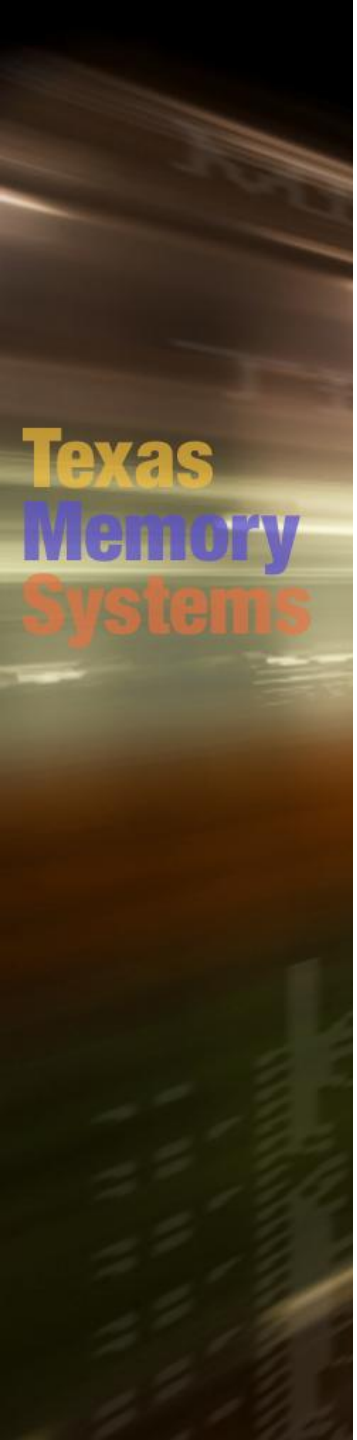
Memory Layer



Texas Memory Systems

Memory Layer Issues

- The memory layer is the actual area where the database caches, pools and other memory structures reside.
- Possible Issues:
  - **Insufficient Memory Resources**
  - **Improper Memory Configuration**



Insufficient Memory Resources

- Excessive physical reads.
- Excessive *db file sequential reads*.
- With automatic memory management may show as numerous deferred actions
- Excessive swapping of memory between components

Insufficient Memory Resource Indications

Example of memory starvation indication:

Load Profile	Per Second	Per Transaction
Physical reads:	5,789.69	10.86
Physical writes:	1,073.95	2.01

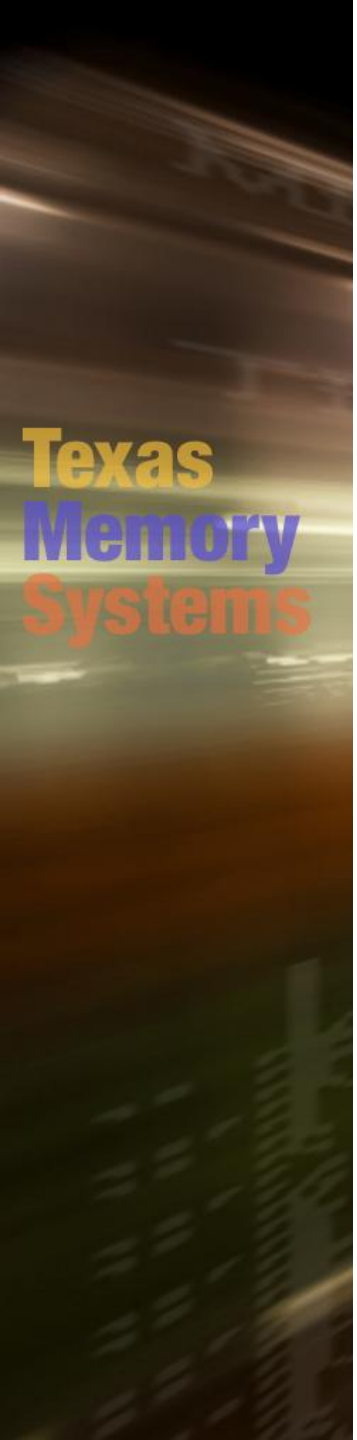
Top 5 Timed Events Event	Waits	Time (s)	Avg wait (ms)	%Total Call Time
db file sequential read	16,555,520	119,937	7	71.3
CPU time		30,420		18.1

