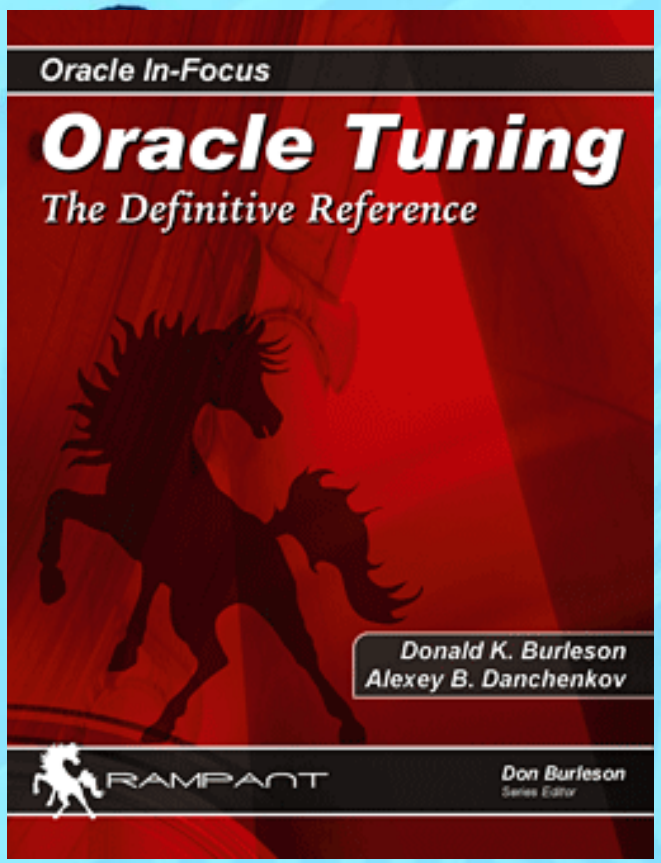
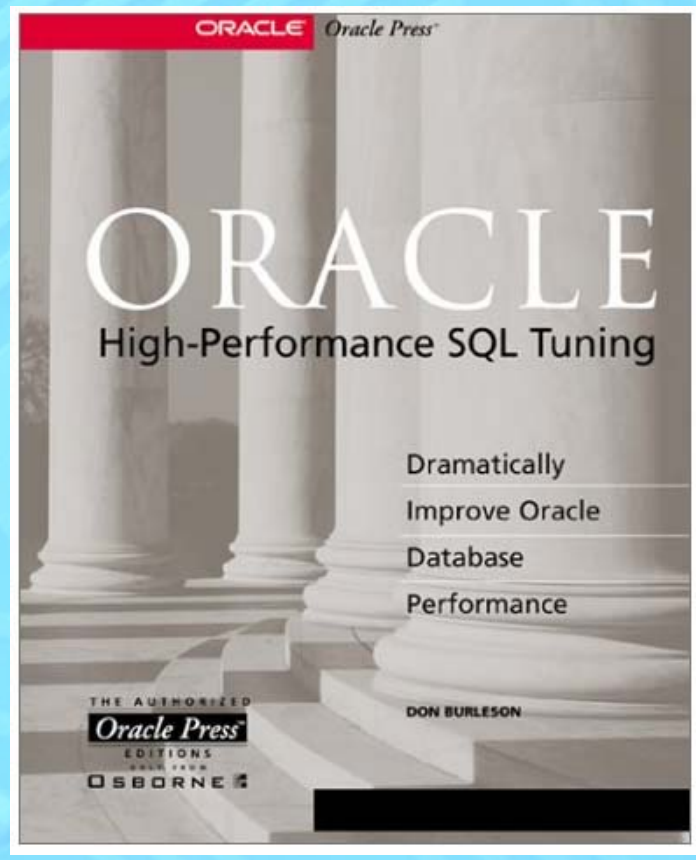