Insufficient Memory Resource Indications

Texas Memory Systems

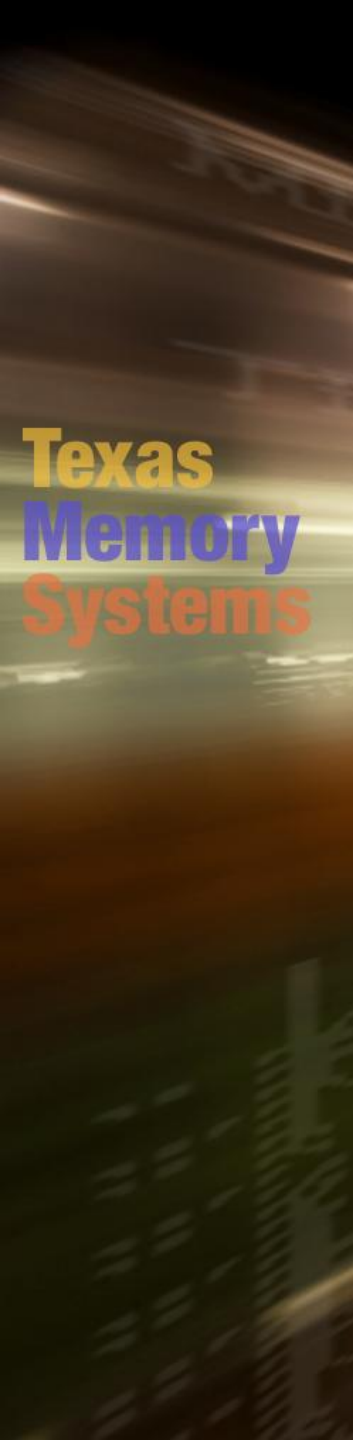
Buffer Pool Advisory
-> Only rows with estimated physical reads >0 are displayed
-> ordered by Block Size, Buffers For Estimate

P	Size for Est (M)	Size Factor	Buffers for Estimate	Est Phys Read Factor	Estimated Physical Reads
D	2,016	.9	63,756	1.0	25,481,605
D	2,240	1.0	70,840	1.0	24,857,599
D	2,320	1.0	73,370	1.0	24,642,801
D	2,464	1.1	77,924	1.0	24,183,584
D	2,688	1.2	85,008	0.9	22,843,987
D	2,912	1.3	92,092	0.8	19,788,845
D	3,136	1.4	99,176	0.7	16,978,075
D	3,360	1.4	106,260	0.6	15,417,232
D	3,584	1.5	113,344	0.6	14,522,845
D	3,808	1.6	120,428	0.6	13,701,710
D	4,032	1.7	127,512	0.5	13,237,721
D	4,256	1.8	134,596	0.5	12,826,268
D	4,480	1.9	141,680	0.5	12,447,012