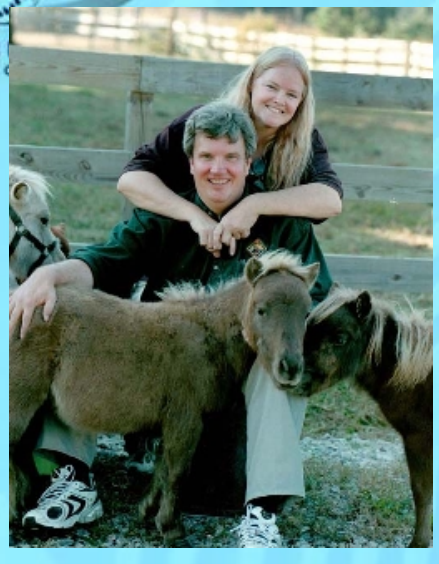
Extending the Power of AWR

Don Burleson



BURLESON CONSULTING

www.GuideHorse.org



CBS NEWS

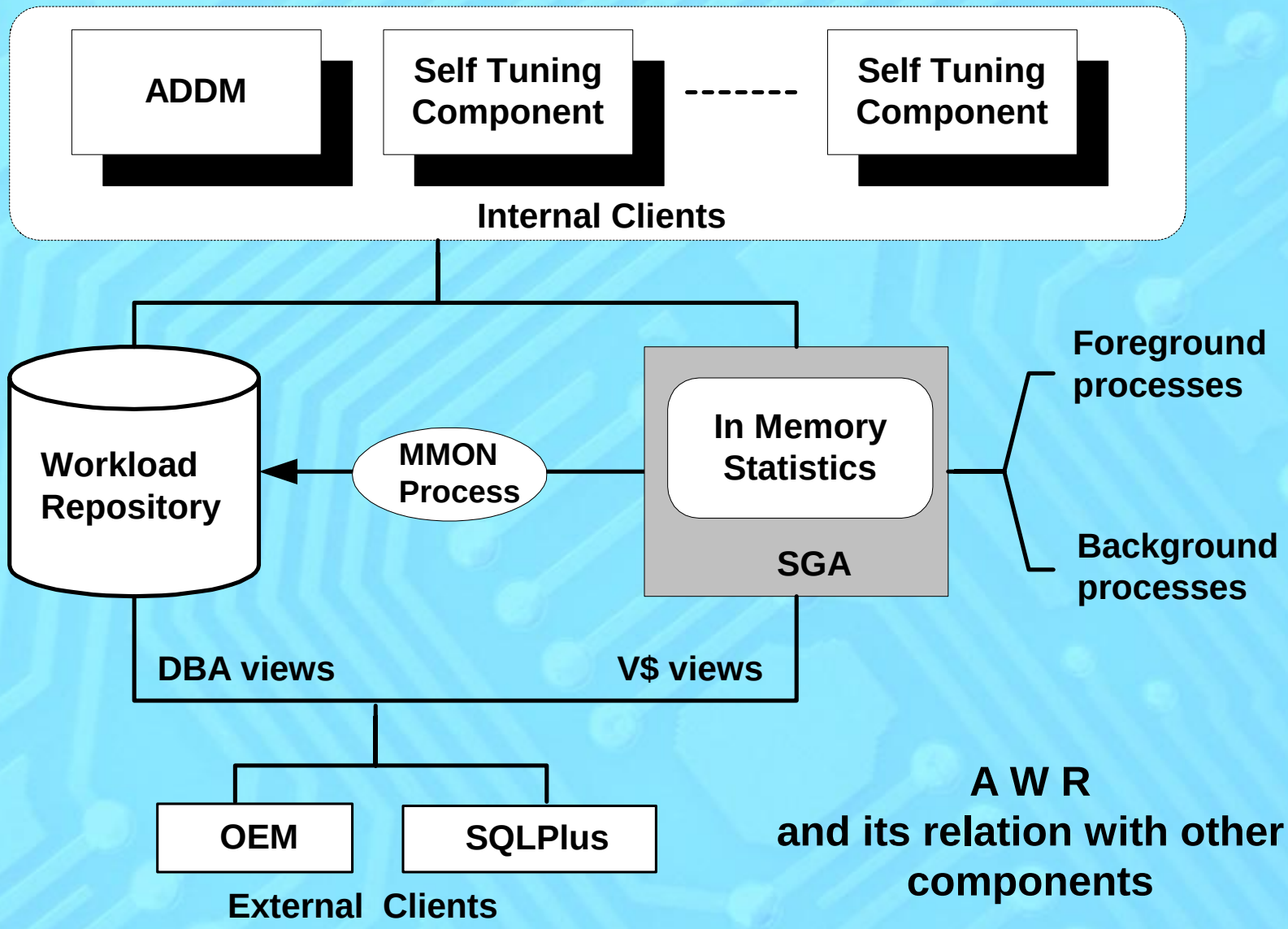
abc 20/20

People

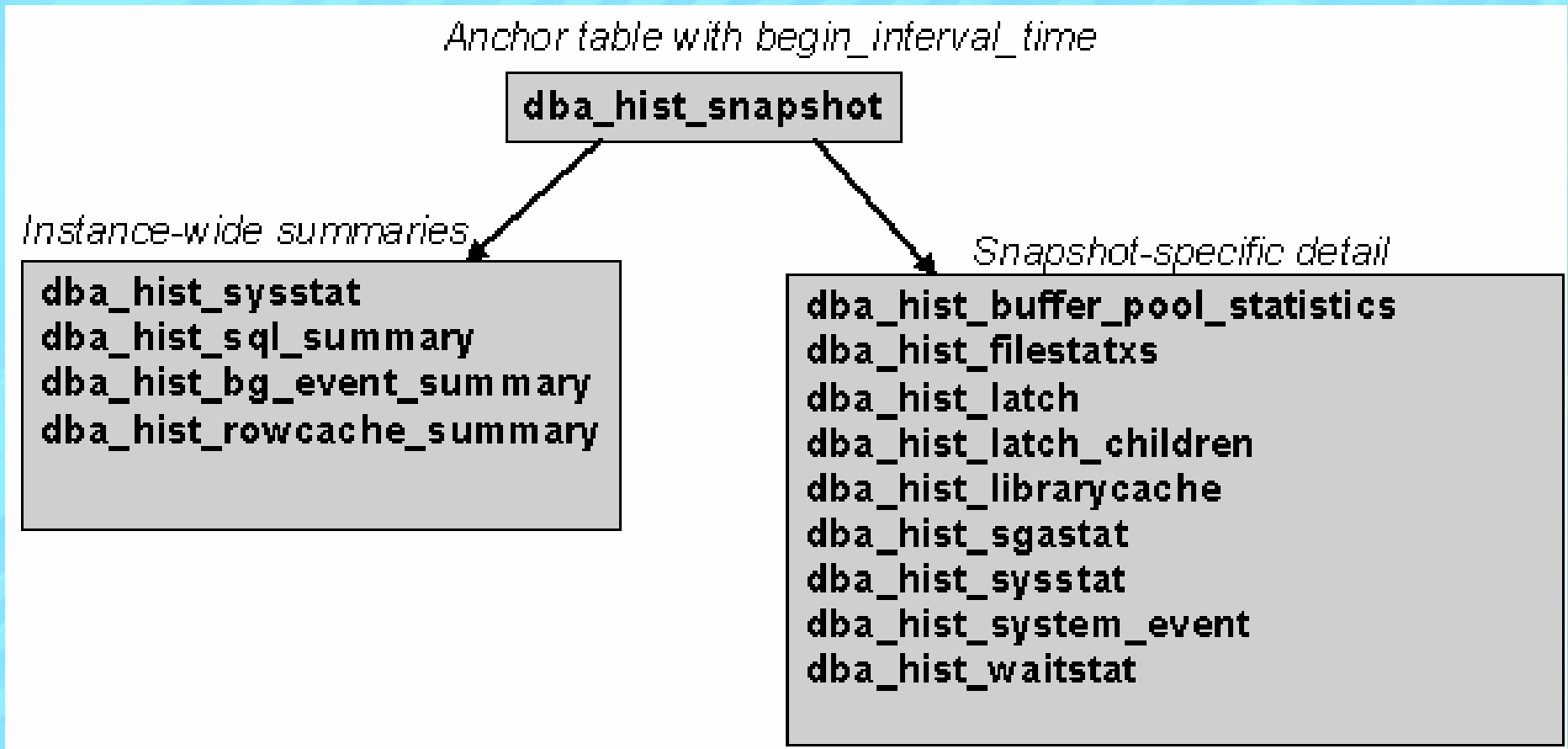
The New York Times

AP Associated Press

TIME



A W R
and its relation with other components


Anchor table with begin_interval_time
dba_hist_snapshot
Instance-wide summaries
dba_hist_sysstat
dba_hist_sql_summary
dba_hist_bg_event_summary
dba_hist_rowcache_summary
Snapshot-specific detail
dba_hist_buffer_pool_statistics
dba_hist_filestatxs
dba_hist_latch
dba_hist_latch_children
dba_hist_librarycache
dba_hist_sgastat
dba_hist_sysstat
dba_hist_system_event
dba_hist_waitstat

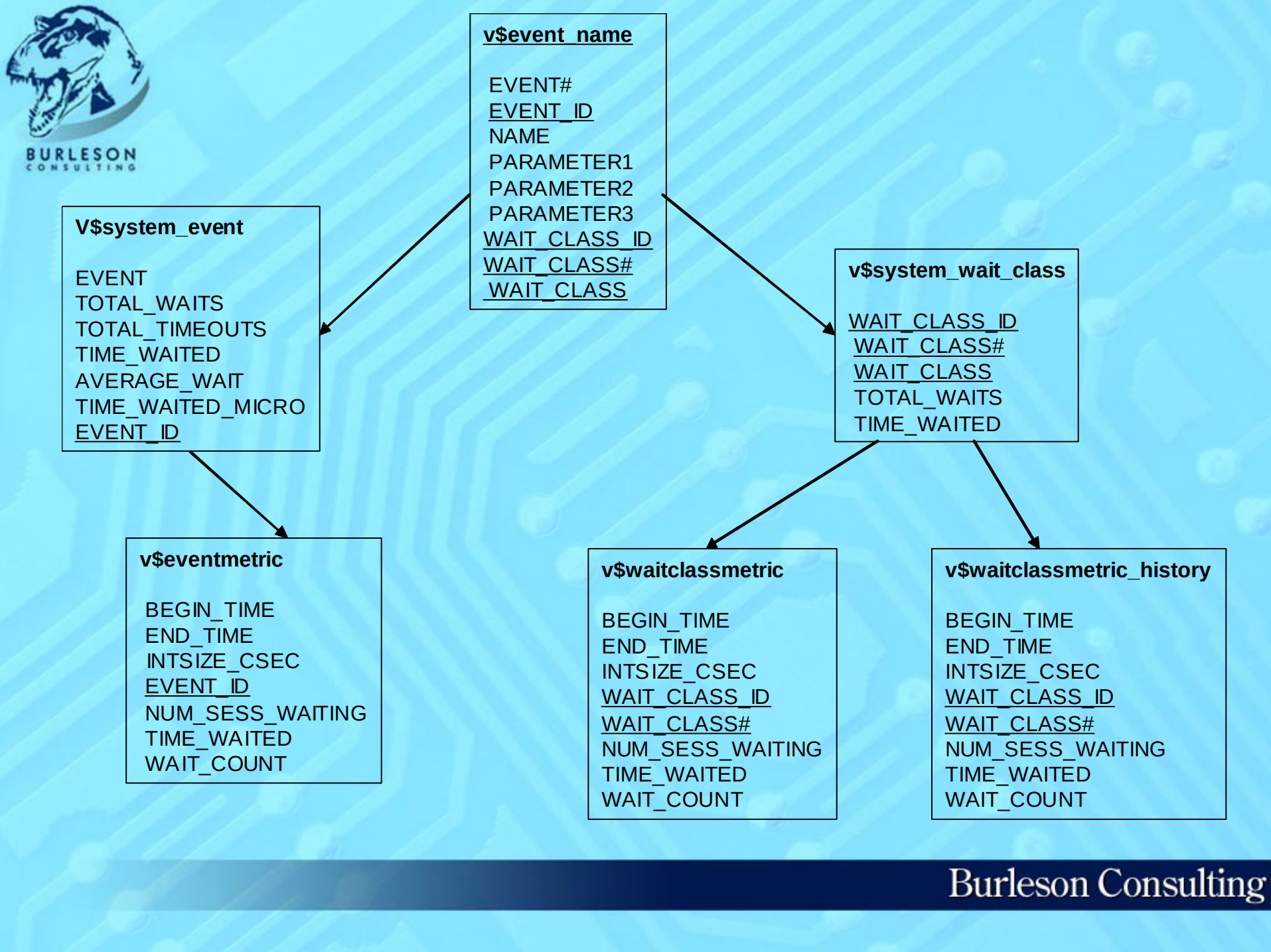
DBA HIST View	*wrh$*Table	STATSPACK Table
dba_hist_event_summary	*wrh$_bg_event_summary*	*stats$bg_event_summary*
dba_hist_buffer_pool_statistics	*wrh$_buffer_pool_statistics*	*stats$buffer_pool_statistics*
dba_hist_filestatxs	*wrh$_filestatxs*	*stats$filestatxs*
dba_hist_latch	*wrh$_latch*	*stats$latch*
dba_hist_latch_children	*wrh$_latch_children*	*stats$latch_children*
dba_hist_librarycache	*wrh$_librarycache*	*stats$librarycache*
dba_hist_rowcache_summary	*wrh$_rowcache_summary*	*stats$rowcache_summary*
dba_hist_sgastat	*wrh$_sgastat*	*stats$sgastat*
dba_hist_sql_summary	*wrh$_sql_summary*	*stats$sql_summary*
dba_hist_sysstat	*wrh$_sysstat*	*stats$sysstat*
dba_hist_system_event	*wrh$_system_event*	*stats$system_event*
dba_hist_waitstat	*wrh$_waitstat*	*stats$waitstat*

AWR Physical Reads

```
select
    begin_interval_time,
    filename,
    phyrds
from
    dba_hist_filestatxs
  natural join
    dba_hist_snapshot
order by
  begin_interval_time;
```

Physical Reads

BEGIN_INTERVAL_TIME	FILENAME	PHYRDS
10-NOV-04 09.00.01.000 PM	/oradata/test10g/system01.dbf	3,982
	/oradata/test10g/undotbs01.dbf	51
	/oradata/test10g/users01.dbf	7
	/oradata/test10g/example01.dbf	14
	/oradata/test10g/sysaux01.dbf	551
	/oradata/test10g/tbsalert.dbf	7