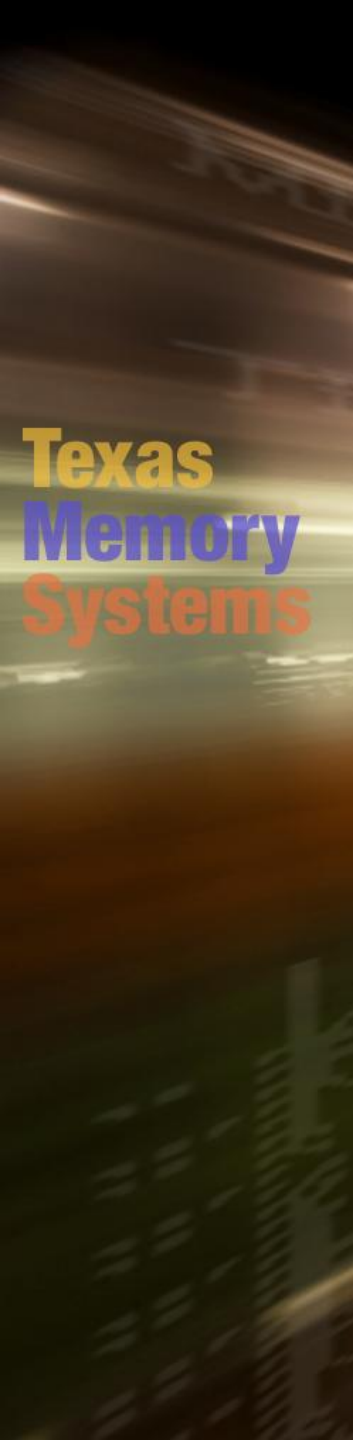
Improper Memory Configuration

- Within Oracle there are multiple caches and pools.
- The DBA should utilize proper settings for the shared pool database cache, keep, recycle and multiple block size caches.
- Additional settings for the large pool, java pool and streams pool should be used as needed
- Even when AMM is used, floor values should be set for these parameters



Improper Memory Configuration

- When using SGA_TARGET and SGA_MAX_SIZE or MEMORY_TARGET and MEMORY_MAX_SIZE, be sure they aren't set equal to each other
- This leads to memory thrashing
- I recommend at least 25% gap
- As you gain operational experience for your system's needs this gap can be reduced, or increased
- The dynamic performance view, v$sga_resize_ops and the AWR resize operations section can be utilized to rethink the floor settings for specific parameters



V$SGA_RESIZE_OPS Query

```
select
COMPONENT,  OPER_TYPE,  OPER_MODE,
INITIAL_SIZE, TARGET_SIZE,
FINAL_SIZE,
STATUS,
to_char(START_TIME,'mmdd hh24:mi')
start_time,
to_char(END_TIME,'mmdd hh24:mi')
end_time
from V$SGA_RESIZE_OPS
order by start_time
```

Example Report

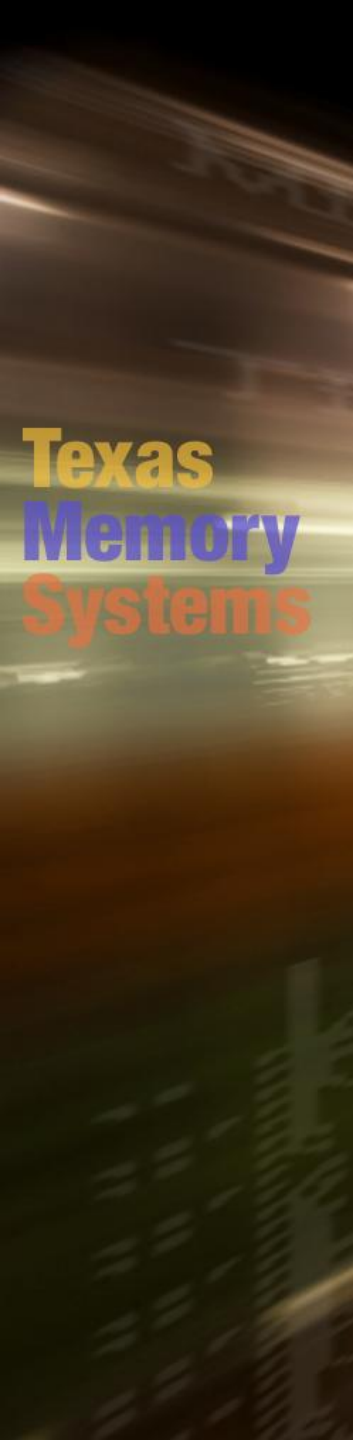
Date: 01/31/08 Page: 1
Time: 05:19 AM Component Resize Operations SYSTEM
supprac1 database