v$event_name
EVENT#
EVENT_ID
NAME
PARAMETER1
PARAMETER2
PARAMETER3
WAIT_CLASS_ID
WAIT_CLASS#
WAIT_CLASS
V$system_event
EVENT
TOTAL_WAITS
TOTAL_TIMEOUTS
TIME_WAITED
AVERAGE_WAIT
TIME_WAITED_MICRO
EVENT_ID
v$system_wait_class
WAIT_CLASS_ID
WAIT_CLASS#
WAIT_CLASS
TOTAL_WAITS
TIME_WAITED
v$eventmetric
BEGIN_TIME
END_TIME
INTSIZE_CSEC
EVENT_ID
NUM_SESS_WAITING
TIME_WAITED
WAIT_COUNT
v$waitclassmetric
BEGIN_TIME
END_TIME
INTSIZE_CSEC
WAIT_CLASS_ID
WAIT_CLASS#
NUM_SESS_WAITING
TIME_WAITED
WAIT_COUNT
v$waitclassmetric_history
BEGIN_TIME
END_TIME
INTSIZE_CSEC
WAIT_CLASS_ID
WAIT_CLASS#
NUM_SESS_WAITING
TIME_WAITED
WAIT_COUNT


```
select event
     , waits "Waits"
     , time "Wait Time (s)"
     , pct*100 "Percent of Total"
     , waitclass "Wait Class"
from (select e.event_name event
                    , e.total_waits - nvl(b.total_waits,0)  waits
                    , (e.time_waited_micro - nvl(b.time_waited_micro,0))/1000000
time
                    , (e.time_waited_micro - nvl(b.time_waited_micro,0))/
                      (select sum(e1.time_waited_micro -
nvl(b1.time_waited_micro,0)) from dba_hist_system_event b1 , dba_hist_system_event
e1
                       where b1.snap_id(+)          = b.snap_id
                         and e1.snap_id             = e.snap_id
                         and b1.dbid(+)             = b.dbid
                         and e1.dbid                = e.dbid
                         and b1.instance_number(+)  = b.instance_number
                         and e1.instance_number     = e.instance_number
                         and b1.event_id(+)         = e1.event_id
                         and e1.total_waits         > nvl(b1.total_waits,0)
                         and e1.wait_class          <> 'Idle'
  )  pct
                    , e.wait_class waitclass
                from
                  dba_hist_system_event b ,
                  dba_hist_system_event e
               where b.snap_id(+)          = &pBgnSnap
                 and e.snap_id             = &pEndSnap
                 and b.dbid(+)             = &pDbId
                 and e.dbid                = &pDbId
                 and b.instance_number(+)  = &pInstNum
                 and e.instance_number     = &pInstNum
                 and b.event_id(+)         = e.event_id
                 and e.total_waits         > nvl(b.total_waits,0)
                 and e.wait_class          <> 'Idle'
        order by time desc, waits desc
       )
```

EVENT	Waits	Wait Time (s)	Percent of Total	Wait Class
control file parallel write	11719	119.13	34,1611762	System I/O
class slave wait	20	102.46	29,3801623	Other
Queue Monitor Task Wait	74	66.74	19,1371008	Other
log file sync	733	20.60	5,90795938	Commit
db file sequential read	1403	14.27	4,09060416	User I/O
log buffer space	178	10.17	2,91745801	Configuration
process startup	114	7.65	2,19243344	Other
db file scattered read	311	2.14	,612767501	User I/O
control file sequential read	7906	1.33	,380047642	System I/O
latch free	254	1.13	,324271668	Other
log file switch completion	20	1.11	,319292495	Configuration

Exception Reporting with the AWR

```
accept stat_name   char   prompt 'Enter Statistic Name:  ';
accept stat_value  number prompt 'Enter Statistics Threshold value:  ';
col snap_time   format a19
col value       format 999,999,999
select
   to_char(begin_interval_time,'yyyy-mm-dd hh24:mi') snap_time,
   value
from
   dba_hist_sysstat
  natural join
   dba_hist_snapshot
where
   stat_name = '&stat_name'
and
  value > &stat_value
order by
   to_char(begin_interval_time,'yyyy-mm-dd hh24:mi')
;
```


```
Enter Statistic Name:  physical writes
Enter Statistics Threshold value:  200000

SNAP_TIME                          VALUE
-------------------- ------------
2004-02-21 08:00                 200,395
2004-02-27 08:00                 342,231
2004-02-29 08:00                 476,386
2004-03-01 08:00                 277,282
2004-03-02 08:00                 252,396
2004-03-04 09:00                 203,407
```

Hot file writes:

```
select
   to_char(begin_interval_time,'yyyy-mm-dd hh24:mi') snap_time,
   filename,
   phywrts
from
   dba_hist_filestatxs
natural join
   dba_hist_snapshot
where
   phywrts > 0
and
   phywrts * 4 >
(
select
   avg(value)              all_phys_writes
from
   dba_hist_sysstat
  natural join
   dba_hist_snapshot
where
   stat_name = 'physical writes'
and
  value > 0
)
order by
   to_char(begin_interval_time,'yyyy-mm-dd hh24:mi'),
   phywrts desc
;
```


```
SQL> @hot_write_files

This will identify any single file who's write I/O
is more than 25% of the total write I/O of the database.


SNAP_TIME            FILENAME                                    PHYWRTS
-------------------- ------------------------------------------ -----------
2004-02-20 23:30     E:\ORACLE\ORA92\FSDEV10G\SYSAUX01.DBF           85,540

2004-02-21 01:00     E:\ORACLE\ORA92\FSDEV10G\SYSAUX01.DBF           88,843

2004-02-21 08:31     E:\ORACLE\ORA92\FSDEV10G\SYSAUX01.DBF           89,463

2004-02-22 02:00     E:\ORACLE\ORA92\FSDEV10G\SYSAUX01.DBF           90,168
2004-02-22 16:30     E:\ORACLE\ORA92\FSDEV10G\SYSAUX01.DBF          143,974
                     E:\ORACLE\ORA92\FSDEV10G\UNDOTBS01.DBF          88,973
```

Signature Analysis

- “Know thy Database”
- Spot “hidden” trends
- Allows holistic tuning
- Allows just-in-time anticipation
- Allows adjusting of object characteristics (freelists, file placement, caching, block population)