COMPONENT	Oper	OPER_MODE	INITIAL	TARGET	FINAL	STATUS	START_TIME	END_TIME
shared pool	SHRINK	DEFERRED	130023424	125829120	125829120	COMPLETE	0128 01:00	0128 01:00
DEFAULT buffer cache	GROW	DEFERRED	29360128	33554432	33554432	COMPLETE	0128 01:00	0128 01:00
DEFAULT buffer cache	SHRINK	DEFERRED	33554432	29360128	29360128	COMPLETE	0128 01:01	0128 01:01
shared pool	GROW	DEFERRED	125829120	130023424	130023424	COMPLETE	0128 01:01	0128 01:01
DEFAULT buffer cache	SHRINK	DEFERRED	29360128	25165824	25165824	COMPLETE	0128 01:02	0128 01:02
shared pool	GROW	DEFERRED	130023424	134217728	134217728	COMPLETE	0128 01:02	0128 01:02
shared pool	SHRINK	DEFERRED	134217728	130023424	130023424	COMPLETE	0128 01:09	0128 01:09
DEFAULT buffer cache	GROW	DEFERRED	25165824	29360128	29360128	COMPLETE	0128 01:09	0128 01:09
DEFAULT buffer cache	SHRINK	DEFERRED	29360128	25165824	25165824	COMPLETE	0128 01:15	0128 01:15
shared pool	GROW	DEFERRED	130023424	134217728	134217728	COMPLETE	0128 01:15	0128 01:15
shared pool	SHRINK	DEFERRED	134217728	130023424	130023424	COMPLETE	0128 01:49	0128 01:49
DEFAULT buffer cache	GROW	DEFERRED	25165824	29360128	29360128	COMPLETE	0128 01:49	0128 01:49
shared pool	SHRINK	DEFERRED	130023424	125829120	125829120	COMPLETE	0128 02:00	0128 02:00
DEFAULT buffer cache	GROW	DEFERRED	29360128	33554432	33554432	COMPLETE	0128 02:00	0128 02:00
DEFAULT buffer cache	SHRINK	DEFERRED	33554432	29360128	29360128	COMPLETE	0128 02:02	0128 02:02
shared pool	GROW	DEFERRED	125829120	130023424	130023424	COMPLETE	0128 02:02	0128 02:02
DEFAULT buffer cache	SHRINK	DEFERRED	29360128	25165824	25165824	COMPLETE	0128 02:08	0128 02:08

Texas Memory Systems

Memory Layer

- In data warehouse (DWH) and decision support (DSS) it may be impossible to provide enough cache to support all operations.
- In many cases the developers or the DBAs or both may not be aware of all the options Oracle provides.
- For Example: utilizing star joins.
- Star joins will:
  - reduce 30 minute queries to 30 seconds
  - Reduce FTS
  - Reduce Memory usage

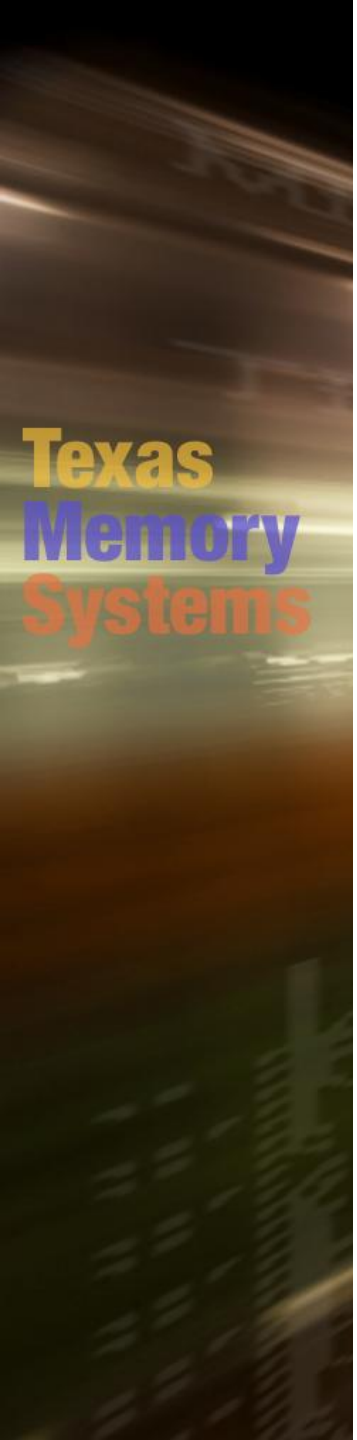


Memory Layer

- When you have to have *db file scattered reads* (full table or index scans)
- Validate index strategy
- Make sure the IO subsystem is properly sized to handle the load.
- It may be necessary to over buy the number of disks needed to the tune of 30-40 or more times the number of disks needed for storage capacity
- The major cause for the need for all of these drives is contention and the queuing that results with too-few spindles.
- By utilizing low latency IO subsystems such as SSD that don’t block, you don’t need to over-buy on storage capacity to meet IOPS needs.

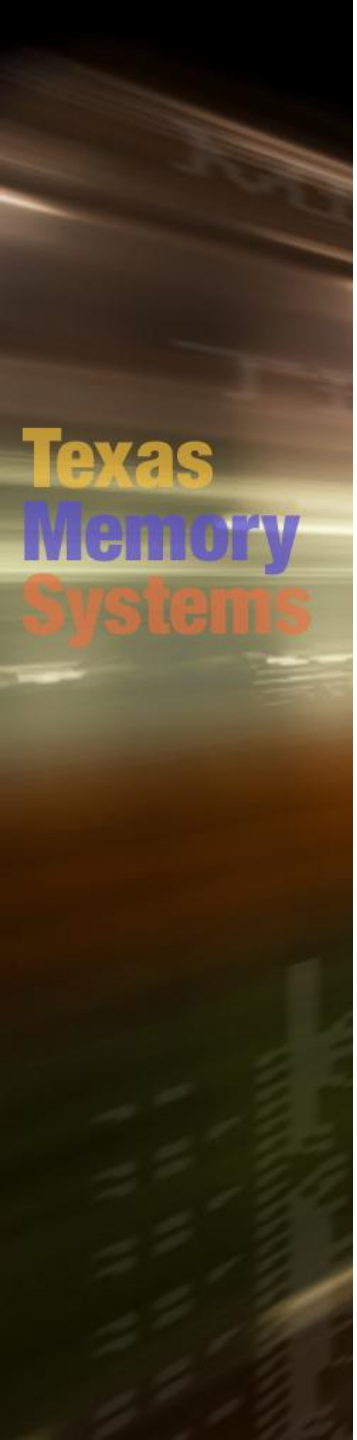
Texas Memory Systems

CPU Layer



CPU Layer

- Where all of the processing occurs within a database.
- With the introduction of extremely distribute processing as in the Exadata systems, this line is blurring
- Some of the processing is actually being done at the storage level.
- This may even blur more as CPUs are built into memory chips (8 CPUs on a single gigabyte memory chip)
- CPU Issues:
  - Insufficient CPU Resources



Insufficient CPU Resources

- Tuning usually pushes the log jam from one part of the information stream to another
- Once we tune the code, fix the IO subsystem, and fix the memory issues we may have with our system we usually end up with higher CPU utilization.
- Higher CPU utilization in itself is not a bad thing, unless it leads to over utilization or overloading of the system CPUs.