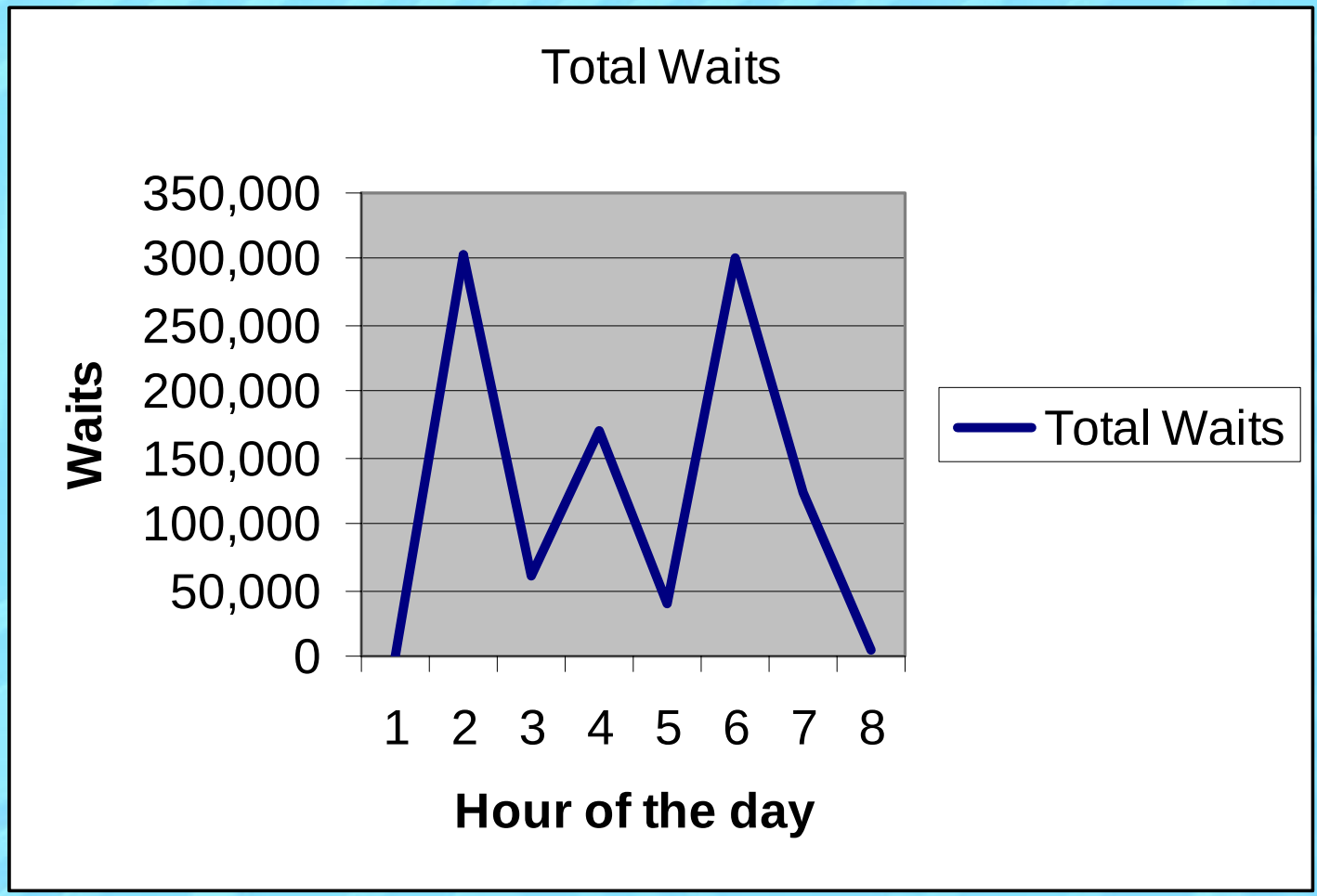
Signature Analysis

```
select
   TO_CHAR(h.sample_time,'HH24') "Hour",
   Sum(h.wait_time/100) "Total Wait Time (Sec)"
from
   dba_hist_active_sess_history h,
   v$event_name                 n
where
   h.session_state = 'ON CPU'
and
   h.session_type = 'FOREGROUND'
and
   h.event_id = n.EVENT_ID
and
   n.wait_class <> 'Idle'
group by
   TO_CHAR(h.sample_time,'HH24')
```

Wait Signature by Hour

Hr	Total Wait Time (Sec)
11	219
12	302,998
13	60,982
14	169,716
15	39,593
16	299,953
17	122,933
18	5,147

Wait Signature by Hour



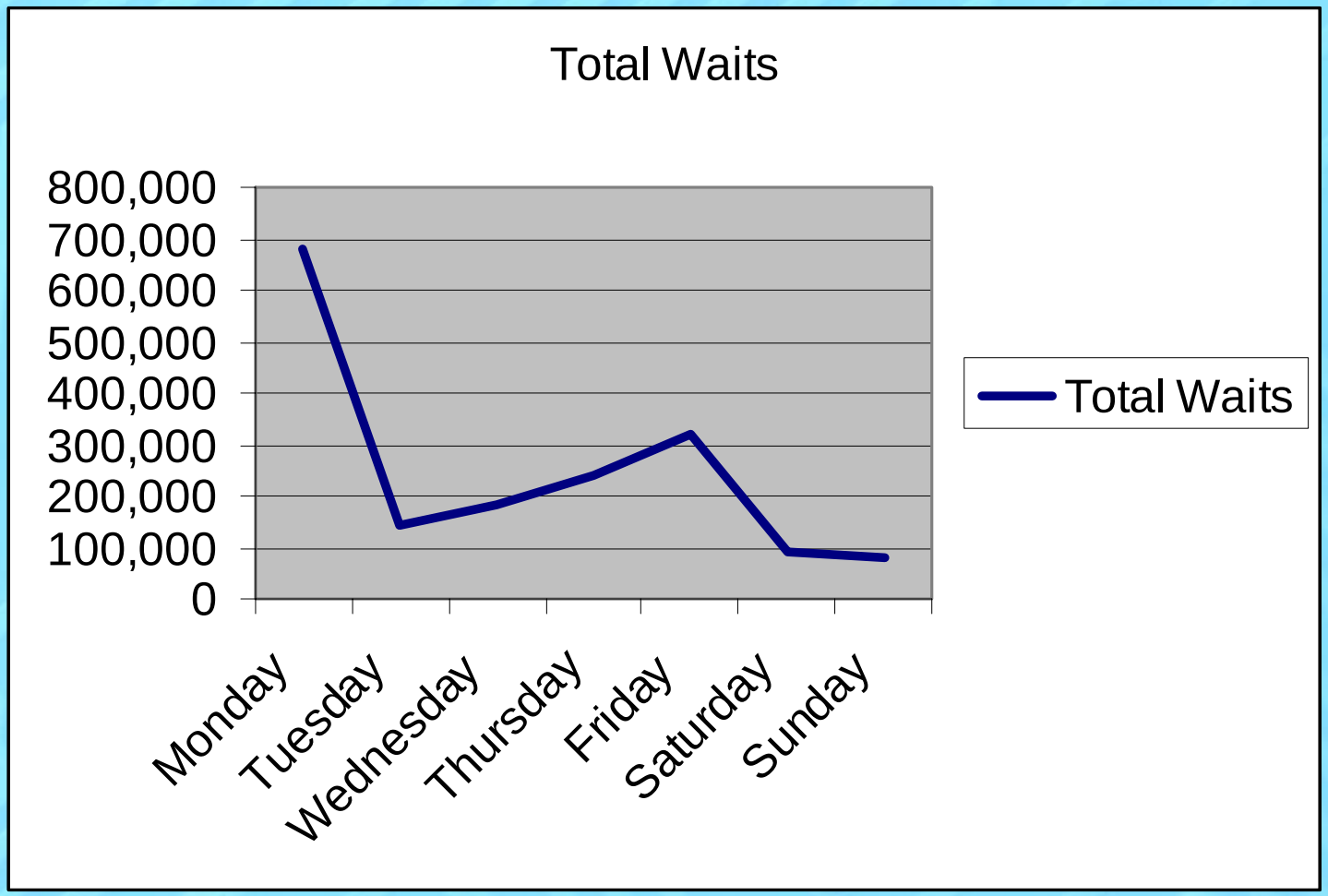
Signature Analysis

```
select
   TO_CHAR(h.sample_time,'Day') "Hour",
   sum(h.wait_time/100) "Total Wait Time (Sec)"
from
   dba_hist_active_sess_history h,
   v$event_name n
where
   h.session_state = 'ON CPU'
and
   h.session_type = 'FOREGROUND'
and
   h.event_id = n.EVENT_ID
and
   n.wait_class <> 'Idle'
group by
   TO_CHAR(h.sample_time,'Day')
```