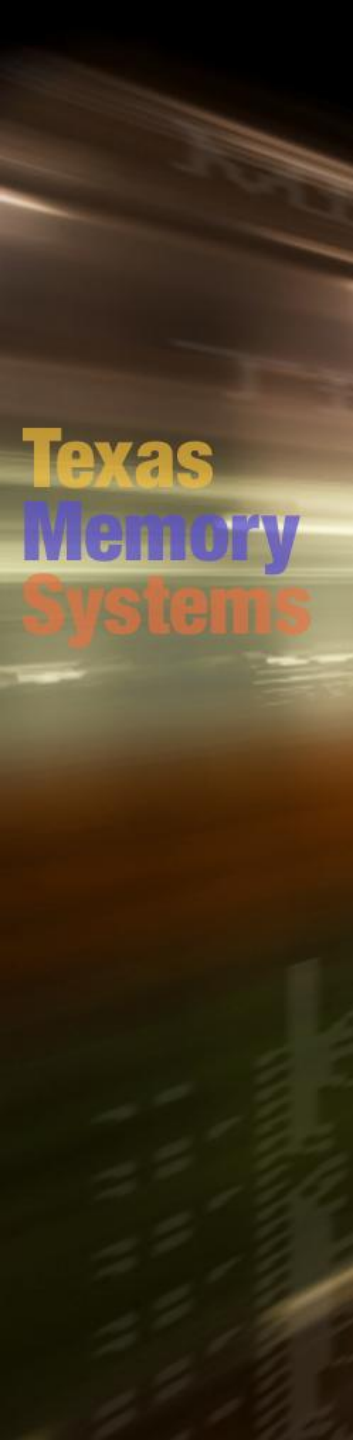
Texas Memory Systems

Insufficient CPU Resources

- Look at the balance between Busy, Idle and IO Wait on the CPUs
- For example, if the percent busy is 30, the percent idle 20 and the IO wait percent 50, then even if we reduce the IO wait to less than 5% we can still only reclaim 20 percent of performance losses.
- In order to get any improvement to the IO subsystem above 20 percent we would need to reduce CPU usage, add more CPUs or replace the CPUs with ones that can sustain a higher level of operations

Texas Memory Systems

Middleware



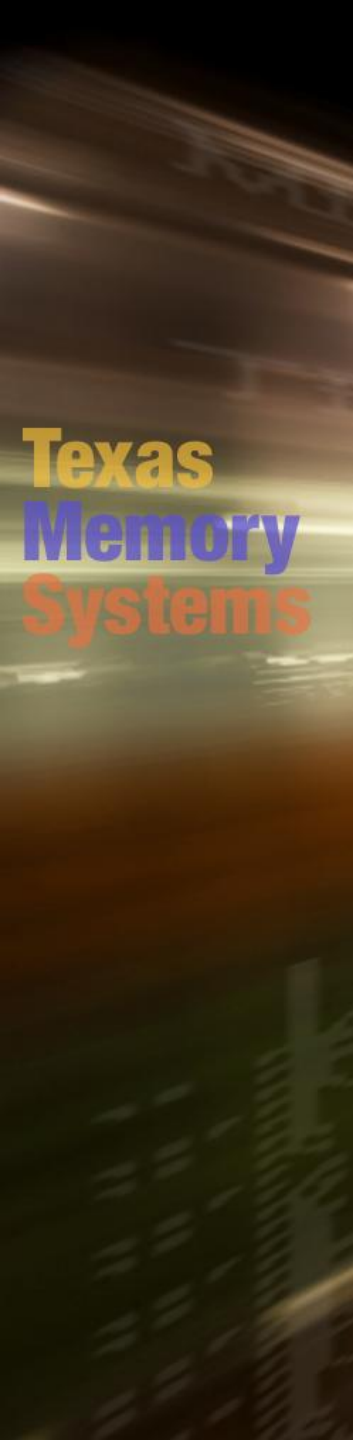
Middleware or Application Issues

- The middleware/application server layer may be were application logic is processed and may also act as a caching area for frequently used data.
- Middleware or Application issues:
  - Bad SQL code generators
  - Improper configurations