Total waits by Day of week

Hour	Total Wait Time (Sec)
Monday	679,089
Tuesday	141,142
Wednesday	181,226
Thursday	241,711
Friday	319,023
Saturday	93,362
Sunday	81,086

Total waits by Day of week



Time-series SQL analysis

Databases are not static, and point-in-time SQL tuning assumes static data.

- Tracking index usage over time
- Tracking table access methods over time
- SQL changes execution plans over time
  - Different bind variables
  - Different schema statistics
  - Different materialized views
  - Different table data

In 10g, you can see v$sql_plan over time with dba_hist_sqlplan

Full table scans and counts

Snapshot Time	OWNER	NAME	NUM_ROWS	C	K	BLOCKS	NBR_FTS
12/08/04 14	APPLSYS	FND_CONC_RELEASE_DISJS	39	N	K	2	98,864
	APPLSYS	FND_CONC_RELEASE_PERIODS	39	N	K	2	98,864
	APPLSYS	FND_CONC_RELEASE_STATES	1	N	K	2	98,864
	SYS	DUAL		N	K	2	63,466
	APPLSYS	FND_CONC_PP_ACTIONS	7,021	N		1,262	52,036
	APPLSYS	FND_CONC_REL_CONJ_MEMBER	0	N	K	22	50,174
12/08/04 15	APPLSYS	FND_CONC_RELEASE_DISJS	39	N	K	2	33,811
	APPLSYS	FND_CONC_RELEASE_PERIODS	39	N	K	2	2,864
	APPLSYS	FND_CONC_RELEASE_STATES	1	N	K	2	32,864
	SYS	DUAL		N	K	2	63,466
	APPLSYS	FND_CONC_PP_ACTIONS	7,021	N		1,262	12,033
	APPLSYS	FND_CONC_REL_CONJ_MEMBER	0	N	K	22	50,174

Source: plan10g.sql from Oracle Tuning by Alexey Danchenkov

Track Nested Loop joins

```
select
   to_char(sn.begin_interval_time,'yy-mm-dd hh24')  c1,
   count(*)                                         c2,
   sum(st.rows_processed_delta)                     c3,
   sum(st.disk_reads_delta)                         c4,
   sum(st.cpu_time_delta)                           c5
from
   dba_hist_snapshot sn,
   dba_hist_sql_plan  p,
   dba_hist_sqlstat  st
where
   st.sql_id = p.sql_id
and
   sn.snap_id = st.snap_id
and
   p.operation = 'NESTED LOOPS'
group by
to_char(sn.begin_interval_time,'yy-mm-dd hh24')
having
   count(*) > &nest_thr
```

Nested Loop Tracking

Nested Loop Join Thresholds

Date	Nested Loops Count	Rows Processed	Disk Reads	CPU Time
04-10-10 16	22	750	796	4,017,301
04-10-10 17	25	846	6	3,903,560
04-10-10 19	26	751	1,430	4,165,270
04-10-10 20	24	920	3	3,940,002

Track Object usage by Day!

```
select
decode(c2,2,'Monday',3,'Tuesday',4,'Wednesday',5,'Thursday',6,'Friday',7
    ,'Saturday',1,'Sunday')
From (
select
    p.object_name                          c1,
    to_char(sn.end_interval_time,'d')      c2,
    count(1)                               c3
from
  dba_hist_sql_plan   p,
  dba_hist_sqlstat    s,
  dba_hist_snapshot  sn
where
  p.object_owner <> 'SYS'
and
  p.sql_id = s.sql_id
and
  s.snap_id = sn.snap_id
group by    p.object_name,    to_char(sn.end_interval_time,'d')
order by  c2,c1);
```

Track Object usage by Day!

Week Day	Object Name	Invocation Count
Monday	CUSTOMER	44
	CUSTOMER_ORDERS	44
	CUSTOMER_ORDERS_PRIMARY	44
	MGMT_CURRENT_METRICS_PK	43
	MGMT_FAILOVER_TABLE	47
	MGMT_JOB	235
	MGMT_JOB_EMD_STATUS_QUEUE	91
	MGMT_JOB_EXECUTION	235
	MGMT_JOB_PK	235
	MGMT_METRICS	65
	MGMT_METRICS_1HOUR_PK	43
Tuesday	CUSTOMER	40
	CUSTOMER_CHECK	2
	CUSTOMER_PRIMARY	1

Track large-table full-table scans

- **May indicate a tuning opportunity**
  (Missing index, Missing Materialized View)
- **High Stress on your disk I/O sub-system**
  (no caching of blocks)
- **Parallel large-table full-table scans drive-up CPU**

Track large-table full-table scans

```
select
  to_char(sn.begin_interval_time,'yy-mm-dd hh24')  c1,
  count(1)                                         c4
from
   dba_hist_sql_plan p,
   dba_hist_sqlstat  s,
   dba_hist_snapshot sn,
   dba_segments      o
Where    p.object_owner <> 'SYS'
And      p.object_owner = o.owner
And      p.object_name = o.segment_name
and
   o.blocks > 1000
and
   p.operation like '%TABLE ACCESS%'
and
   p.options like '%FULL%'
group by
to_char(sn.begin_interval_time,'yy-mm-dd hh24')
```

Only return tables with More than 1,000 blocks (from DBA_SEGMENTS)

Track large-table full-table scans

Large Full-table scans
Per Snapshot Period

Begin Interval time	FTS Count
04-10-18 11	744
04-10-21 17	**2,364**
04-10-21 23	32
04-10-22 15	945
04-10-22 16	2
04-10-22 23	62

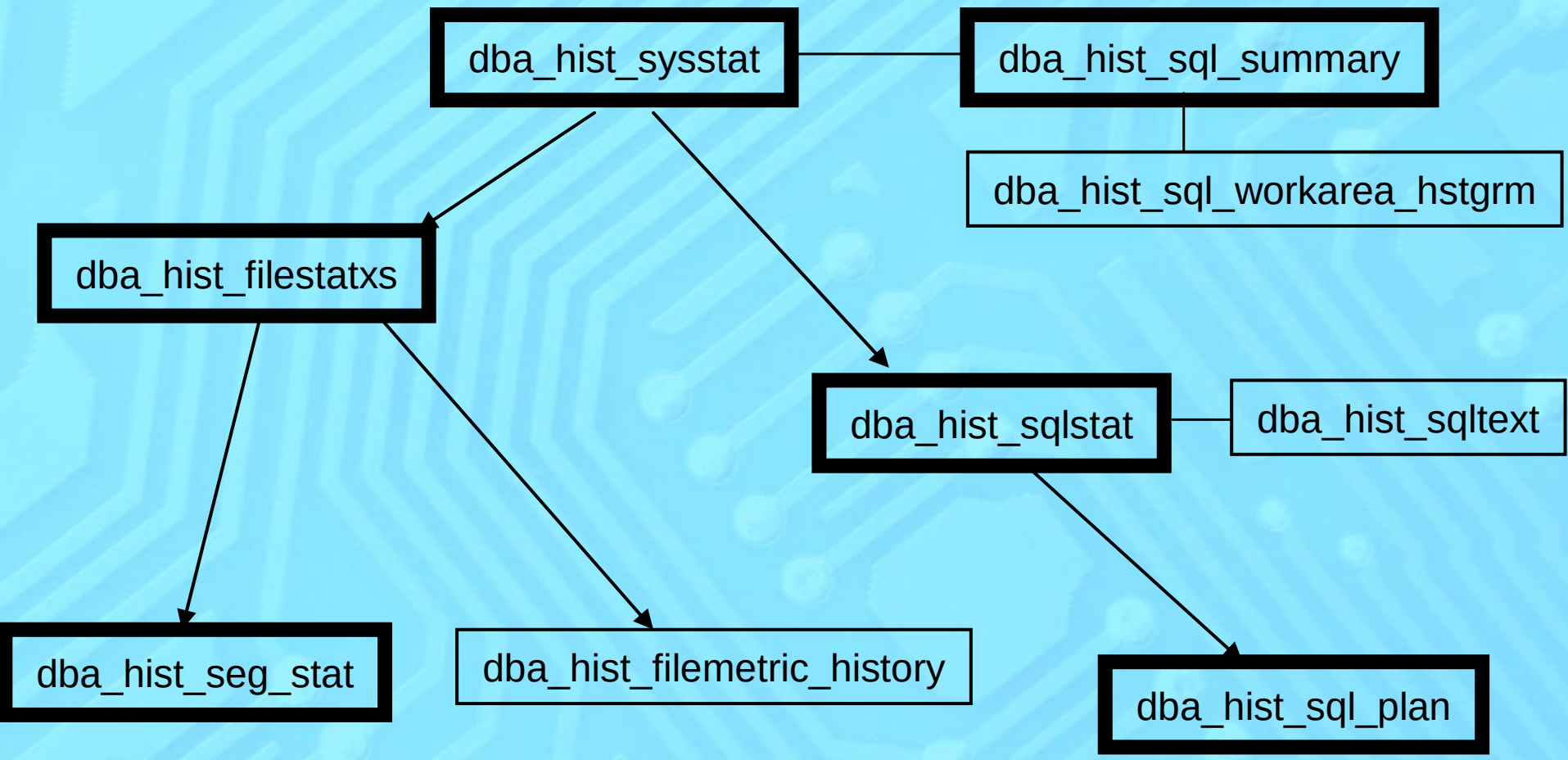
look for "exceptions" & "trends"

Oracle 10g automatic stats!

- Automatic histogram detection
- Automated sample size
- Automatic parallelism (when enabled)
- Automatic re-sampling (after 20% change)

```
execute dbms_stats.gather_schema_stats(
    ownname          => 'SCOTT',
    estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
    method_opt       => 'for all columns size skewonly',
    degree           => DBMS_STATS.DEFAULT_DEGREE);
```