Texas Memory Systems

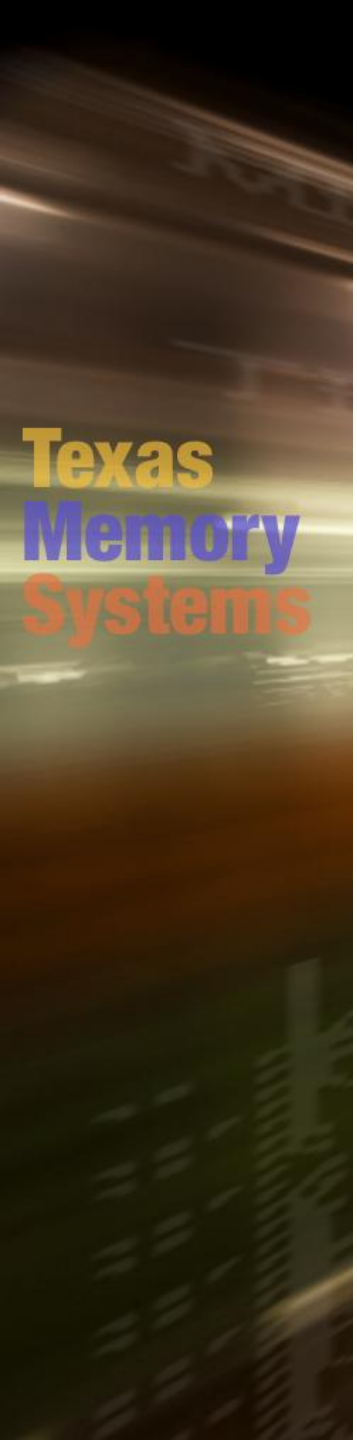
Bad SQL Code Generators

- Middleware and software configurations can be a real problem.
- The local staff usually have no control over the code
- The biggest problems usually have to do with code generators that develop ad-hoc queries against the database.
- Too often these code generators (actually, their developers) have no clue how to write optimal SQL for Oracle, they:
  - Don’t use proper joins
  - Overuse outside joins
  - Use distincts when they aren’t needed
  - Fail to use bind variables



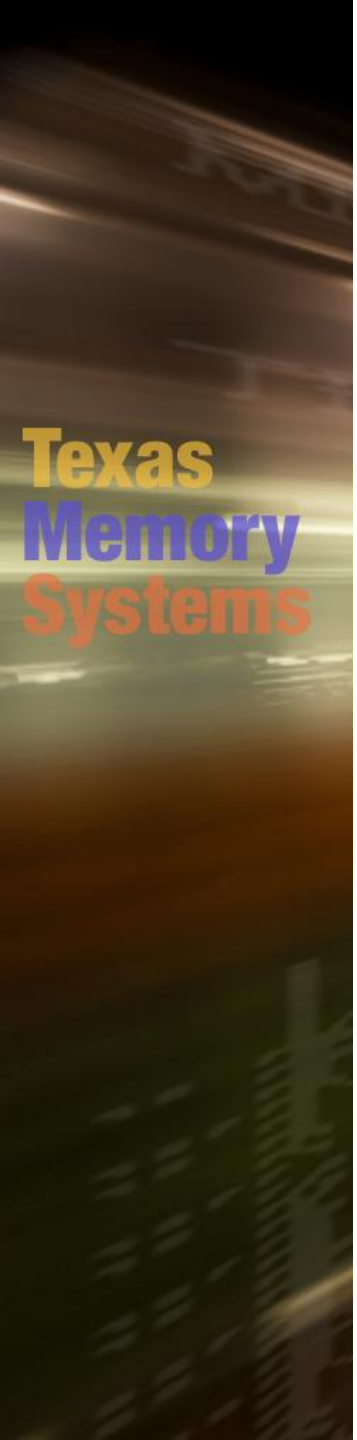
Bad Configurations

- Failure to allow for array processing
- SQLNet uses a default array size of 10, JAVA uses 15.
- If your result set is in the hundreds or thousands of values and you leave these settings at their default, you just got hundreds of SQLNet roundtrips added to your response time