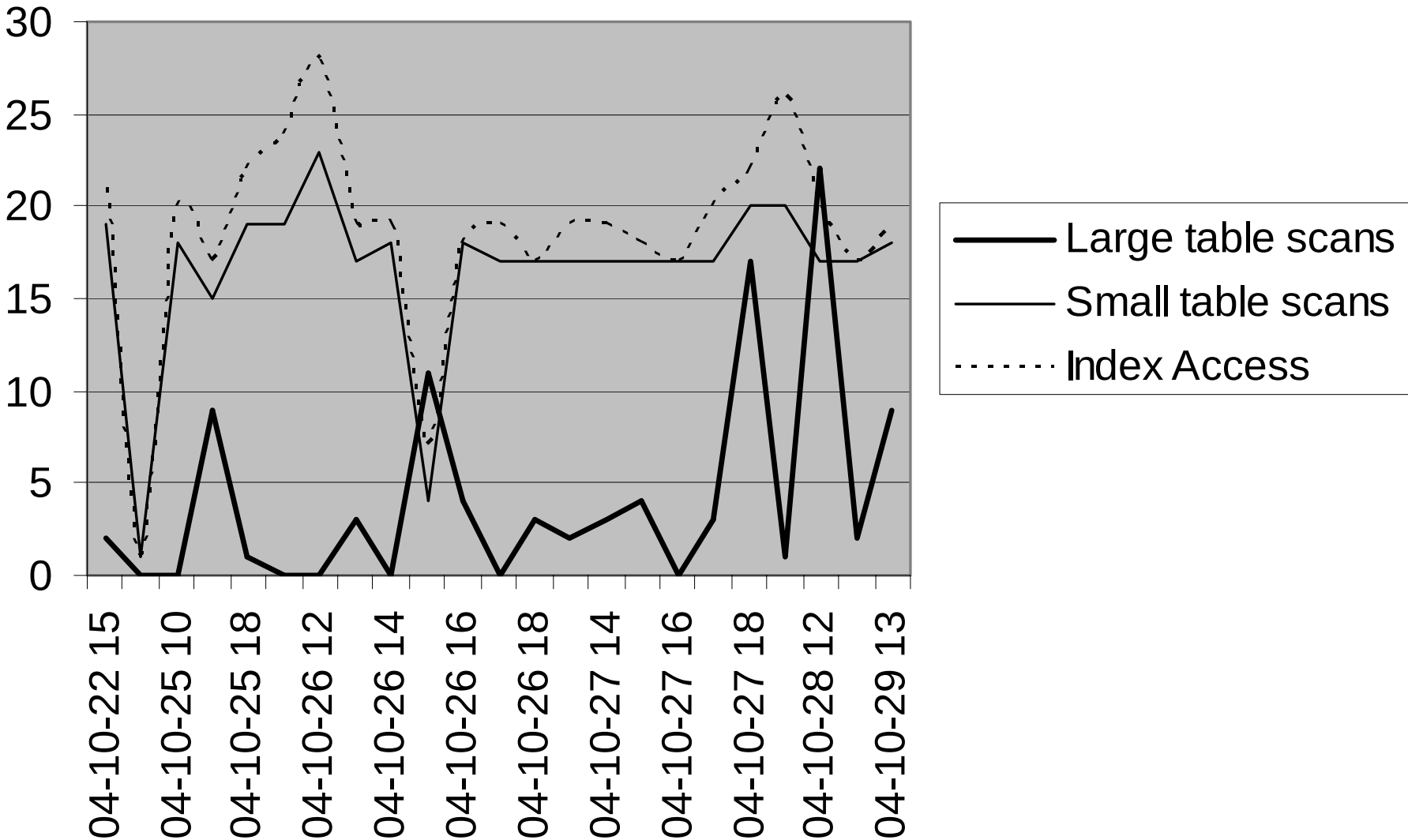
DBA_HIST Views



Inside DBA_HIST_SQLSTAT

Begin Interval time	Exec Delta	Buffer Gets Delta	Disk Reads Delta	IO Wait Delta	App Wait Delta	Cncr Wait Delta	CPU Time Delta	Elpsd Time Delta
10-14 10	709	2,127	0	0	0	0	398,899	423,014
10-14 11	696	2,088	0	0	0	0	374,502	437,614
10-14 12	710	2,130	0	0	0	0	384,579	385,388
10-14 13	693	2,079	0	0	0	0	363,648	378,252
10-14 14	708	2,124	0	0	0	0	373,902	373,902
10-14 15	697	2,091	0	0	0	0	388,047	410,605

Plot AWR data with MS Excel:



Show average SQL elapsed time:

```
select
   decode(c2,2,'Monday',3,'Tuesday',4,'Wednesday',5,'Thursday',6,'Friday',7,'S
   aturday',1,'Sunday') c2,
  c1,
  c3
from
(
select
   p.object_name                          c1,
   to_char(sn.end_interval_time,'d')      c2,
   count(1)                               c3
from
  dba_hist_sql_plan   p,
  dba_hist_sqlstat    s,
  dba_hist_snapshot  sn
where
  p.object_owner <> 'SYS'
and
  p.sql_id = s.sql_id
and
  s.snap_id = sn.snap_id
group by p.object_name, to_char(sn.end_interval_time,'d')
order by   c2,c1 );
```

Plotting Index Usage Over time!

Begin Interval time	Index Name	Disk Reads	Rows Processed
10-14 12	I_CACHE_STATS_1		114
10-14 12	I_COL_USAGE$	201	8,984
10-14 12	I_FILE1	2	0
10-14 12	I_IND1	93	604
10-14 12	I_JOB_NEXT	1	247,816
10-14 11	I_KOPM1	4	2,935
10-14 11	I_MON_MODS$_OBJ	12	28,498
10-14 11	I_OBJ1	72,852	604
10-14 11	I_PARTOBJ$	93	604
10-14 11	I_SCHEDULER_JOB2	4	0

How are your tables accessed?

Object Name	Operation	Option	Object Count
CUSTOMER	TABLE ACCESS	FULL	305
CUSTOMER_CHECK	INDEX	RANGE SCAN	2
CUSTOMER_ORDERS	TABLE ACCESS	BY INDEX ROWID	311
CUSTOMER_ORDERS		FULL	1
CUSTOMER_ORDERS_PRIMARY	INDEX	FULL SCAN	2
CUSTOMER_ORDERS_PRIMARY		UNIQUE SCAN	311
AVAILABILITY_PRIMARY_KEY		RANGE SCAN	4

How are your indexes used?

Invocation Counts for cust_index

Begin Interval time	Search Columns	Invocation Count
04-10-21 15	1	3
04-10-10 16	0	1
04-10-10 19	1	1
04-10-11 02	0	2
04-10-11 04	2	1
04-10-11 06	3	1
04-10-11 11	0	1
04-10-11 12	0	2
04-10-11 13	2	1
04-10-11 15	0	3
04-10-11 17	0	14

Latch Waits

```
select
    begin_interval_time,
    latch_name,
    gets,
    misses,
    sleeps
from
    dba_hist_latch
natural join
    dba_hist_snapshot
where
    (misses + sleeps ) > 0
order by    begin_interval_time,    misses
  DESC,    sleeps DESC
;
```

Latch Waits

BEGIN TIME	LATCH NAME	GETS	MISSES	SLEEPS
6 AM	library cache	4,451,177	856	943
	shared pool	3,510,651	482	611
	redo allocation	146,500	139	139
	cache buffers chains	13,050,732	52	104
	session allocation	8,176,366	43	43
	slave class create	2,534	41	41
	cache buffers lru chain	347,142	33	33
	row cache objects	2,556,877	24	26
7 AM	library cache	4,540,521	862	950
	shared pool	3,582,239	485	614
	redo allocation	149,434	140	140
	cache buffers chains	13,214,066	53	105
	session allocation	8,342,651	43	43

Wait stat delta

```
select
  begin_interval_time,
  new.stat_name,
  (new.value - old.value) “Difference”
from
   dba_hist_sys_time_model old,
   dba_hist_sys_time_model new,
   dba_hist_snapshot       ss
where
   new.stat_name = old.stat_name
and
   new.snap_id = ss.snap_id
and
   old.snap_id = ss.snap_id - 1
and
   new.stat_name like '%&stat_name%'
order by
   begin_interval_time;
```

Wait stat delta

BEGIN_INTERVAL_TIME	STAT_NAME	Difference
12-NOV-04 08.00.20.745 PM	hard parse elapsed time	10,605,028
12-NOV-04 09.00.48.205 PM	hard parse elapsed time	15,628,615
12-NOV-04 10.00.13.470 PM	hard parse elapsed time	54,707,455
12-NOV-04 11.00.41.412 PM	hard parse elapsed time	96,643,842
13-NOV-04 12.00.06.899 AM	hard parse elapsed time	16,890,047

Conclusions

- AWR will revolutionize Oracle tuning
- AWR includes SQL execution details
- The ASH contains detailed wait events
- Future releases of OEM may have predictive capability

Special thanks to Alexey Danchenkov for sharing the AWR scripts from his book “Oracle Tuning: The Definitive Reference”.