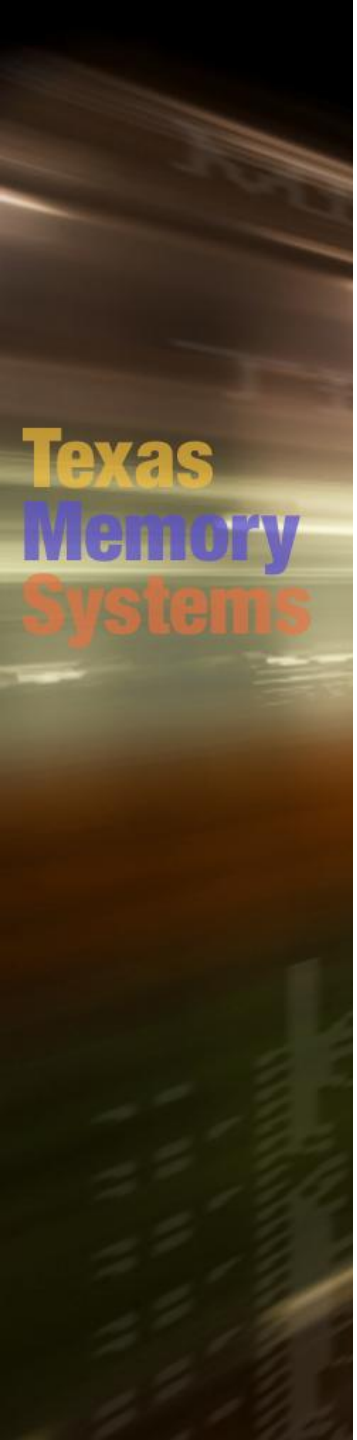
Example

- A company called me in to examine their application.
- It was taking 30 seconds to get a response and of course Oracle was getting the blame.
- Using the v$sqltext and v$sqlarea views I was able to isolate the key SQL for specific forms and reports, then I used the Quest – Benchmark Factory to model the application
- In over 90 percent of the scenarios they could describe accurately the response at the database layer was less and 1 second for all queries.
- The Cognos report layer was not using bind variables and was using the default SQLPlus arraysize of 10.
- This resulted in excessive recursive SQL, and, hundreds of SQLNet roundtrips on a network where some areas where remote (India, South America, Europe) and have anywhere from 100-500 millisecond network ping times.



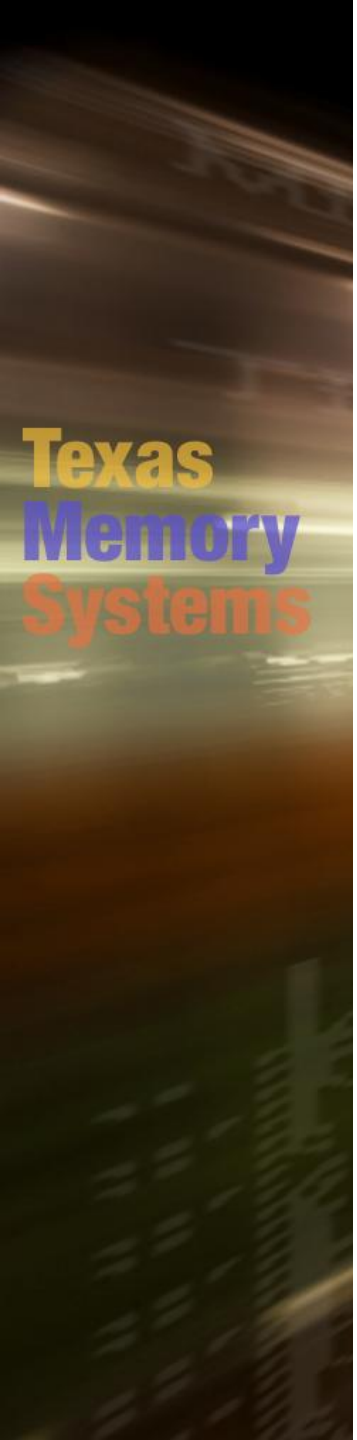
Solution:

- Always test assuming data volumes will be hundreds or thousands of times your test set, unless you are lucky enough to test with full data sets.
- Always monitor the affects of all layers upon the others.



Summary

- In order to be holistically tuned a transaction (insert, update, delete or select) utilizes the minimum number of resources (Physical IO, logical IO, memory and CPU) possible.
- The database must be properly designed, the IO subsystem adequate to handle both the storage capacity and the IOPS needed with the proper latency, and that the CPU and memory resources are enough to handle the needed load.
- Once you start thinking holistically about performance and how the entire system, not a single statement, is tuned, you are on the way to better, longer lasting performance.



Questions?

- Mike.a@ramsan.com
- http://www.statspackanalyzer.com
- Http://www.ramsan.com